# Eingebettete dynamische Bayessche Netze *n*-ter Ordnung

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultät I der Universität des Saarlandes

vorgelegt von
Boris Brandherm

Saarbrücken
29. November 2006

**Datum des Kolloquiums:**
22.12.2006
**Dekan:**
Prof. Dr. Thorsten Herfet
**Vorsitzender:**
Prof. Dr. Joachim Weikert
**Gutachter:**
1. Prof. Dr. Dr. h. c. mult. Wolfgang Wahlster
2. Prof. Dr. Anthony Jameson
**Akademischer Beisitzer:**
Dr. Dominik Heckmann

# Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Diese Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Saarbrücken, den 29. November 2006

# Danksagung

Die vorliegende Arbeit entstand in den Projekten READY und BAIR des von der Deutschen Forschungsgemeinschaft geförderten Sonderforschungsbereichs 378 „Ressourcenadaptive kognitive Prozesse“ an der Universität des Saarlandes in Saarbrücken.

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr. Dr. h. c. mult. Wolfgang Wahlster, der es mir mit einer Anstellung an seinem Lehrstuhl ermöglicht hat, dieses interessante Thema im Rahmen einer Doktorarbeit in einem interdisziplinären Umfeld zu bearbeiten. Ich danke ihm für seine zahlreichen Anregungen und das Interesse, mit der er diese Arbeit begleitet hat.

Prof. Dr. Anthony Jameson danke ich für seine Unterstützung bei meinen Veröffentlichungen und Vorträgen. Ebenso danke ich ihm für eine Vielzahl von Vorschlägen und Tipps, die es mir ermöglichten, diese Arbeit in der vorliegenden Form zu erstellen. Von den Erfahrungen mit seiner wissenschaftlichen Arbeitsweise, die ich im Verlauf der Zusammenarbeit erlangt habe, werde ich sicherlich in Zukunft profitieren können.

Ich möchte allen Kollegen danken, die es mir ermöglicht haben, diese Arbeit zu realisieren. Allen Mitarbeitern und Studenten des Lehrstuhls gilt mein Dank für die angenehme und produktive Arbeitsatmosphäre.

Mein wichtigster Dank gilt meiner Familie, die mich in allen Belangen immer tatkräftig unterstützt hat.

# Kurzzusammenfassung

Das Ziel dieser Arbeit war die Konzeption, die Realisation und die Anwendung eines Systems, das eingebettete Systeme mit geringer Rechenleistung und wenig Arbeitsspeicher mit der Fähigkeit ausstattet, probabilistische Prozesse verarbeiten zu können.

Die Grundlage für dieses System bildet der differentielle Ansatz von Darwiche zur Lösung Bayesscher Netze, der ein Bayessches Netz in ein multivariates Polynom umwandelt und dann auswertet. Diesen Ansatz von Darwiche haben wir so erweitert, dass nun auch die speziellen Bedürfnisse dynamischer Bayesscher Netze berücksichtigt werden.

Aufbauend auf dieser theoretischen Ausarbeitung wurde eine Anwendung mit dem Namen *JavaDBN* entwickelt, die dynamische Bayessche Netze in spezielle Polynome umwandelt und für diese Quellcode generiert. Dieser Quellcode führt die Berechnungen für die Auswertung des Polynoms und das Anhängen neuer Zeitscheiben mit dem gleichzeitigen Rollup bei konstantem Speicherverbrauch durch.

Für die Modellierung dynamischer Bayesscher Netze spezifizieren wir neue Modellierungsstrukturen im Zusammenhang mit der Benutzermodellierung und der Sensorverarbeitung und führen damit den Begriff der dynamischen Bayesschen Netze *n*-ter Ordnung ein.

# Short Abstract

The aim of this work was the conception, realisation and application of a system that enables embedded systems with low computing power and memory to execute probabilistic processes.

The foundation of this system is the differential approach by Darwiche used to solve Bayesian networks, which converts a Bayesian network into a multivariate polynomial and then evaluates it. We extended this approach, such that it also fulfils the specialized requirements of dynamic Bayesian networks.

Based on this theoretical elaboration, an application called *JavaDBN* was developed to convert dynamic Bayesian networks into specific polynomials and generate their networks' source code. This source code executes the computations for the evaluation of the polynomials and for the addition of new time slices with simultaneous roll-up with constant space requirements.

To model dynamic Bayesian networks, we specified new modelling structures for the domains of user modelling and sensor processing and introduce the term of dynamic Bayesian networks of *n*-th order.

# Zusammenfassung

Das Ziel dieser Arbeit war die Konzeption, die Realisation und die Anwendung eines Systems, das eingebettete Systeme mit geringer Rechenleistung und wenig Arbeitsspeicher mit der Fähigkeit ausstattet, probabilistische Prozesse verarbeiten zu können.

Die Grundlage für dieses System bildet der differentielle Ansatz von Darwiche zur Lösung Bayesscher Netze, der ein Bayessches Netz in ein multivariates Polynom umwandelt und dann auswertet. Diesen Ansatz von Darwiche haben wir so erweitert, dass nun auch die speziellen Bedürfnisse dynamischer Bayesscher Netze berücksichtigt werden. Wir können im dynamischen Bayesschen Netz eine Vorwärts- und Rückwärtspropagierung und auch eine Kombination der beiden Verfahren durchführen. Alte Zeitscheiben und andere überflüssige Netzstrukturen können ohne Informationsverlust aufgerollt werden, wobei das Polynom bei konstantem Speicherverbrauch ausgewertet wird.

Aufbauend auf dieser theoretischen Ausarbeitung wurde eine Anwendung mit dem Namen *JavaDBN* entwickelt, die dynamische Bayessche Netze in spezielle Polynome umwandelt und für diese Quellcode (wahlweise Java oder C++) generiert. Dieser Quellcode führt die Berechnungen für die Auswertung des Polynoms (die Inferenz im dynamischen Bayesschen Netz) und das Anhängen neuer Zeitscheiben mit dem gleichzeitigen Rollup (dem Abschneiden der vorhergehenden Zeitscheibe) durch.

Der Quellcode wurde so realisiert, dass in der Initialisierungsphase alle notwendigen Variablen eingeführt werden, und danach während der Laufzeit kein Speicher mehr allokiert oder deallokiert werden muss. Somit wird auch keine automatische Speicherbereinigung benötigt, die gerade in Echtzeit-Systemen wie z. B. eingebetteten Systemen nicht toleriert werden kann, da der Zeitpunkt ihrer Durchführung oft nicht vorhergesehen werden kann, und dadurch die Programmausführung zu nicht voraussehbaren Zeitpunkten unterbrochen werden würde. Zusätzlich lässt sich dadurch der Speicherbedarf und die Laufzeit des Programmes nach oben abschätzen.

Ein weiterer großer Vorteil des Quellcodes liegt darin, dass er leicht lesbar ist, so dass das Verfahren der Inferenz und des Rollups für den Anwender transpa-

rent bleibt. Dadurch lassen sich Veränderungen im Quellcode vornehmen, so dass sich Zusatzfunktionen wie beispielsweise Datenbankzugriffe oder Methodenaufrufe realisieren lassen, die nicht oder nur sehr schwer möglich wären, wenn der Inferenzalgorithmus vorkompiliert ist und das dynamische Bayessche Netz zur Verarbeitung als Eingabe erhält. Dazu haben wir gezeigt, wie sich konstante Tabelleneinträge zu variablen Tabelleneinträgen umwandeln lassen, indem Arrays durch Methodenaufrufe ersetzt werden. Dies ist z. B. dann sinnvoll, wenn Tabelleneinträge nicht im Voraus bekannt sind oder sich zur Laufzeit verändern können. Ein Methodenaufruf könnte beispielsweise einen Datenbankzugriff realisieren (um beispielsweise die Tabelleneinträge aus einem Benutzerprofil einzulesen) oder von der abgelaufenen Zeit seit der letzten Instantiierung einer Zeitscheiben abhängig sein. Weiterhin erzeugt JavaDBN Quellcode für die Sensitivitätsanalyse und unterstützt das Werkzeug *MatLab* (siehe [MatLab 06]).

Von den Anwendungen, die mit *JavaDBN* erzeugt wurden, möchte wir an dieser Stelle den Alarm Manager und den multisensoriellen Positionierungsservice LORIOT erwähnen.

Anhand der prototypischen Anwendung Alarm Manager demonstrierten wir erstmals, wie physiologische Daten eines Benutzers durch physiologische Sensoren erfasst und durch ein dynamisches Bayessches Netz in Echtzeit auf einem persönlichen eingebetteten System interpretiert werden können, um dem Anwender Nachrichten (in diesem Fall eine wichtige Benachrichtigung) an seinen aktuellen Zustand angepasst zu präsentieren. Als Sensoren verwendeten wir einen Beschleunigungssensor, einen Muskeltätigkeitssensor und einen Augenbrauensensor.

In der Anwendung LORIOT führen wir als Innovation die sogenannten *georeferenzierten dynamischen Bayesschen Netze* (geoDBNs) ein, die es ermöglichen, dass die Benutzerposition auf dem persönlichen Handheld-Gerät berechnet werden kann, ohne dass eine externe Verbindung zu einem Server besteht, auf dem dann die Positionsberechnung durchgeführt würde. Diese neue Methode hat sowohl an Rechenzeit als auch Arbeitsspeicher nur einen geringen Ressourcenbedarf und ist hoch-skalierbar für die In- und Outdoor-Positionierung. Momentan werden IR- und RFID-Sensoren verwendet, aber ebenso lassen sich leicht andere Sensoren einbauen, indem das dynamische Bayessche Netz um den entsprechenden Sensorknoten erweitert wird.

Für die Modellierung dynamischer Bayesscher Netze spezifizieren wir neue Modellierungsstrukturen im Zusammenhang mit der Benutzermodellierung und der Sensorverarbeitung und führen damit den Begriff der dynamischen Bayesschen Netze *n*-ter Ordnung ein. Wir zeigen insbesondere im Zusammenhang mit der Sensorverarbeitung wie sich die Probleme lösen lassen, die durch die Latenz der Sensordaten auftreten können.

# Abstract

The aim of this work was the conception, realisation and application of a system that enables resource-limited embedded systems with low computing power and memory to execute probabilistic processes.

The foundation of this system is the differential approach by Darwiche used to solve Bayesian networks, which converts a Bayesian network into a multivariate polynomial and then evaluates it. We extended this approach, such that it also fulfils the specialized requirements of dynamic Bayesian networks. We can execute forward and backward propagation in a dynamic Baysian network and also a combination of both operations. All time slices and other dispensable network structures can be rolled-up without loss of information, whereby the polynomial can be evaluated with constant space requirements.

Based on this theoretical elaboration, an application called *JavaDBN* was developed to convert dynamic Bayesian networks into specific polynomials and generate their networks' source code selectively, in either Java or C++.

This source code executes the computations for the evaluation of the polynomials (the inference in dynamic Bayesian networks) and for the addition of new time slices with simultaneous roll-up (elimination of the previous time slice).

The source code is realised in a way that during the initialisation phase all required variables are instantiated and no further memory has to be allocated or deallocated afterwards. Thus, garbage collection is not required –a feature that is, especially in embedded real-time systems, not tolerable, since the timing of its execution is not predictable and could therefore interrupt the program execution at unpredictable points in time.

Additionally, memory requirements and run-time maxima of the application can be estimated.

A further benefit of the source code is its easy readibility, such that inference and roll-up processes remain transparent to the user. Thereby it is feasible to adjust the source code directly and realise features that would only be hardly possible or not at all possible to incorporate, if the inference algorithm would be a precompiled system, accepting and processing the dynamic Bayesian network only as a static input. For this reason we showed how constant table entries can be trans-

formed into variable table entries, by replacing arrays with method calls. This is for instance useful whenever table entries are not known in advance or are likely to change at runtime. A method-call could for example realize a database access (in order to read a table entry from a user profile for instance) or depend on the time that has passed since the last instantiation of a time slice. *JavaDBN* further generates source code for sensitivity analysis and supports the tool *MatLab* (see [MatLab 06]).

Of all applications that have been created with *JavaDBN* we would now like to mention the Alarm Manager and the positioning service LORIOT.

With the prototypical application Alarm Manager, we demonstrated for the first time, how physiological data of a user was measured by physiological sensors and interpreted in real-time by a dynamic Bayesian network on a personal mobile device in order to present messages to the user (in our example an urgent notification) that adapt to her or his current state. We utilised an acceleration sensor, a muscle activity sensor and an eyebrow sensor.

Within the application LORIOT, we introduced the innovation of the so called georeferenced dynamic Bayesian networks (geoDBNs), which allows for the computation of user positions on a personal handheld device without the need for a connection to an external server, which operates the positioning service. This new method demands only very low resource requirements on both processing power and memory and is highly scalable for in- and outdoor positioning. Currently, infrared and RFID sensors are used, but other sensor technologies could be easily incorporated as well by extending the dynamic Bayesian network with an appropriate sensor node.

To model dynamic Bayesian networks, we specified new modelling structures for the domains of user modelling and sensor processing and introduce the term of dynamic Bayesian networks of $n$-th order. We show in particular, how problems that can arise due to the latency of sensor data can be solved.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Kapitel 1

# Vorwort

## 1.1 Einleitung und Motivation

Im folgenden zeigen wir an einem Beispiel aus der Automobilbranche einige Möglichkeiten auf, die eingebettete dynamische Bayessche Netze *n*-ter Ordnung bieten.

Zuerst wurden Lastkraftwagen und die Wagen der Oberklasse aber zunehmend werden nun auch Wagen der Mittelklasse mit Assistenzsystemen ausgestattet, die den Fahrer beim Fahren entlasten sollen und beispielsweise in kritischen Situationen automatisch abbremsen oder die Spur halten. Dazu ist das Auto mit *Sensoren* bestückt, die über das Fahrzeug und sein Umfeld wachen.

Beispielsweise warnt ein System bei Temperaturen um den Gefrierpunkt den Fahrer vor einer möglicherweise vereisten Fahrbahn. Dabei wird die Außentemperatur über einen vor Fahrtwind geschützten Temperatursensor erfasst. Ein *Scheibenwischerassistenzsystem* steuert die Scheibenwischer in Abhängigkeit der Regenmenge, die von den Regensensoren gemessen wird, die in der Frontscheibe eingebaut sind. Ein *intelligenter Tempomat* regelt automatisch den Abstand zu vorausfahrenden Fahrzeugen. Dazu sind Sensoren im Fahrzeug eingebaut, die permanent den Abstand zum vorausfahrenden Fahrzeug messen. Der Tempomat hält die gewählte Wunschgeschwindigkeit, bis ein Fahrzeug eingeholt wird. Das System reduziert dann automatisch das Tempo und hält die festgelegte Wunschdistanz ein. Sobald die Strecke wieder frei ist, wird selbstständig auf die gewählte Geschwindigkeit beschleunigt. Ein *Spurassistent* erkennt durch spezielle Sensoren, wenn von den Fahrbahnmarkierungen abgewichen wird, und macht den Fahrer beispielsweise über Vibrationen im Sitz darauf aufmerksam. Ein *Bremsassistent* erkennt an der Geschwindigkeit, mit der auf das Bremspedal getreten wird, eine Notbremsung und baut automatisch den Maximaldruck im System auf. Beim *adaptiven Kurvenlicht* erfassen Sensoren den Lenkwinkel, die Gierrate (d. h.

die Drehgeschwindigkeit um die Hochachse) und die Fahrgeschwindigkeit. Die Scheinwerfer sind horizontal schwenkbar und werden so gesteuert, dass sie den Kurven ihrem Verlauf entsprechend folgen und sie somit besser ausleuchten. Dies sorgt für mehr Fahrsicherheit bei Dunkelheit und schlechten Lichtverhältnissen. Ein *Telematiksystem* im Zusammenhang mit einem *Navigationssystem* gibt Auskunft über schwierige Straßensituationen wie eine Baustelle oder einen Stau und berechnet automatisch eine neue Route, um beispielsweise den Stau zu umfahren, wenn es möglich ist. Ein *Displayassistent* misst über Lichtsensoren das einfallende Umgebungslicht und passt die Display-Beleuchtung an die aktuelle Gegebenheit an.

Bei allen Assistenzsystemen ist es aber wichtig, dass der Fahrer nicht entmündigt wird und die Kontrolle über das Fahrzeug behält. Einige der oben genannten Assistenzsysteme gehören in den Bereich der sogenannten *aktiven Sicherheit* und dienen der *Unfallvermeidung*, indem sie dem Fahrer bei lästigen Aufgaben entlasten (z. B. der Scheibenwischerassistent) oder beispielsweise in kritischen Situationen automatisch abbremsen (wie z. B. der intelligente Tempomat). Der Sicherheitsgurt gehört in den Bereich der sogenannten *passiven Sicherheit* und dient der *Unfallfolgenminderung*.

Betrachten wir das Beispiel, dass sich eine Person mit dem Auto auf dem Weg zur Arbeit befindet. Es regnet leicht, und die Temperatur liegt um den Gefrierpunkt. Im Bereich der Autobahnauffahrt, die sie täglich auf dem Weg zu ihrer Arbeit nutzt, befindet sich seit heute eine Baustelle. Sie ist gerade dabei, sich im Reißverschluss-Prinzip in den zäh fließenden Verkehr einzuordnen, als ihr Handy klingelt. Offensichtlich sollte sie sich ganz und gar der aktuellen Verkehrssituation widmen. Selbst wenn sie nicht ans Handy geht, so wurde sie jedoch durch das Klingeln abgelenkt. Ein wichtiger Aspekt bei der Unfallvermeidung ist auch, dass der Fahrer, wenn er durch eine schwierige Fahrsituation stark beansprucht ist, beispielsweise durch einen eingehenden Telefonanruf nicht von der aktuellen Verkehrssituation abgelenkt wird.

Wenn man die Sensorinformationen der Sensoren der bereits vorhandenen Assistenzsysteme intelligent miteinander verknüpfen würde, so könnte man hieraus auch *Schlüsse* auf die aktuelle Fahrerbeanspruchung ziehen. Ein entsprechendes *Telefonassistenzsystem* würde den eingehenden Telefonanruf unterdrücken und zu einem geeigneten Zeitpunkt den Anrufer automatisch zurückrufen, um im obigen Beispiel des ablenkenden Telefonanrufes zu bleiben. Beim adaptiven Kurvenlicht werden verschiedene Sensoren ausgewertet, um den optimalen Schwenkwinkel der Scheinwerfer zu bestimmen. Bei Geschwindigkeiten bis ca. 30 km/h lässt sich der gefahrene Kurvenradius und damit der Schwenkwinkel durch eine einfache Funktion [1] bestimmen, wobei der Radstand des Fahrzeuges und der Lenkwinkel

[1] Wenn wir mit $d$ den Radstand des Fahrzeuges und mit $\delta$ den Lenkwinkel der Vorderräder

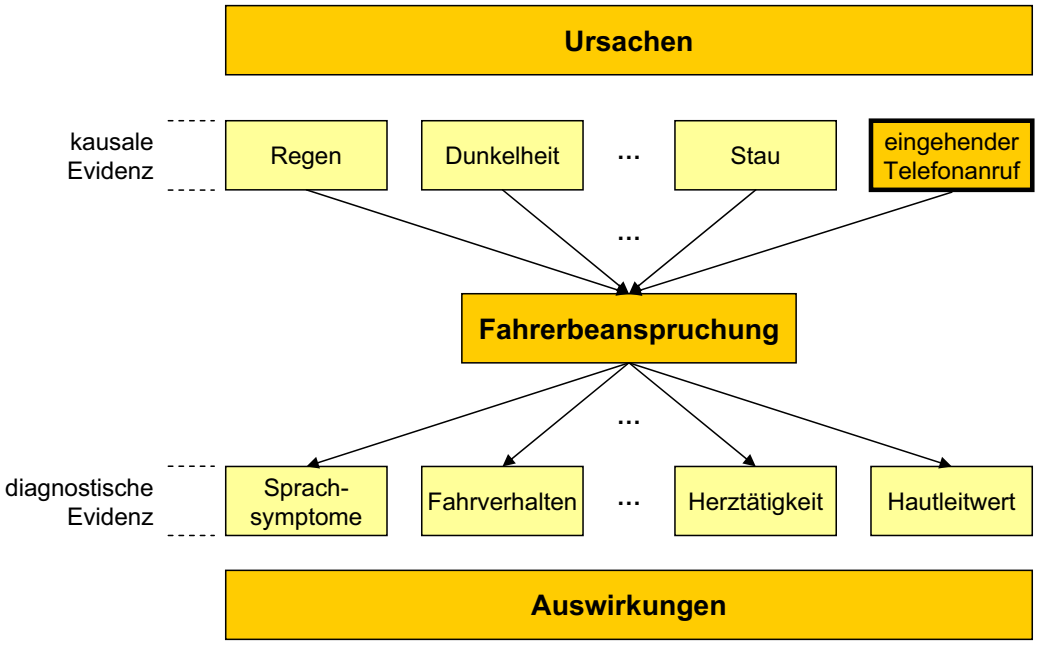



Abbildung 1.1: Anhand kausaler als auch diagnostischer Evidenz kann die Fahrerbeanspruchung eingeschätzt werden.

der Vorderräder in die Berechnung einfließen. Bei schnelleren Fahrten als 30 km/h wird der Kurvenradius mit Hilfe der Gierrate berechnet. Aber auch hier ist der Einfluss der verschiedenen Faktoren zur Bestimmung des Schwenkwinkels des Scheinwerfers genau bekannt.

Im Gegensatz dazu wären beim Telefonassistenzsystem der Einfluss der verschiedenen Faktoren wie Wetter- und Verkehrsverhältnisse auf die Fahrerbeanspruchung mit *Unsicherheiten* bzw. *Wahrscheinlichkeiten* behaftet. Es wird also ein Verfahren benötigt, mit dem sich unsicheres Wissen modellieren und verarbeiten lässt.

Bei der Einschätzung der Fahrerbeanspruchung, wie sie zuvor geschildert wurde, sind die aktuellen Sensordaten ausreichend, und es müssen keine vorhergehenden Sensordaten berücksichtigt werden. Im folgenden werden wir aber sehen, dass vorhergehende Sensordaten wichtig sein können und berücksichtigt werden müssen, um die Fahrerbeanspruchung adäquat einzuschätzen.

Auch wenn keine kritische Verkehrssituation vorliegt bzw. die Verkehrssituation durch die Sensoren als nicht kritisch eingeschätzt wird, kann ein Fahrer stark beansprucht sein. Um dieses feststellen zu können, müsste der Fahrer direkt beobachtet werden. Dabei lassen sein Verhalten und seine biophysiologischen Daten

bezeichnet, so lautet die Funktion zur Berechnung des Kurvenradius $R$: $R = d/\sin(\delta)$.

*Rückschlüsse* auf seine Beanspruchung zu. Eine erhöhte Beanspruchung des Fahrers würde beispielsweise zu einer in ihren Merkmalen veränderten Herztätigkeit führen. Bei diesen Daten sind auch die vorhergehenden Daten zu berücksichtigen, um daraus verlässliche Rückschlüsse über die aktuelle Fahrerbelastung ziehen zu können. Sensoren, die z. B. im Sitz und im Lenkrad eingebaut sind, erfassen Biosignale wie den Hautleitwert, die Herztätigkeit und die Atemfrequenz, und wiederum andere Sensoren überwachen das Geschehen im Fahrzeuginnenraum.

In Abbildung 1.1 ist dargestellt, wie die Wetter- und Verkehrssituation und der einkommende Telefonanruf die *Ursachen* der erhöhten Fahrerbeanspruchung sind. Die Informationen darüber bezeichnet man als *kausale Evidenz*. Das symptomatische Verhalten des Fahrers und seine biophysiologischen Daten sind beobachtbare *Auswirkungen* seiner Beanspruchung, und die Informationen darüber bezeichnet man als *diagnostische Evidenz*.

Die Sensorfusion ist ein weites Forschungsfeld und ist beispielsweise in der Fahrzeugtechnik erst noch eine Vision der Entwickler. Es müssen die kausalen Zusammenhänge zwischen den einzelnen Faktoren wie Wetter- und Verkehrsbedingungen, Fahrerbeanspruchung, Herztätigkeit, Hautleitwert und symptomatisches Verhalten qualitativ und quantitativ erforscht und in der passenden Form modelliert werden.

## 1.2 Einordnung

Zur Modellierung und Verarbeitung von unsicherem Wissen existieren verschiedene Verfahren wie *Inferenznetze* (s. [Duda et al. 76]), *Bayessche Netze* (siehe [Pearl 88]), *Dempster-Shafer-Theorie* oder *Fuzzy-Logik* ([Russell & Norvig 03]). Sie berücksichtigen allerdings nicht die zeitliche Komponente, und somit lässt sich mit ihnen die Einschätzung der Fahrerbeanspruchung nur dann adäquat berechnen, wenn über die Ursachen der Fahrerbeanspruchung wie *Regen* oder *Stau* aber nicht über die Auswirkungen der Fahrerbeanspruchung wie *symptomatisches Verhalten* oder *Hautleitwert* Evidenz vorliegt. Denn, wenn auch die Daten aus den Biosensoren in die Einschätzung einfließen sollen, so müssen neben den aktuellen auch die vorhergehenden Daten berücksichtigt werden.

Um probabilistisch auch über die *Zeit* schließen zu können, existieren beispielsweise *Hidden Markov Modelle*, *Kalman-Filter* oder *dynamische Bayessche Netze*. Wie in [Russell & Norvig 03] gezeigt wird, sind Hidden Markov Modelle und Kalman-Filter Spezialfälle der dynamischen Bayesschen Netze. Jedes Hidden Markov Modell lässt sich als dynamisches Bayessches Netz betrachten, und jedes dynamische Bayessche Netz kann in ein Hidden Markov Modell umgewandelt werden. Das dynamische Bayessche Netz kann im allgemeinen platzsparender als das entsprechende Hidden Markov Modell modelliert werden und benötigt ent-

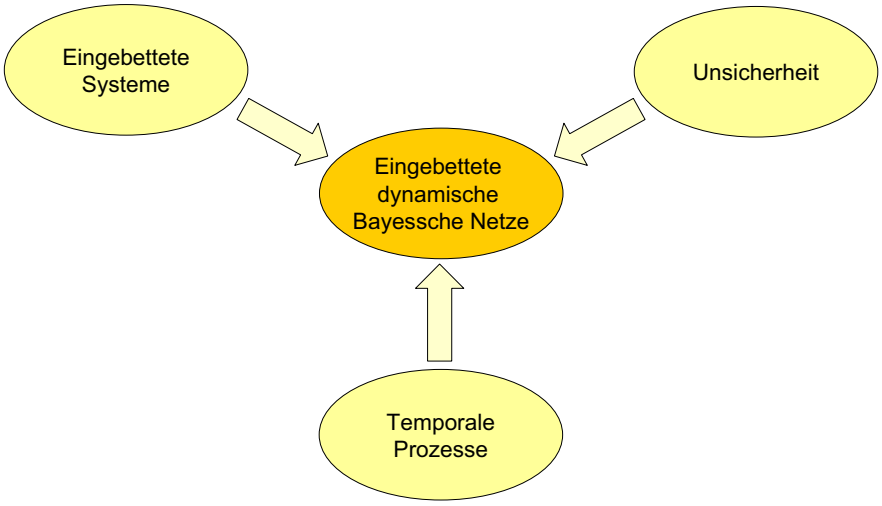



Abbildung 1.2: Anforderungen an das zu entwickelnde System.

sprechend weniger Rechenleistung bei der Auswertung. Ebenso ist es möglich, jeden Kalman-Filter als ein dynamisches Bayessches Netz zu modellieren. Umgekehrt lässt sich aber nicht jedes dynamische Bayessche Netz als ein Kalman-Filter darstellen.

Dynamische Bayessche Netze sind von diesen drei zuvor genannten Verfahren die mächtigste und effizienteste Methode in der Modellierung und Verarbeitung unsicheren Wissens. Sie sind eine Erweiterung der Bayesschen Netze. Damit alle Messungen berücksichtigt werden können, wird mit jeder Messung an das dynamische Bayessche Netz ein Bayessches Netz (sogenannte Zeitscheibe) angehängt. So werden sowohl vorhergehende Zustände als auch neue Evidenz berücksichtigt. Indem mit jeder Messung eine Zeitscheibe an das dynamische Bayessche Netz angehängt wird, steigt die Rechenkomplexität des dynamischen Bayesschen Netzes an, und es müssen Verfahren angewendet werden, die alte Zeitscheiben abschneiden, aber deren Einfluss auf die verbleibenden Zeitscheiben des dynamischen Bayesschen Netzes erhalten. Solche Verfahren bezeichnet man als *Rollup*-Verfahren.

Nicht nur aber auch gerade im Automobilbereich ist es wichtig, dass ein System stabil ist bzw. fehlerfrei läuft und so wenig wie möglich an Rechenzeit und Arbeitsspeicher benötigt. Denn jede zusätzlich verbaute Komponente oder eine größere Komponente, die verbaut wird, damit das System laufen kann, bedeutet weniger Platz und mehr Gewicht im Auto, was dazu führen kann, dass andere Systeme nicht eingebaut werden können. Zusätzlich wird beim Fahren mit jedem Kilogramm mehr im Fahrzeug auch mehr Kraftstoff verbraucht.

Zusammenfassend benötigen wir ein Verfahren, mit dem sich dynamische

probabilistische Prozesse modellieren und ressourcenschonend verarbeiten lassen (siehe Abbildung 1.2).

Mit Software-Paketen wie Hugin ([Hugin 04]), Netica (siehe [Netica 06]) oder JavaBayes (siehe [Cozman 01]) lassen sich Bayessche Netze modellieren und verarbeiten. Sie erhalten als Eingabe ein Bayessches Netz, das von ihnen in eine Struktur umgewandelt wird, auf der gerechnet wird. Dynamische Bayessche Netze werden von ihnen allerdings nicht ohne weiteres verarbeitet. Außer für Hugin existiert ein Aufsatz, genannt dHugin (siehe [Kjærulff 95]), mit dem sich dynamische Bayessche Netze verarbeiten lassen. Bei allen diesen Systemen wird das (dynamische) Bayessche Netz eingelesen, für das durch komplexe Berechnungen eine weitere (sekundäre) Struktur bestimmt wird, auf der erst gerechnet werden kann. D. h. dass sowohl das (dynamische) Bayessche Netz, als auch die sekundäre Struktur und eventuell weitere Konstrukte, die zur Bestimmung der sekundären Struktur notwendig waren, im Speicher verwaltet werden müssen. Zusätzlich wird auch immer noch das System benötigt, um auf der sekundären Struktur zu rechnen.

Unsere Idee ist es nun, ein Verfahren zu entwickeln, das dynamische Bayessche Netze in einen Quellcode umwandelt, der selbständig ohne einem großen Programmpaket ausgeführt werden kann (sozusagen wie selbstentpackende Zip-Dateien, die ihren Code zum Auspacken selbst mitbringen).

Ein vielversprechendes Verfahren im ressourcenschonenden Umgang bei der Verarbeitung Bayesscher Netze ist der Ansatz von Darwiche, bei dem Bayessche Netze in multivariate Polynome umgewandelt werden. Diese Polynome dienen dann als Eingabe für ein einfaches Verfahren, das die Polynome entsprechend der Anfragen auswertet.

## 1.3 Ziel und Gegenstand der Arbeit

In der vorliegenden Arbeit wollen wir den differentiellen Ansatz von Darwiche zum Lösen Bayesscher Netz so erweitern, dass auch die speziellen Bedürfnisse dynamischer Bayesscher Netze wie das Anhängen neuer Zeitscheiben oder das Aufrollen alter Zeitscheiben ohne Informationsverlust berücksichtigt werden.

Dazu sollen die entsprechenden theoretischen Grundlagen geschaffen und in einer Anwendung umgesetzt werden, die dynamische Bayessche Netze in Polynome umwandelt und insbesondere für diese Polynome einen Quellcode generiert, der die Berechnungen für die Auswertung des Polynoms (die Inferenz im dynamischen Bayesschen Netz) und den Rollup (das Abschneiden der vorhergehenden Zeitscheibe) ressourcenschonend durchführt.

Die Polynome, die aus den dynamischen Bayesschen Netzen bestimmt werden, sollen nicht mehr als Eingabe für einen Algorithmus dienen, sondern wir

wollen stattdessen das Polynom und den Algorithmus ineinander verschmelzen. Dadurch versprechen wir uns vielfältige Möglichkeiten, die bei einer Trennung von Polynom und Algorithmus nur sehr schwer umzusetzen wären. Dazu gehört beispielsweise die Parametrisierung des dynamischen Bayesschen Netzes, so dass beispielsweise die Wahrscheinlichkeiten aus einer Datenbank herausgelesen werden können, wenn sie zum Zeitpunkt der Modellierung noch nicht vorgelegen haben oder von anderen Umständen abhängig sind, die nicht im dynamischen Bayesschen Netz modelliert sind. Dazu wollen wir einige Anwendungen entwickeln, an denen wir die Parametrisierung und ihre Vorzüge demonstrieren wollen.

Zusätzlich wollen wir einen Positionierungsservice für ein eingebettetes System realisieren, in dem unter Berücksichtigung der zeitlichen Komponente aller eingehenden Messungen sowohl fehlerbehaftete Sensordaten verarbeitet als auch die Sensordaten der verschiedenen Sensortypen fusioniert werden.

Zusammenfassend sollen in der vorliegenden Arbeit die folgenden Fragen behandelt werden:

- Wie lässt sich der differentielle Ansatz zur Lösung Bayesscher Netze für dynamische Bayessche Netze erweitern?

  Die zu bestimmenden Polynome müssen die speziellen Bedürfnisse der dynamischen Bayesschen Netze wie das Anhängen neuer Zeitscheiben oder das Aufrollen alter Zeitscheiben ohne Informationsverlust ermöglichen.

- Wie lassen sich dynamische Bayessche Netze effizient auf Geräten auswerten, die nur einen minimalen Speicherplatz oder eine geringe Rechenleistung aufweisen?

  Bei eingebetteten Systemen mit wenig Ressourcen wird oftmals kein *Betriebssystem* oder nur ein Betriebssystem *ohne Speicherschutz* eingesetzt, so dass sich der Entwickler um die Speicherverwaltung kümmern muss. Optimal wäre ein Programmcode, der das dynamische Bayessche Netz verarbeitet und während der Laufzeit keinen Speicher allokiert oder deallokiert.

- Wie lassen sich dynamische Bayessche Netze parametrisieren, so dass die mit den Knoten des dynamischen Bayesschen Netzes behafteten Wahrscheinlichkeiten erst zur Laufzeit vorliegen müssen?

  Durch eine Parametrisierung des dynamischen Bayesschen Netzes ließen sich die bedingten Wahrscheinlichkeiten zur Laufzeit berechnen bzw. aus einer Datenbank einlesen.

- Wie lässt sich in dynamischen Bayesschen Netzen (a) der Einfluss einzelner Knoten und (b) der Einfluss mehrerer Knoten auf die anderen Knoten im Netz bestimmen?

Man bezeichnet dies als *Sensitivitätsanalyse* in dynamischen Bayesschen Netzen. Mit ihr lässt sich beispielsweise feststellen, welche Wahrscheinlichkeiten, die mit den Knoten behaftet sind, besonders genau modelliert werden müssen.

- Wie lassen sich biophysiologische Signale zur Einschätzung der Ressourcenlage eines Benutzers auf Geräten mit minimalem Speicherplatz oder geringer Rechenleistung in Echtzeit auswerten?

  Wie wir schon in der Einleitung mit dem Fahrerassistenzsystem gesehen haben, können physiologische Daten wertvolle Hinweise auf die aktuelle Ressourcenlage des Benutzers geben.

- Wie lässt sich der Modellierungsprozess der dynamischen Bayesschen Netze vereinfachen und ihre Darstellung übersichtlicher gestalten?

  In heutigen Systemen muss der Entwickler darauf achten, dass bei der Modellierung dynamischer Bayesscher Netze bestimmte Voraussetzungen erfüllt werden. Werden diese Voraussetzungen verletzt, so kann kein Rollup durchgeführt werden. Hier wäre beispielsweise ein System wünschenswert, dass die Modellierungsfehler anzeigt und gegebenenfalls automatisch behebt.

- Wie lassen sich zusammengehörende Signale gemeinsam auswerten, die zeitlich verzögert eintreffen (Latenz der Signale)?

  Bei Sensordaten gibt es das Problem der Latenz, der zeitliche Abstand zwischen dem dargebotenen Reiz und eine darauf messbare Reaktion. Wenn die Latenz so groß ist, dass in der Zwischenzeit schon wieder mehrere Zeitscheiben instantiert wurden, bevor die messbare Reaktion gemessen wird, so können die zusammengehörenden Signale nie gemeinsam ausgewertet werden. Lässt sich dieses Problem auf einfache Art und Weise lösen?

- Wie können dynamische Bayessche Netze genutzt werden, um die Position eines Anwenders aufgrund von Sensordaten zu bestimmen, die das mobile Gerät empfängt? Wie lässt sich dieses Verfahren auf einem eingebetteten System mit wenig Ressourcen realisieren?

  Für ein weites Anwendungsspektrum im Bereich des *Pervasive Computing* ist es wichtig, die Position des Anwenders zu kennen. Es wurden verschiedene Techniken entwickelt, um das Problem von Unsicherheit, verrauschten Sensordaten und der Sensorfusion zu bewältigen. Eine davon sind dynamische Bayessche Netze.

## 1.4 Gliederung

Im Anschluss an diese Einleitung werden in **Kapitel 2** anhand zweier Beispielnetze die Definition Bayesscher Netze und die dazugehörige Notation eingeführt sowie zwei grundlegende Lösungsverfahren vorgestellt. Ebenso zweigen wir, wie sich Bayessche Netze als Polynom darstellen und auswerten lassen, wobei die zwei zuvor vorgestellten Lösungsverfahren gewinnbringend für die kompakte Darstellung des Polynomes genutzt werden können.

In **Kapitel 3** behandeln wir dynamische Bayessche Netze. Dazu führen wir die Begriffe und Schreibweisen für diese Netze ein. Wir stellen grundlegende Überlegungen an, wie es sich auswirkt, wenn Strukturen direkt in Bayesschen Netzen gelöscht werden, und wie sich Zeitscheiben in dynamischen Bayesschen Netzen effizient anhängen lassen. Im Anschluss daran vergleichen wir systematisch einige Rollup-Verfahren miteinander.

In **Kapitel 4** erweitere ich die Idee der differentiellen Herangehensweise von Darwiche und zeige, wie sich eine Polynom-Propagierung in dynamischen Bayesschen Netzen durchführen lässt. Dabei stelle ich sowohl die Vorwärts- als auch die Rückwärtspropagierung vor und wie sich eine Approximation realisieren lässt.

In **Kapitel 5** erweitere ich die dynamischen Bayesschen Netze zu dynamischen Bayesschen Netzen *n*-ter Ordnung, indem ich besondere Modellierungsstrukturen einführe, die nicht nur in der Benutzermodellierung oder der Sensorverarbeitung als sinnvoll erscheinen. Diese Modellierungsstrukturen werden allgemeingültig beschrieben, so dass sie auch in anderen Domänen eingesetzt werden können. Mit ihnen soll der Modellierungprozess der dynamischen Bayesschen Netze vereinfacht und deren Darstellung übersichtlicher gestaltet werden können. Man wird sehen, dass diese neuen Modellierungsstrukturen die direkte Anwendung der Rollup-Verfahren aus Kapitel 3 verhindern, aber dass im allgemeinen durch graphentheoretische Umformungen der Rollup wieder ermöglicht wird.

Im darauffolgenden **Kapitel 6** stellen wir die Anwendung JavaDBN vor, die dynamische Bayessche Netze in Polynome umwandelt und insbesondere für diese Polynome einen Quellcode generiert, der die Berechnungen für die Auswertung des Polynoms und den Rollup ressourcenschonend durchführt.

In **Kapitel 7** stellen wir eine Auswahl der Anwendungen vor, die mit JavaDBN entwickelt wurden.

**Kapitel 8** beschließt die Arbeit mit einer *Zusammenfassung* der erzielten Ergebnisse und einem Ausblick, in dem offene Fragestellungen angesprochen werden, die sich aus dieser Arbeit ergeben und weiter untersucht werden können.

# Kapitel 2

# Bayessche Netze

In diesem Kapitel werden wir zunächst zwei kleine Beispielnetze vorstellen und anhand von ihnen die Begriffe und Schreibweisen für Bayessche Netze einführen. Daraufhin werden verschiedene Lösungsverfahren für Bayessche Netze vorgestellt. Diese Lösungsverfahren werden in einem späteren Kapitel so angepasst oder weiterentwickelt, dass ein sogenannter *Rollup* von dynamischen Bayesschen Netzen ausgeführt werden kann.

Mit Bayesschen Netzen lässt sich unsicheres Wissen modellieren und verarbeiten. Die in den Bayesschen Netzen modellierten Abhängigkeiten zwischen Aussagen sind dabei mit Wahrscheinlichkeiten behaftet. Mit Bayesschen Netzen lassen sich auch dann Schlüsse ziehen, wenn nicht für alle Aussagen Evidenzen eingetragen werden.

Sie finden ihren Einsatz in den unterschiedlichsten Bereichen, wie beispielsweise in der Medizin, der Robotik und der Verbrechensbekämpfung (BayesNet-Crime) oder zur Diagnose von Mobilfunkzellen. Diese Beispiele sind Anwendungen, wo nicht für jede Aussage Evidenz eingetragen werden kann, da beispielsweise noch ein Bluttest auf sich warten lässt, oder nur unzureichende Indizien gesammelt werden konnten. In der *Bioinformatik* werden Bayessche Netze zur Modellierung zellulärer Prozesse verwendet, um die Wirkungsweise zellulärer Prozesse besser zu verstehen (siehe [Friedman 04]).

Es existieren verschiedene Lösungsverfahren, um Bayessche Netze zu verarbeiten, die sich beispielsweise in *exakte* und *approximative* Lösungsverfahren einteilen lassen (siehe [Neapolitan 90, Pearl 88]). Zu den approximativen Verfahren gehören zum Beispiel *stochastische Simulation*, *Markov Chain Monte Carlo* Methoden, oder *loopy belief propagation* (siehe [Russell & Norvig 03]). Exakte Verfahren sind beispielsweise *Clustering*, *Conditioning* und *Junction Tree Verfahren*.

Wir wollen zwei exakte Verfahren vorstellen, die zur Kategorie der *Junction-Tree-Verfahren* zählen und mit eine Grundlage für das Verfahren sind, das wir

in Kapitel 4 vorstellen werden. Diese Verfahren berechnen aus der Struktur des Bayesschen Netzes eine sekundäre Struktur, den sogenannten *Junction Tree*, auf dem dann die Berechnungen[1] durchgeführt werden können. Die Bestimmung eines Junction Trees wird Schritt für Schritt präsentiert. Jeder Schritt wird dabei mit der notwendigen Begriffsbildung sowie weiterführendem Hintergrundwissen ebenso allgemeingültig wie auch an einem Beispielnetz dargestellt.

Nachdem der Junction Tree konstruiert ist, müssen die quantitativen Zusammenhänge aus dem Bayesschen Netz in den Junction Tree konsistent übertragen werden. Dazu wird eine Schreibweise eingeführt, die die mathematischen Gesichtspunkte der quantitativen Zusammenhänge außer acht lässt[2]. Dadurch kann man sich auf die Berechnungen konzentrieren, die bei der Inferenz im Junction Tree durchgeführt werden.

Anhand von zwei Beispielnetzen werden nun die Bayesschen Netze informell mit der entsprechenden Notation eingeführt.

## 2.1 Informelle Einführung der Bayesschen Netze

Bayessche Netze dienen zur Repräsentation und zur Verarbeitung von unsicherem Wissen. In ihnen werden kausale Zusammenhänge zwischen einzelnen Propositionen, grafisch dargestellt. Dieses wird in den beiden folgenden Beispielen dargestellt. Im Anschluss daran folgen die formalen Definitionen für Bayessche Netze.

Das nachfolgende Beispiel wurde aus [Pearl 88] entnommen und behandelt einen einfachen medizinischen Sachverhalt (siehe dazu auch [Spiegelhalter 86], [Cooper 84] und [Neapolitan 90]).

***Beispiel 2.1.1 (Gehirntumor)***
Es ist ein einfaches Modell über die Zusammenhänge von Krebsmetastasen, Gehirntumor, Kalzium-Spiegel, Koma und Kopfschmerzen.

Krebsmetastasen sind die Ursache eines Gehirntumors und können auch den Kalzium-Spiegel erhöhen. Ein Gehirntumor oder ein erhöhter Kalzium-Spiegel können dazu führen, dass ein Patient ins Koma fällt. Ein Gehirntumor wiederum kann starke Kopfschmerzen verursachen. Diese Zusammenhänge sind übersichtlich als gerichteter Graph in Abbildung 2.1 dargestellt.

Jeder Knoten des Netzes symbolisiert eine *diskrete* Zufallsvariable, die als Wertemenge zwei Hypothesen besitzt, die zwei sich gegenseitig ausschließende

[1] Die Berechnungen auf dem Junction Tree werden auch als *Inferenz* oder *Propagierung* bezeichnet.

[2] Es ist zum Beispiel zwischen bedingten Wahrscheinlichkeiten, A-priori-Wahrscheinlichkeiten und Instantiierungen von Knoten zu unterscheiden.

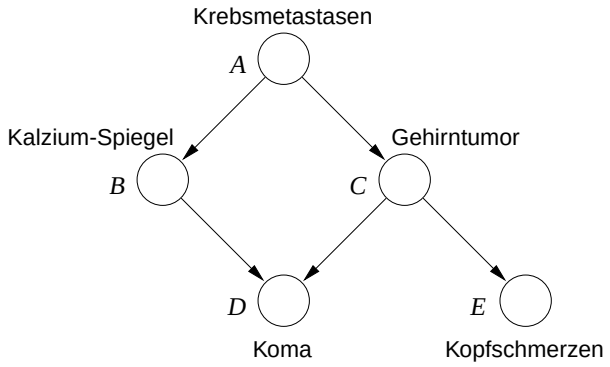



Abbildung 2.1: Beispielnetz Gehirntumor $[\Pr(a,b,c,d,e) = \Pr(a) \cdot \Pr(b \mid a) \cdot \Pr(c \mid a) \cdot \Pr(d \mid b,c) \cdot \Pr(e \mid c)]$

| | | |
|---|---|---|
| Krebsmetastasen (nein, ja) | | |
| $\Pr(a)$: | $\Pr(a_1) = 0.80$ | $\Pr(a_2) = 0.20$ |
| Kalzium-Spiegel (normal, erhöht) | | |
| $\Pr(b \mid a)$: | $\Pr(b_1 \mid a_1) = 0.80$ | $\Pr(b_2 \mid a_1) = 0.20$ |
| | $\Pr(b_1 \mid a_2) = 0.20$ | $\Pr(b_2 \mid a_2) = 0.80$ |
| Gehirntumor (nein, ja) | | |
| $\Pr(c \mid a)$: | $\Pr(c_1 \mid a_1) = 0.95$ | $\Pr(c_2 \mid a_1) = 0.05$ |
| | $\Pr(c_1 \mid a_2) = 0.80$ | $\Pr(c_2 \mid a_2) = 0.20$ |
| Koma (nicht im Koma, komatös) | | |
| $\Pr(d \mid b,c)$: | $\Pr(d_1 \mid b_1,c_1) = 0.95$ | $\Pr(d_2 \mid b_1,c_1) = 0.05$ |
| | $\Pr(d_1 \mid b_1,c_2) = 0.20$ | $\Pr(d_2 \mid b_1,c_2) = 0.80$ |
| | $\Pr(d_1 \mid b_2,c_1) = 0.20$ | $\Pr(d_2 \mid b_2,c_1) = 0.80$ |
| | $\Pr(d_1 \mid b_2,c_2) = 0.20$ | $\Pr(d_2 \mid b_2,c_2) = 0.80$ |
| Kopfschmerzen (keine, starke) | | |
| $\Pr(e \mid c)$: | $\Pr(e_1 \mid c_1) = 0.40$ | $\Pr(e_2 \mid c_1) = 0.60$ |
| | $\Pr(e_1 \mid c_2) = 0.20$ | $\Pr(e_2 \mid c_2) = 0.80$ |

Tabelle 2.1: Bedingte Wahrscheinlichkeiten bzw. A-priori-Wahrscheinlichkeiten für die Knoten des Beispielnetzes Gehirntumor in Abbildung 2.1.

und erzwingende Aussagen repräsentieren, d. h. genau dann ist die eine Aussage wahr, wenn die andere Aussage falsch ist.

Der Knoten A symbolisiert die Zufallsvariable Krebsmetastasen, die die beiden Aussagen ‘Patient hat keine Krebsmetastasen’ und ‘Patient hat Krebsmetastasen’ zusammenfasst. Der Wertebereich von Krebsmetastasen ist $\{\text{nein}, \text{ja}\}$. Die Hypothese ‘nein’ der Zufallsvariablen Krebsmetastasen steht für die Aussage

‘Patient hat keine Krebsmetastasen’, und die Hypothese ‘ja’ steht für die Aussage ‘Patient hat Krebsmetastasen’. Die zugehörigen Hypothesen der Zufallsvariablen Kalzium-Spiegel sind ‘normal’ und ‘erhöht’. Sie repräsentieren die beiden Aussagen ‘Patient hat normalen Kalzium-Spiegel’ und ’Patient hat erhöhten Kalzium-Spiegel’. Der Knoten C symbolisiert die Zufallsvariable Gehirntumor mit der Hypothese ‘nein’ für die Aussage ‘Patient hat keinen Gehirntumor’ und der Hypothese ‘ja’ für die Aussage ‘Patient hat einen Gehirntumor’. Dem Knoten D ist die Zufallsvariable Koma mit den Hypothesen ‘komatös’ und ‘nicht im Koma’ zugeordnet. Dabei steht die Hypothese ‘komatös’ für die Aussage ‘Patient befindet sich in tiefer Bewusstlosigkeit’ und die Hypothese ‘nicht im Koma’ für die Aussage ‘Patient ist bei Bewusstsein’. Die Hypothesen des Knotens E, der die Aussage Kopfschmerzen repräsentiert, sind ‘keine’ und ‘starke’.

Die Hypothesen $h$ der einzelnen Knoten sind mit $h_1$ bzw. $h_2$ für das Nichteintreffen bzw. für das Eintreffen beschriftet, wobei $h \in \{a, b, c, d, e\}$.

In Tabelle 2.1 sind die bedingten Wahrscheinlichkeiten der jeweiligen Knoten bzw. die A-priori-Wahrscheinlichkeiten der Wurzelknoten des Beispiels festgelegt. Aus dieser Tabelle geht zum Beispiel hervor, dass mit einer A-priori-Wahrscheinlichkeit von 80% keine Krebsmetastasen zu befürchten sind, oder auch, dass ein Patient ohne Krebsmetastasen mit einer Wahrscheinlichkeit von 20% einen erhöhten Kalzium-Spiegel hat. **Bsp**

Das nachfolgende Beispiel ist um einige Knoten größer als das vorherige Beispiel. Es wurde aus [Neapolitan 90] entnommen und behandelt wie das Beispielnetz zuvor einen vereinfacht dargestellten medizinischen Sachverhalt.

***Beispiel 2.1.2 (Asienbesuch)***
Es ist ein vereinfachtes Modell über die Zusammenhänge zwischen Atembeschwerden, Lungenkrebs, Tuberkulose, Bronchitis, Röntgenbefunden und kürzlichen Asienbesuchen.

Atembeschwerden können sowohl durch Lungenkrebs als auch durch Tuberkulose verursacht werden. Ein positiver Röntgenbefund ist zu gleichen Teilen ein Indiz für Lungenkrebs oder Tuberkulose. Ein kürzlicher Asienbesuch verstärkt dabei die Wahrscheinlichkeit von Tuberkulose. Auch Bronchitis ist eine Ursache für Atembeschwerden, wie Rauchen für Lungenkrebs oder Bronchitis.

Der gerichtete Graph in Abbildung 2.2 stellt diese Zusammenhänge noch einmal anschaulich dar. Jeder Knoten des Netzes ist binär, d. h. er hat zwei Hypothesen, die die dazugehörige Aussage falsch oder wahr werden lassen. Die dazugehörigen bedingten Wahrscheinlichkeiten der Knoten bzw. die A-priori-Wahrscheinlichkeiten der Wurzelknoten können dabei aus der Tabelle 2.2 abgelesen werden.

**Bsp**

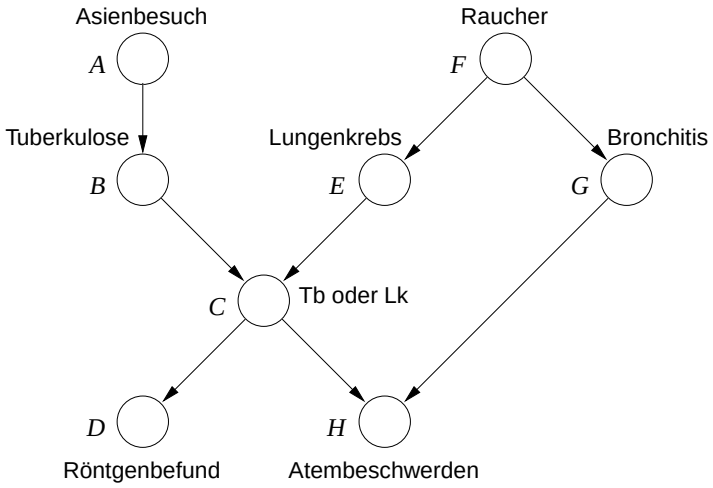



Abbildung 2.2: Beispielnetz Asienbesuch [$\Pr(a,b,c,d,e,f,g,h) = \Pr(a) \cdot \Pr(b \mid a) \cdot \Pr(c \mid b,e) \cdot \Pr(d \mid c) \cdot \Pr(e \mid f) \cdot \Pr(f) \cdot \Pr(g \mid f) \cdot \Pr(h \mid c,g)$]

## 2.2 Formale Definition der Bayesschen Netze

Nach der informellen Einführung der Bayesschen Netze durch zwei einfache Beispiele werden wir im folgenden die Begriffe und Schreibweisen formal definieren. Dazu betrachten wir zuerst die Struktur der Bayesschen Netze, die dann mit Zufallsvariablen und Wahrscheinlichkeiten versehen wird.

Für die Bezeichnung von Graphen und Bayesschen Netzen werden im folgenden verschiedene Begriffe definiert. Dabei haben wir uns für die Definition von Graphen an die Definition von *orientierten Graphen* aus [Hotz 90] angelehnt. Die Bezeichnungen und Schreibweisen wurden so angepasst, wie sie im Umgang mit Bayesschen Netzen üblich sind. Wie wir schon im vorhergehenden Abschnitt gesehen haben, gibt es einen qualitativen und einen quantitativen Teil zur Modellierung Bayesscher Netze. Der qualitative Teil des Bayesschen Netzes ist ein *gerichteter azyklischer Graph* und der quantitative Teil wird durch die den Knoten zugeordneten *Tabellen bedingter Wahrscheinlichkeiten* repräsentiert.

Formal gesprochen, ist ein Graph $\mathscr{G}$ ein Tupel $(\mathbb{V},\mathbb{E})$, wobei $\mathbb{V}$ eine endliche Menge von *Knoten* und $\mathbb{E}$ eine endliche Menge von *Kanten* ist. $\mathbb{E}$ ist eine Teilmenge von $\mathbb{V} \times \mathbb{V}$, und man schreibt: $\mathscr{G} = (\mathbb{V},\mathbb{E})$. Bei Kanten unterscheidet man *gerichtete* und *ungerichtete* Kanten. Bei ungerichteten Kanten spielt die Anordnung der Knoten keine Rolle, man spricht von einer Kante *zwischen* zwei Knoten. Eine Kante zwischen den Knoten $A$ und $B$ schreibt man mit $\{A,B\}$. Durch eine gerichtete Kante vom Knoten $A$ zum Knoten $B$ wird eine *Eltern-Kind-Relation* beschrieben; $A$ ist ein *Elternknoten* von $B$ und $B$ ist ein *Kinderknoten* von $A$. Man schreibt $(A,B)$. In einem gerichteten Graphen bezeichnet man Knoten

| | | |
|---|---|---|
| Asienbesuch ($A$ : $a_1 = \text{ja},\ a_2 = \text{nein}$) | | |
| $\Pr(a)$: | $\Pr(a_1) = 0.01$ | $\Pr(a_2) = 0.99$ |
| Tuberkulose ($B$ : $b_1 = \text{positiv},\ b_2 = \text{negativ}$) | | |
| $\Pr(b \mid a)$: | $\Pr(b_1 \mid a_1) = 0.05$ | $\Pr(b_2 \mid a_1) = 0.01$ |
| | $\Pr(b_1 \mid a_2) = 0.95$ | $\Pr(b_2 \mid a_2) = 0.99$ |
| Tb oder Lk ($C$ : $c_1 = \text{ja},\ c_2 = \text{nein}$) | | |
| $\Pr(c \mid b,e)$: | $\Pr(c_1 \mid b_1,e_1) = 1$ | $\Pr(c_2 \mid b_1,e_1) = 0$ |
| | $\Pr(c_1 \mid b_1,e_2) = 1$ | $\Pr(c_2 \mid b_1,e_2) = 0$ |
| | $\Pr(c_1 \mid b_2,e_1) = 1$ | $\Pr(c_2 \mid b_2,e_1) = 0$ |
| | $\Pr(c_1 \mid b_2,e_2) = 0$ | $\Pr(c_2 \mid b_2,e_2) = 1$ |
| Röntgenbefund ($D$ : $d_1 = \text{positiv},\ d_2 = \text{negativ}$) | | |
| $\Pr(d \mid c)$: | $\Pr(d_1 \mid c_1) = 0.98$ | $\Pr(d_2 \mid c_1) = 0.02$ |
| | $\Pr(d_1 \mid c_2) = 0.05$ | $\Pr(d_2 \mid c_2) = 0.95$ |
| Lungenkrebs ($E$ : $e_1 = \text{positiv},\ e_2 = \text{negativ}$) | | |
| $\Pr(e \mid f)$: | $\Pr(e_1 \mid f_1) = 0.10$ | $\Pr(e_2 \mid f_1) = 0.90$ |
| | $\Pr(e_1 \mid f_2) = 0.01$ | $\Pr(e_2 \mid f_2) = 0.99$ |
| Raucher ($F$ : $f_1 = \text{ja},\ f_2 = \text{nein}$) | | |
| $\Pr(f)$: | $\Pr(f_1) = 0.50$ | $\Pr(f_2) = 0.50$ |
| Bronchitis ($G$ : $g_1 = \text{positiv},\ g_2 = \text{negativ}$) | | |
| $\Pr(g \mid f)$: | $\Pr(g_1 \mid f_1) = 0.60$ | $\Pr(g_2 \mid f_1) = 0.40$ |
| | $\Pr(g_1 \mid f_2) = 0.30$ | $\Pr(g_2 \mid f_2) = 0.70$ |
| Atembeschwerden ($H$ : $h_1 = \text{ja},\ h_2 = \text{nein}$) | | |
| $\Pr(h \mid c,g)$: | $\Pr(h_1 \mid c_1,g_1) = 0.90$ | $\Pr(h_2 \mid c_1,g_1) = 0.10$ |
| | $\Pr(h_1 \mid c_1,g_2) = 0.70$ | $\Pr(h_2 \mid c_1,g_2) = 0.30$ |
| | $\Pr(h_1 \mid c_2,g_1) = 0.80$ | $\Pr(h_2 \mid c_2,g_1) = 0.20$ |
| | $\Pr(h_1 \mid c_2,g_2) = 0.10$ | $\Pr(h_2 \mid c_2,g_2) = 0.90$ |

Tabelle 2.2: Bedingte Wahrscheinlichkeiten bzw. A-priori-Wahrscheinlichkeiten für die Knoten des Beispielnetzes Asienbesuch in Abbildung 2.2.

ohne Elternknoten als *Wurzelknoten*. Die Menge aller Eltern- bzw. Kinderknoten des Knotens $V$ bezeichnet man mit $\mathbf{pa}(V)$ bzw. $\mathbf{ch}(V)$. Als *Familie* eines Knotens $V$ in einem gerichteten Graphen bezeichnet man $\mathbf{fa}(V) = \mathbf{pa}(V) \cup \{V\}$. Mit $\text{adj}(V)$ bezeichnet man die Menge der Knoten, die mit dem Knoten $V$ eine gemeinsame Kante haben, also miteinander benachbart (*adjazent*) sind. Eine Folge von Kanten $\phi$ eines gerichteten Graphen $\mathscr{G} = (\mathbb{V}, \mathbb{E})$ mit $\phi = (\phi_1, \ldots, \phi_t) \in \mathbb{E}^*$ und $\forall i \in \{1, \ldots, t-1\}$: $\text{Z}(\phi_i) = \text{Q}(\phi_{i+1})$ heißt *gerichteter Pfad* in $\mathscr{G}$. Ein gerich-

teter Pfad $\phi = (\phi_1, \ldots, \phi_t)$ mit $\mathrm{Q}(\phi_1) = \mathrm{Z}(\phi_t)$ heißt *Zyklus*. Enthält der gerichtete Graph $\mathscr{G}$ keine Zyklen, so heißt $\mathscr{G}$ *azyklisch*. Ein *gerichteter azyklischer Graph* (kurz: DAG (von *directed acyclic graph*)) ist ein Graph $\mathscr{G} = (\mathbb{V}, \mathbb{E})$, der nur gerichtete Kanten und keine Zyklen enthält.

Ein gerichteter azyklischer Graph kann mehrere gerichtete Pfade von $A$ nach $B$ enthalten. Man spricht dann von einem *mehrfach* ansonsten von einem *einfach* verknüpften Graphen.

Ein Bayessches Netz baut auf der Struktur eines gerichteten azyklischen Graphen auf. Jeder Knoten $V \in \mathbb{V}$ ist mit einer Zufallsvariablen $X_V \in \mathbb{X}$ mit einem endlichen Wertebereich $\mathbb{W}_{X_V}$ assoziiert. Die Struktur des gerichteten azyklischen Graphen gibt dabei die kausalen Abhängigkeiten der Zufallsvariablen untereinander an. Jedem Knoten $V \in \mathbb{V}$ mit der Zufallsvariablen $X_V$ ist eine Tabelle bedingter Wahrscheinlichkeiten (Conditional Probability Table (CPT)) $\mathrm{CPT}(V) = \theta_{(V|\mathbf{pa}(V))}$ zugeordnet, die den Einfluss der Zufallsvariablen $X_U$ der Knoten $U \in \mathbf{pa}(V)$ auf die Zufallsvariable $X_V$ des Knotens $V$ quantifizieren. Die Elemente der Menge $\mathbb{W}_{X_V}$ sind die Elementaraussagen der Zufallsvariablen $X_V$. Sie sind paarweise disjunkt und decken den gesamten Wertebereich der Zufallsvariablen ab. Jede Elementaraussage wird durch eine Hypothese $v_i$ des Knotens $V$ symbolisiert.

***Beispiel 2.2.1 (Hypothesen)***
Soll ein Knoten beispielsweise die Geschwindigkeit eines Autos repräsentieren, so kann er Hypothesen wie “langsam”, “mittel” und “schnell” oder “0–50 km/h”, “50–100 km/h” und “100–150 km/h” enthalten. In diesem Fall kann das Auto nicht schneller als 150 km/h fahren.

Der Knoten steht dann zum Beispiel für die Zufallsvariable Geschwindigkeit Auto mit den Elementaraussagen “Das Auto fährt mit langsamer Geschwindigkeit.”, “Das Auto fährt mit mittlerer Geschwindigkeit.” und “Das Auto fährt mit schneller Geschwindigkeit.”. **Bsp**

Eine Hypothese kann die beiden Werte 0 für “tritt nicht ein” und 1 für “kann noch eintreten” annehmen. Der Zustand der Hypothesen eines Knotens $V$ wird in einer Tabelle $\lambda_V$ festgehalten. Hat ein Knoten 3 Hypothesen, so sieht die Tabelle im Anfang so aus: $\lambda_V = \left[\begin{array}{ccc} v_1 & v_2 & v_3 \\ \hline 1 & 1 & 1 \end{array}\right]$, d. h., dass alle Hypothesen noch eintreten können. Soll markiert werden, dass die Elementaraussage wahr ist, die durch die Hypothese $v_2$ des Knotens $V$ symbolisiert wird, so sieht $\lambda_V$ wie folgt aus: $\lambda_V = \left[\begin{array}{ccc} v_1 & v_2 & v_3 \\ \hline 0 & 1 & 0 \end{array}\right]$. $\lambda_V$ wird auch als *Instantiierung* des Knotens $V$ bezeichnet. Der Belief-Wert eines Knotens $V$, $\mathrm{BEL}(V)$, gibt an, wie wahrscheinlich es ist, dass die Elementaraussage der zugehörigen Zufallsvariablen von $V$ bei gegebenen Instantiierungen der Knoten des Graphen noch zutrifft. Der BEL-Wert des Knotens $V$ $\mathrm{BEL}(V) = \left[\begin{array}{ccc} v_1 & v_2 & v_3 \\ \hline 0.50 & 0.30 & 0.20 \end{array}\right]$ zum Beispiel gibt an, dass die Elementaraussage, für das die Hypothese $v_2$ steht, in der gegebenen Situation noch mit einer Wahr-

scheinlichkeit von 30% zutrifft.

Die Struktur des Graphen $\mathscr{G}$ repräsentiert also die Abhängigkeiten der Zufallsvariablen. Eine Kante von $U$ zu $V$ bedeutet intuitiv, dass $X_U$ einen direkten Einfluss auf $X_V$ besitzt.

Die Wahrscheinlichkeitsverteilung aller Zufallsvariablen lässt sich als Produkt ihrer bedingten Wahrscheinlichkeiten schreiben: $\Pr(\mathbb{X}) = \Pr(X_1,\ldots,X_n) = \prod_{V\in\mathbb{X}} \Pr(V \mid \mathbf{pa}(V))$. Die bedingten Wahrscheinlichkeiten $\Pr(V \mid \mathbf{pa}(V))$ sind dabei dem Knoten $V$, der die Zufallsvariable $V$ repräsentiert, als Tabelle bedingter Wahrscheinlichkeiten $\mathrm{CPT}(V) = \theta_{(V|\mathbf{pa}(V))}$ zugeordnet (siehe Abschnitt 2.3.2).

Ist $V$ ein Wurzelknoten, so reduzieren sich die bedingten Wahrscheinlichkeiten $\Pr(V \mid \mathbf{pa}(V))$ zu $\Pr(V)$. Die den Ereignissen der Zufallsvariable eines Wurzelknotens zugeordneten Wahrscheinlichkeiten heißen *A-priori-Wahrscheinlichkeiten*.

***Beispiel 2.2.2 (Begriffe)***
Am Beispiel 2.1.1 werden nun einige Begriffe verdeutlicht. Der gerichtete azyklische Graph ist $\mathscr{G} = (\mathbb{V},\mathbb{E})$ mit den Knoten $\mathbb{V} = \{A,B,C,D,E\}$ und den Kanten $\mathbb{E} = \{(A,B),(A,C),(B,D),(C,D),(C,E)\}$. Die *Menge der diskreten Zufallsvariablen* ist $\mathbb{X} = \{$Krebsmetastasen, Kalzium-Spiegel, Gehirntumor, Koma, Kopfschmerzen$\}$. Die dem Knoten $B$ zugeordnete diskrete Zufallsvariable ist $X_B =$ Kalzium-Spiegel. Der Wertebereich von Kalzium-Spiegel ist $W_{X_B} = \{$normal, erhöht$\}$. Die bedingten Wahrscheinlichkeiten von Kalzium-Spiegel sind

$$\begin{aligned}
&\Pr(\text{Kalzium-Spiegel} = \text{normal} \mid \text{Krebsmetastasen} = \text{nein}) = 0.80,\\
&\Pr(\text{Kalzium-Spiegel} = \text{erhöht} \mid \text{Krebsmetastasen} = \text{nein}) = 0.20,\\
&\Pr(\text{Kalzium-Spiegel} = \text{normal} \mid \text{Krebsmetastasen} = \text{ja}) = 0.20 \text{ und}\\
&\Pr(\text{Kalzium-Spiegel} = \text{erhöht} \mid \text{Krebsmetastasen} = \text{ja}) = 0.80.
\end{aligned}$$

Die Tabelle der bedingten Wahrscheinlichkeiten von $B$ lautet

$$\mathrm{CPT}(B) = \theta_{(B|A)} = \left[\begin{array}{c|cc} & b_1 & b_2 \\ \hline a_1 & 0.80 & 0.20 \\ a_2 & 0.20 & 0.80 \end{array}\right].$$

**Bsp**

Zwischen Knoten und ihren Zufallsvariablen wird im weiteren nicht mehr unterschieden, außer dass ausdrücklich darauf hingewiesen wird.

Es gibt verschiedene Lösungsverfahren für mehrfach verknüpfte Bayessche Netze, die sich zum Beispiel in exakte und approximative Algorithmen einteilen lassen.

Zu den *exakten* Algorithmen gehören zum Beispiel das *Conditioning*- und das *Clustering*-Verfahren. Beim Conditioning wird die Netzstruktur des mehrfach verbundenen Bayesschen Netzes aufgebrochen, um dann auf einem einfach

verbundenen Bayesschen Netz den Standardalgorithmus für einfach verbundene Bayessche Netze anzuwenden. Das *Clustering*-Verfahren von Jensen wandelt ein mehrfach verbundenes Bayessches Netz in ein einfach verbundenes Bayessches Netz um, indem Knoten zusammengeführt werden. Auf diesem einfach verbundenen Bayesschen Netz kann dann wieder der Standardalgorithmus angewendet werden.

Als *approximatives* Verfahren sei zum Beispiel die *Stochastische Simulation* erwähnt, die auf dem unveränderten Bayesschen Netz arbeitet. Hierbei werden mehrfach in einer Reihenfolge alle Knoten des Bayesschen Netzes nach den Wahrscheinlichkeiten für das Eintreffen der Hypothesen des aktuellen Knotens in Abhängigkeit der sie umgebenen Knoten (genauer: die Knoten der sogenannten *Markov-Überdeckung* des betrachteten Knotens) zufällig instantiiert. Der Durchschnitt über alle Instantiierungsdurchläufe ergibt dann die gesuchte Lösung. Generell kann gesagt werden, dass die approximative Lösung um so näher an die genaue Lösung herankommt, je mehr Instantiierungsdurchläufe durchgeführt werden.

Zu den exakten Algorithmen gehören ebenso die *Junction-Tree*-Verfahren, die sich im Vergleich zu den anderen Verfahren besser für die Behandlung von dynamischen Bayesschen Netzen (siehe Abschnitt 3.3) eignen. In dieser Arbeit werden somit nur die Junction-Tree-Verfahren bearbeitet und die anderen Verfahren weitestgehend außer Betracht gelassen.

In den nachfolgenden Abschnitten werden nun verschiedene Junction-Tree-Verfahren vorgestellt, mit denen man in Bayesschen Netzen propagieren kann. Die verschiedenen Lösungsalgorithmen werden dabei anhand der beiden oben eingeführten Beispielnetze illustriert.

## 2.3 Junction-Tree-Verfahren

Der Junction Tree ist die einfach verbundene Struktur, die aus dem mehrfach verbundenen Bayesschen Netz bestimmt wird. Er wird benötigt, um die für die Bayesschen Netze notwendigen Berechnungen durchführen zu können.

Im Abschnitt 2.3.3 werden wir zum exakten Lösen eines Bayesschen Netzes die *Aalborg*-Architektur und die *Shafer-Shenoy*-Architektur vorstellen, die auf einem Junction Tree arbeiten.

Desweiteren werden wir in Kapitel 3.4 sehen, dass bei einigen Rollup-Verfahren der Junction Tree auf eine bestimmte Art konstruiert werden muss, damit ein Rollup durchgeführt werden kann.

Für die Bestimmung des Junction Trees werden die Knoten des Bayesschen Netzes zu *Megaknoten* zusammengefasst, und so miteinander durch Kanten verbunden, dass eine einfach verbundene Struktur entsteht. Diese Struktur und die

Megaknoten müssen bestimmte Eigenschaften erfüllen, so dass man von einem Junction Tree sprechen kann. Im folgenden Abschnitt wird gezeigt, wie man einen Junction Tree für ein Bayessches Netz bestimmt.

### 2.3.1 Bestimmung des Junction Trees

Für ein Bayessches Netz existieren mehrere Junction Trees als Lösungen, die unterschiedlich gut sind: Vom Junction Tree ist es abhängig, wie schnell ein Bayessches Netz gelöst werden kann und wie hoch die Speicherkomplexität liegt.

Im folgenden wird nun die allgemeine Vorgehensweise zur Bestimmung eines Junction Trees für ein Bayessches Netz den Einzelschritten nach strukturiert dargestellt. Zu den verschiedenen Einzelschritten werden gelegentlich einzelne Effizienzaspekte für einen Junction Tree angesprochen. Zum besseren Verständnis werden die Einzelschritte zur Bestimmung des Junction Trees anhand der beiden Beispielnetze Gehirntumor und Asienbesuch aus Abschnitt 2.1 grafisch dargestellt.

Dieser Abschnitt gibt einen Überblick über die Bestimmung eines *Junction Trees* für ein Bayessches Netz $\mathscr{BN} = (\mathbb{V}, \mathbb{E}, \mathbb{X}, \Pr)$ über den Zufallsvariablen $\mathbb{X}$. Bei der Umwandlung des Bayesschen Netzes in seine zweite Struktur, dem *Junction Tree*, werden Einzelschritte durchgeführt, die wie folgt zusammengefasst werden können:

1. **Moralisierung:** Aus dem gerichteten azyklischen Graphen $\mathscr{G}$ wird der *moralisierte Graph* $\mathscr{G}^m$ konstruiert, indem für alle Knoten des Graphen die jeweiligen Elternknoten durch Kanten miteinander verbunden werden.

2. **Eliminationsreihenfolge und Triangulierung:** Es wird eine *Eliminationsreihenfolge elim* bestimmt und damit aus dem moralisierten Graphen $\mathscr{G}^m$ der *triangulierte Graph* $\mathscr{G}^t$ konstruiert.

3. **Junction Graph:** Aus dem *triangulierten Graphen* $\mathscr{G}^t$ werden die *Cliquen* herausgelesen und der mehrfach verknüpfte *Junction Graph* $\mathscr{J}$ bestimmt, wobei die Cliquen die Knoten des Junction Graphen $\mathscr{J}$ bilden.

4. **Junction Tree:** Aus dem mehrfachverknüpften Junction Graphen $\mathscr{J}$ wird durch Entfernung der überflüssigen Kanten ein einfach verknüpfter *Junction Tree* $\mathscr{T}$ bestimmt.

5. **Knotenzuordnung:** Jeder Knoten des Bayesschen Netzes $\mathscr{BN}$ wird einer Clique des Junction Trees $\mathscr{T}$ zugeordnet.

6. **Wurzelknoten:** Wähle einen beliebigen Knoten des *Junction Trees* $\mathscr{T}$ als Wurzelknoten.

Die Schritte von 1 bis 4 sind dabei von graphentheoretischer Art, d. h. es werden nur Umformungen des Graphen des Bayesschen Netzes vorgenommen. Wir verweisen auf [Pearl 88] , [Neapolitan 90], [Jensen 96], [Huang & Darwiche 96] und [Brandherm et al. 97] für Details und Hintergründe. Im folgenden geben wir für jeden Einzelschritt einen Überblick.

#### 2.3.1.1 Moralisierung

Bei der Moralisierung werden die Elternknoten eines Knotens durch Kanten miteinander verbunden. Man sagt auch, dass die Elternknoten miteinander verheiratet werden, was den Begriff *Moralisierung* erklärt.

Die Moralisierung eines Graphen $\mathscr{G}$ stellt sicher, dass ein Knoten zusammen mit seinen Elternknoten in einer Clique des Junction Trees zu liegen kommt. Dies spielt eine wichtige Rolle bei der Zuordnung von Knoten zu Cliquen, die weiter hinter in diesem Abschnitt beschrieben wird.

Sei nun $\mathscr{G} = (\mathbb{V}, \mathbb{E})$ der gerichtete azyklische Graph des Bayesschen Netzes $\mathscr{BN} = (\mathbb{V}, \mathbb{E}, \mathbb{X}, \mathrm{Pr})$. Den *moralisierten Graphen* $\mathscr{G}^m$ erhält man aus dem Graphen $\mathscr{G}$ wie folgt:

1. Es wird der Graph $\mathscr{G}$ in $\mathscr{G}^m = (\mathbb{V}^m, \mathbb{E}^m)$ “kopiert”.
2. Für alle Knoten $V \in \mathbb{V}^m$ werden die Elternknoten $\mathbf{pa}(V)$ durch ungerichtete Kanten miteinander verbunden.
3. Es werden sämtliche gerichteten Kanten $(A, B) \in \mathbb{E}^m$ durch ungerichtete Kanten $\{A, B\}$ ersetzt.

Der aus diesen Umformungen resultierende Graph $\mathscr{G}^m = (\mathbb{V}^m, \mathbb{E}^m)$ ist ein ungerichteter Graph. Es gilt: $\mathbb{V} = \mathbb{V}^m$ und $\mathbb{E} \subset \mathbb{E}^m$. In den Abbildungen 2.3(a) und 2.4(a) sind die moralisierten Graphen der Beispielnetze aus den Beispielen 2.1.1 und 2.1.2 dargestellt. Die neu eingefügten Kanten sind gestrichelt dargestellt.

#### 2.3.1.2 Eliminationsreihenfolge

Die Eliminationsreihenfolge ist eine Liste der Knoten des Bayesschen Netzes, die die Struktur des Junction Graphen –einer Vorstufe des Junction Trees– für dieses Bayessche Netz eindeutig bestimmt. Ändert man die Reihenfolge der Knoten in der Liste, so entsteht im allgemeinen ein anderer Junction Tree. Für die sogenannte Triangulierung, die im folgenden Abschnitt beschrieben wird, ist die Eliminationsreihenfolge der Fahrplan für die Bestimmung der Triangulationskanten. Wie man sieht, sind die Eliminationsreihenfolge und die Triangulierung stark miteinander verbunden. Ist die Eliminationsreihenfolge bekannt, so kann das Bayessche Netz trianguliert werden.

In diesem Abschnitt werden nur die notwendigsten Begriffe eingeführt und auf Beweise verzichtet, da sie für diese Arbeit im weiteren nicht notwendig sind.

Um ein Bayessches Netz in eine einfach verbundene Struktur umzuwandeln, werden Knoten zu "Megaknoten" zusammengefasst. Dazu werden die Knoten des Bayesschen Netzes in einer bestimmten Reihenfolge *eliminiert*. Bei der Elimination von Knoten werden Kanten im Graphen des Bayesschen Netzes eingefügt, so dass das Bayessche Netz zusammenhängend bleibt, wenn man den Knoten aus dem Netz entfernt. Dies ist wichtig für die Junction-Tree-Eigenschaft (siehe Abschnitt 2.3.1.5), die vom *Junction Tree* erfüllt werden muss. Die Reihenfolge, in der die Knoten des Bayesschen Netzes eliminiert werden, wird als *Eliminationsreihenfolge* bezeichnet. Die Eliminationsreihenfolge gibt zusammen mit den Kanten (auch den neu eingefügten) des Netzes an, welche Knoten zusammen in einem "Megaknoten" zu liegen kommen. Es entsteht zunächst ein mehrfach verbundener Graph, der Junction Graph, der dann durch einfaches Löschen bestimmter Kanten in einen einfach verbundenen Graphen, den Junction Tree, umgewandelt wird. Dazu mehr in den zugehörigen Abschnitten. Die "Megaknoten" nennt man *Cliquen*. Bei unterschiedlichen Eliminationsreihenfolgen entstehen unterschiedlich viele und unterschiedlich große Cliquen. Ziel ist es, möglichst kleine Cliquen zu erhalten. Eine Eliminationsreihenfolge, die dies leistet, wird als *optimal* bezeichnet.

Die Bestimmung einer optimalen Eliminationsreihenfolge ist jedoch *NP-vollständig* (siehe [Cooper 87]). Deswegen empfehlen sich zur Berechnung einer Eliminationsreihenfolge Heuristiken, wie sie z. B. in [Kjærulff 90, Kjærulff 93] vorgestellt werden.

Das traditionelle Verfahren zur Bestimmung einer Eliminationsreihenfolge ist maximum cardinality search. Dabei wird zufällig ein beliebiger Knoten des Graphen als Startknoten ausgewählt und markiert. Der Knoten und alle seine unmarkierten Nachbarknoten bilden dabei einen *Cluster*. Alle Knoten des Clusters werden miteinander mit ungerichteten Kanten verbunden. Als nächstes wird der Knoten bearbeitet, der jetzt die meisten markierten Nachbarknoten besitzt. Gibt es mehrere solcher Knoten, dann wird ein Knoten zufällig ausgewählt. Das Verfahren terminiert, wenn alle Knoten des Graphen markiert sind. Die Knoten ergeben nun sortiert nach der Reihenfolge ihrer Abarbeitung die Eliminationsreihenfolge.

Wird ein Knoten $V \in \mathbb{V}$ mitsamt seinen Kanten aus $\mathscr{G} = (\mathbb{V}, \mathbb{E})$ entfernt, so sagt man, dass der Knoten $V$ *gelöscht* wird. Werden vor dem Löschen von $V$ alle Nachbarknoten von $V \in \mathbb{V}$ ($\mathrm{adj}(V)$) durch Kanten miteinander verbunden, so sagt man, dass der Knoten $V$ *eliminiert* wird. Bei der Elimination eines Knotens bleibt der Teilgraph, der vom Knoten induziert wird, durch das Einfügen der Kanten zwischen den restlichen Knoten des Teilgraphen als Zusammenhangskomponente erhalten. Dieses kommt bei der Junction-Tree-Eigenschaft zum Tragen.

Bei den Clustering-Verfahren in [Pearl 88, Neapolitan 90] ist die Eliminati-

onsreihenfolge durch maximum cardinality search vorbestimmt. Die Clustering-Verfahren von [Jensen 96, Shafer 96] akzeptieren beliebige Eliminationsreihenfolgen.

Andere Algorithmen zur Bestimmung der Eliminationsreihenfolge als maximum cardinality search berücksichtigen das *Gewicht* der Clique, die bei der Elimination eines Knotens entstehen würde, und nehmen die Clique mit dem kleinsten Gewicht. Die Heuristik dabei ist, dass insgesamt nur Cliquen mit kleinen Gewichten entstehen. Als Gewicht einer Clique bezeichnet man die Anzahl der verschiedenen Zustände, die die Clique haben kann. Enthält die Clique die Knoten $K_1, \ldots, K_n$, dann bestimmt sich die Anzahl der Zustände der Clique zu $|K_1| \cdot \ldots \cdot |K_n|$. Für $1 \leq i \leq n$ gibt $|K_i|$ die Anzahl der Hypothesen des Knotens $K_i$ an.

Durch die Eliminationsreihenfolge *elim* wird auf dem Graphen eine Ordnung der Knoten vorgenommen. Formal gesehen ist eine Ordnung auf $\mathscr{G} = (\mathbb{V}, \mathbb{E})$ eine Bijektion $\# : \mathbb{V} \leftrightarrow \{1, 2, \ldots, n\}$ mit $|\mathbb{V}| = n$. Den Graphen $\mathscr{G}_\# = (\mathbb{V}_\#, \mathbb{E})$ bezeichnet man auch als *geordneten Graphen*. Es gilt: $\mathscr{G} = \mathscr{G}_\#$ und $\mathbb{V} = \mathbb{V}_\#$. Der Index # gibt dabei an, dass dem Graphen bzw. der Knotenmenge eine Ordnung zugewiesen wurde. Die Ordnung des Graphen $\mathscr{G}_\#$ steht dabei mit der Eliminationsreihenfolge *elim* im folgenden Zusammenhang: $elim = (\#^{-1}(1), \ldots, \#^{-1}(n)) = (V_{\#(1)}, \ldots, V_{\#(n)})$.

Die Eliminationsreihenfolge in Abbildung 2.3(b) wurde durch maximum cardinality search bestimmt: Der Knoten $A$ wurde durch Zufall als erster Knoten ausgewählt. Als nächste Kandidaten stehen die beiden Knoten $B$ und $C$ zur Auswahl. Das Los fällt auf den Knoten $B$. Der Knoten $C$ hat nun die meisten markierten Nachbarknoten und wird als nächster markiert. Dann kommt der Knoten $D$ mit den meisten markierten Nachbarknoten an die Reihe und zum Schluss der Knoten $E$.

Bei der Bestimmung der Eliminationsreihenfolge in Abbildung 2.4(b) wurde die folgende Heuristik angewandt: *Eliminiere die Knoten des Graphen, die keine Kanten im Graphen induzieren. Gibt es keine solche Knoten, so wähle den Knoten, der die kleinste Clique entstehen lässt.* Bei der Eliminierung der Knoten $H$, $D$, $A$ und $B$ werden in dieser Reihenfolge keine Kanten im Graphen eingefügt. Jetzt steht kein Knoten mehr zur Auswahl, der keine Kanten bei seiner Eliminierung im Graphen einfügen würde. Jeder der verbleibenden Knoten fügt bei seiner Eliminierung genau eine Kante ein. Die entstehenden Cliquen wären somit alle gleich groß. Die zufällige Wahl fällt auf den Knoten $G$. Er induziert die Kante zwischen den Knoten $C$ und $F$. Die Eliminierung der Knoten $F$, $E$ und $C$ in dieser Reihenfolge fügt keine weiteren Kanten mehr im Graphen ein.

Bei der vorhergehenden Bestimmung der Eliminationsreihenfolge werden bereits Kanten in den Graphen eingefügt, und der resultierende Graph ist am Ende *trianguliert*. Im Gegensatz dazu wurde bei der Bestimmung der Eliminationsreihenfolge durch maximum cardinality search der Graph noch nicht trianguliert.

Wie man bei einer gegebenen Eliminationsreihenfolge die Triangulierung eines Graphen berechnet wird nun im folgenden Abschnitt beschrieben.

#### 2.3.1.3 Triangulierung

Eine Triangulierung des Bayesschen Netzes kann zusammen mit der Bestimmung der Eliminationsreihenfolge durchgeführt werden. Da in manchen Fällen aber die Eliminationsreihenfolge durch den Benutzer vorgegeben wird, wollen wir die Triangulierung getrennt von der Eliminationsreihenfolge vorstellen.

In diesem Abschnitt wird beschrieben, wann ein Graph trianguliert ist, wie man bei gegebener Eliminationsreihenfolge die Triangulierung eines Graphen bestimmt, und wie man bei bereits triangulierten Graphen ihre Triangulierung verbessern kann, denn eine gute Triangulierung ist die Voraussetzung für einen schlanken Junction Tree, der für schnelle Berechnungen notwendig ist. Anhand der beiden Beispielnetze Gehirntumor und Asienbesuch zeigen wir, woran man eine gute von einer schlechten Triangulierung unterscheiden kann.

Die *Triangulierung* wird auf dem moralisierten Graphen $\mathscr{G}^m$ ausgeführt. $\mathscr{G}^m$ ist dann trianguliert, wenn jeder Zyklus der Länge vier oder mehr eine Kante besitzt, die zwei nicht benachbarte Knoten im Zyklus im Graphen miteinander verbindet. Diese Kante wird auch als "*chord*" bezeichnet.

Sei $\mathscr{G}^m = (\mathbb{V}^m, \mathbb{E}^m)$ der moralisierte Graph des Bayesschen Netzes. Den *triangulierten Graphen* $\mathscr{G}^t$ erhält man mithilfe der Eliminationsreihenfolge *elim* wie folgt:

1. Es wird der Graph $\mathscr{G}^m$ in $\mathscr{G}^t = (\mathbb{V}^t, \mathbb{E}^t)$ kopiert. Alle Knoten $V \in \mathbb{V}^t$ sind noch nicht markiert.
2. Für alle Knoten $V \in \mathbb{V}^t$ wird der Reihe nach, wie sie in der Eliminationsreihenfolge *elim* stehen, folgendes durchgeführt:
   (a) Die Nachbarknoten von $V$, die noch nicht markiert sind, werden durch eine ungerichtete Kante miteinander verbunden. (Dadurch kann sich die Menge der Nachbarknoten eines noch nicht markierten Knotens $V$ vergrößern.)
   (b) Der Knoten $V$ wird markiert.
3. $\mathscr{G}^t$ ist nun der triangulierte Graph von $\mathscr{G}$.

Im Gegensatz zu dem oben vorgestellten Verfahren zur Triangulierung eines moralisierten Graphen wird beispielsweise in [Pearl 88] die Eliminationsreihenfolge *rückwärts* abgearbeitet. Je nachdem, ob die Eliminationsreihenfolge *vorwärts* oder *rückwärts* abgearbeitet wird, entstehen im allgemeinen zwei unterschiedliche Triangulierungen, die sich auch stark in ihrer Qualität unterscheiden

können. Deswegen ist es wichtig zu wissen, ob vorwärts oder rückwärts eliminiert wird. Wegen der dynamischen Bayesschen Netze, die in Kapitel 3 besprochen werden, haben wir uns für die Vorwärtselimination entschieden, so dass man die Knoten einer Zeitscheibe, die an das existierende dynamische Bayessche Netz angehängt wird, einfach an die existierende Eliminationsreihenfolge anhängen kann. An dieser Stelle würden die Ausführungen für dieses Vorgehen jedoch zu weit gehen, weshalb wir auf den Abschnitt 3.3.2.2 verweisen wollen, in dem es genau beschrieben wird.

Die Menge der Kanten, die durch die Elimination der Knoten $V \in \mathbb{V}$ von $\mathscr{G}$ in der Reihenfolge *elim* dem Graphen $\mathscr{G}$ hinzugefügt werden, bezeichnet man als Triangulierung $\mathbb{T}(\mathscr{G}_{\#})$ oder auch als Triangulierung von $\mathscr{G}^m$, da $(\mathbb{V}, \mathbb{E}^m \cup \mathbb{T})$ *trianguliert* ist. Die Kanten $\mathbb{T}$ bezeichnet man als *Triangulationskanten*. Die Triangulationskanten sind in den Abbildungen 2.3(b), 2.4(b), 2.3(c) und 2.4(c) gestrichelt eingezeichnet. Eine Eliminationsreihenfolge *elim* mit $\mathbb{T}(\mathscr{G}_{\#}) = \emptyset$ nennt man *perfekt*. Eine perfekte Eliminationsreihenfolge existiert nicht für jeden Graphen $\mathscr{G}$, wie man am Graphen in Abbildung 2.4(b) sehen kann, da es für ihn keine Eliminationsreihenfolge gibt, die keine Triangulationskante einfügt.

In den Abbildungen 2.3(b), 2.4(b), 2.3(c) und 2.4(c) wird deutlich, wie die Qualität der Triangulierung von der Eliminationsreihenfolge abhängt. In Abbildung 2.3(b) wurde keine Kante und in Abbildung 2.4(b) nur eine Kante eingefügt, für die sich der Knoten $G$ verantwortlich zeigt. In Abbildung 2.3(c) wurden fünf Kanten eingefügt, so dass alle Knoten des Graphen $\mathscr{G}$ miteinander verbunden sind. Eine schlechtere Triangulierung existiert für diesen Graphen nicht. In Abbildung 2.4(c) kamen sieben Kanten hinzu. Für diesen Graphen gibt es allerdings noch schlechtere Triangulierungen.

Der Vergleich von Abbildung 2.3(b) mit Abbildung 2.3(c) zeigt, dass die neu eingefügten Kanten in Abbildung 2.3(c) überflüssig für eine Triangulierung sind. Man bezeichnet diese Kanten als *redundante* Triangulationskanten. Werden die redundanten Triangulationskanten nacheinander entfernt, so dass keine redundanten Triangulationskanten mehr vorhanden sind, so erhält man eine *minimale* Triangulierung, d. h. wenn eine weitere Triangulationskante der Triangulierung $\mathbb{T}$ entfernt würde, dann wäre $\mathbb{T}$ keine Triangulierung mehr. Dazu wird in [Kjærulff 93] ein Algorithmus vorgeschlagen, der redundante Triangulationskanten aus einer nichtminimalen Triangulation $\mathbb{T}$ eines Graphen $\mathscr{G}$ entfernt (siehe auch Abschnitt B.4). Die Komplexität dieses Algorithmus liegt bei $O(c^2\,|\,\mathbb{T}\,|^2)$. Eine minimale Triangulierung ist aber nicht unbedingt die *absolut minimale* Triangulierung eines Graphen. Es gilt zwar, dass $\forall\ \mathbb{T}' \subset \mathbb{T}$ ist $\mathscr{G}' = (\mathbb{V}, \mathbb{E}^m \cup \mathbb{T}')$ nicht trianguliert, es kann aber eine Triangulierung $\mathbb{T}_<$ geben mit $\mathbb{T}_< \not\subset \mathbb{T}$ und $|\mathbb{T}_<| < |\mathbb{T}|$.

Wie sich die Triangulierung des Graphen auf den Junction Tree auswirkt und deshalb von *guten* und *schlechten* Triangulierungen die Rede ist, wird in Ab-

schnitt 2.3.1.5 beleuchtet. Für effizientere Methoden zur Graphentriangulierung verweisen wir auf [Becker & Geiger 96].

Im triangulierten Graphen können nun mit der Eliminationsreihenfolge die Cliquen des Junction Graphen $\mathscr{J}$ bestimmt werden.

#### 2.3.1.4 Junction Graph

Der Junction Graph $\mathscr{J}$ ist die Vorstufe vom Junction Tree $\mathscr{T}$. Die Knoten des Junction Graphen sind die Cliquen des triangulierten Graphen $\mathscr{G}^t_\#$. Die Knoten des Junction Graphen bezeichnet man auch als Cliquen. Die Cliquen des Junction Graphen, die gemeinsame Knoten enthalten, sind dabei durch Kanten miteinander verbunden.

In einem triangulierten Graphen mit der Ordnung #, bildet der Knoten $V \in \mathscr{G}^t_\#$ zusammen mit seinen Nachbarknoten $N \in \text{adj}(V)$, die eine höhere Ordnung als er haben (d. h. $\#(N) > \#(V)$), einen *Cluster*. Ein Cluster $Clu_i$ heißt *maximal*, wenn sich kein weiterer Cluster $Clu_\ell$ des Graphen $\mathscr{G}^t_\#$ findet, so dass der Cluster $Clu_i$ eine Teilmenge vom Cluster $Clu_\ell$ ist, d. h. $\nexists \ell : Clu_i \subset Clu_\ell$. Die *maximalen Cluster* des Graphen $\mathscr{G}^t_\#$ werden *Cliquen* genannt und bilden die Knoten des Junction Graphen $\mathscr{J}$.

Die Cliquen des Junction Graphen $\mathscr{J}$ werden durch eine Kante verbunden, wenn sie wenigstens einen Knoten gemeinsam haben. Auf diese Weise erhält man einen mehrfach verbundenen Graphen $\mathscr{J}$. Den Kanten des Junction Graphen $\mathscr{J}$ ist dabei in Abhängigkeit von den beiden durch die Kante verbundenen Cliquen ein *Gewicht* zugeordnet. Das Gewicht einer Kante zwischen $Clq_i$ und $Clq_\ell$ bestimmt sich zu $\gamma := |Clq_i \cap Clq_\ell|$. $Clq_i \cap Clq_\ell$ wird als *Separator* der beiden Cliquen $Clq_i$ und $Clq_\ell$ bezeichnet.

In Tabelle 2.3 ist beispielhaft ein Triangulationsprozess mit den Triangulationskanten, den induzierten Clustern und Cliquen und den Kanten zwischen den Cliquen für das Beispielnetz Asienbesuch dargestellt (vgl. dazu auch die Abbildungen 2.4(b) und 2.4(d)).

Durch das Entfernen bestimmter Kanten im Junction Graphen $\mathscr{J} = (\mathbb{C}, \mathbb{E}_\gamma)$ erhält man einen Junction Tree $\mathscr{T}$, wobei mit $\mathbb{C}$ die Cliquen des Graphen und mit $\mathbb{E}_\gamma$ die mit einem Gewicht versehenen Kanten des Graphen bezeichnet werden.

#### 2.3.1.5 Junction Tree

Der Junction Graph $\mathscr{J}$ ist eine mehrfach verbundene Struktur. Im folgenden wird gezeigt, wie aus dieser mehrfach verbundenen Struktur eine einfach verbundene Struktur, der sogenannte *Junction Tree* $\mathscr{T}$, bestimmt wird, der die sogenannte *Junction-Tree-Eigenschaft* erfüllt.

Der Junction Tree $\mathscr{T} = (\mathbb{C}, \mathbb{E}^{\mathscr{T}}_{Sep})$ des Junction Graphen $\mathscr{J}$ ist ein minimal aufspannender Graph von $\mathscr{J}$ (siehe auch [Jensen & Jensen 94]), der die *Junction-Tree-Eigenschaft* erfüllt. Dazu muss für alle Knoten $V$ des Graphen $\mathscr{G}$ gelten: wenn alle Cliquen aus $\mathscr{J}$ entfernt werden, die den Knoten $V$ nicht enthalten, dann ist der entstehende Unterbaum zusammenhängend, d. h. zwei Cliquen in $\mathscr{M}$, die den Knoten $V$ enthalten, sind in $\mathscr{M}$ benachbart, oder über einen Pfad verbunden, dessen Cliquen den Knoten $V$ enthalten.

Jeder Kante $(Clq_i, Clq_j)$ des *Junction Trees* $\mathscr{T}$ wird ein Separatorknoten $Sep_{i,j}$ zugeordnet, wobei der Separatorknoten die gemeinsamen Knoten der beiden Cliquen enthält, d. h. $Sep_{i,j} = Clq_i \cap Clq_j$. Es gilt: $\mathbb{E}^{\mathscr{T}}_{Sep} \subseteq \mathbb{E}_\gamma$, wobei $\mathbb{E}_\gamma$ die gewichteten Kanten des Junction Graphen und $\mathbb{E}^{\mathscr{T}}_{Sep}$ die mit einem Separator versehenen Kanten des Junction Trees sind.

Wie man aus einem gegebenen Junction Graphen einen *optimalen* Junction Tree bestimmt, so dass später die benötigte Zeit für Inferenzen minimiert wird, wird z. B. in [Jensen & Jensen 94, Huang & Darwiche 96] beschrieben. Beispielsweise bestimmt der Algorithmus MST-Kruskal($\mathscr{J}, \gamma$) einen Junction Tree aus einem gegebenen Junction Graphen, indem er die Kanten, nach ihrem Gewicht sortiert und mit dem kleinsten Gewicht beginnend, durchgeht und all diejenigen Kanten löscht, bei denen der Graph zusammenhängend bleibt, wenn sie gelöscht werden (siehe Abschnitt B.4 oder [Cormen et al. 92]).

Im allgemeinen ist der Junction Tree eines Junction Graphen $\mathscr{J}$ nicht eindeutig bestimmt. In Abbildung 2.4(f) hätte anstatt der Kante zwischen Clique $Clq_2$ und Clique $Clq_4$ auch die Kante zwischen Clique $Clq_2$ und Clique $Clq_6$ gelöscht werden können.

In den Abbildungen 2.3 und 2.4 wird für das Beispielnetz Gehirntumor bzw.

| *triangulierter Graph* | | | *Junction Graph* | |
|---|---|---|---|---|
| *eliminierter Knoten* | *Triang.-kante* | *induzierter Cluster* | *induzierte Clique* | *Kante zur Clique* |
| $H$ | - | $CGH$ | $Clq_1 = \{C, G, H\}$ | - |
| $D$ | - | $CD$ | $Clq_2 = \{C, D\}$ | $Clq_1$ |
| $A$ | - | $AB$ | $Clq_3 = \{A, B\}$ | - |
| $B$ | - | $BCE$ | $Clq_4 = \{B, C, E\}$ | $Clq_1$,$Clq_2$,$Clq_3$ |
| $G$ | $\{C, F\}$ | $CFG$ | $Clq_5 = \{C, F, G\}$ | $Clq_1$,$Clq_2$,$Clq_4$ |
| $F$ | - | $CEF$ | $Clq_6 = \{C, E, F\}$ | $Clq_1$,$Clq_2$,$Clq_4$,$Clq_5$ |
| $E$ | - | $CE$ | - | - |
| $C$ | - | $C$ | - | - |

Tabelle 2.3: Triangulierungsprozess mit Cliquen des Beispielnetzes Asienbesuch

für das Beispielnetz Asienbesuch jeweils die Bestimmung eines Junction Trees anhand einer guten und einer schlechten Eliminationsreihenfolge vorgeführt.

#### 2.3.1.6 Knotenzuordnung

Im folgenden werden die Zufallsvariablen des Bayesschen Netzes den Cliquen des Junction Trees zugeordnet. Dazu werden die bedingten Wahrscheinlichkeiten, die mit einer Zufallsvariablen assoziiert sind, zu einer Tabelle der Clique hinzumultipliziert.

Jeder Knoten des Bayesschen Netzes wird eineindeutig einer Clique *Clq* seines Junction Trees $\mathscr{T}$ *zugeordnet*. Ein Knoten kann in *mehreren Cliquen liegen* aber *nur einer Clique zugeordnet* sein, d. h. $\exists f\colon \mathbb{V} \to \mathbb{C}$ mit $K \mapsto f(K) = Clq_i \in \mathbb{C}$. In Worten: Es gibt eine Funktion $f$, die für jeden Knoten $K \in \mathbb{V}$ des Graphen $\mathscr{G}$ angibt, welcher Clique des zugehörigen *Junction Trees* er zugeordnet ist.

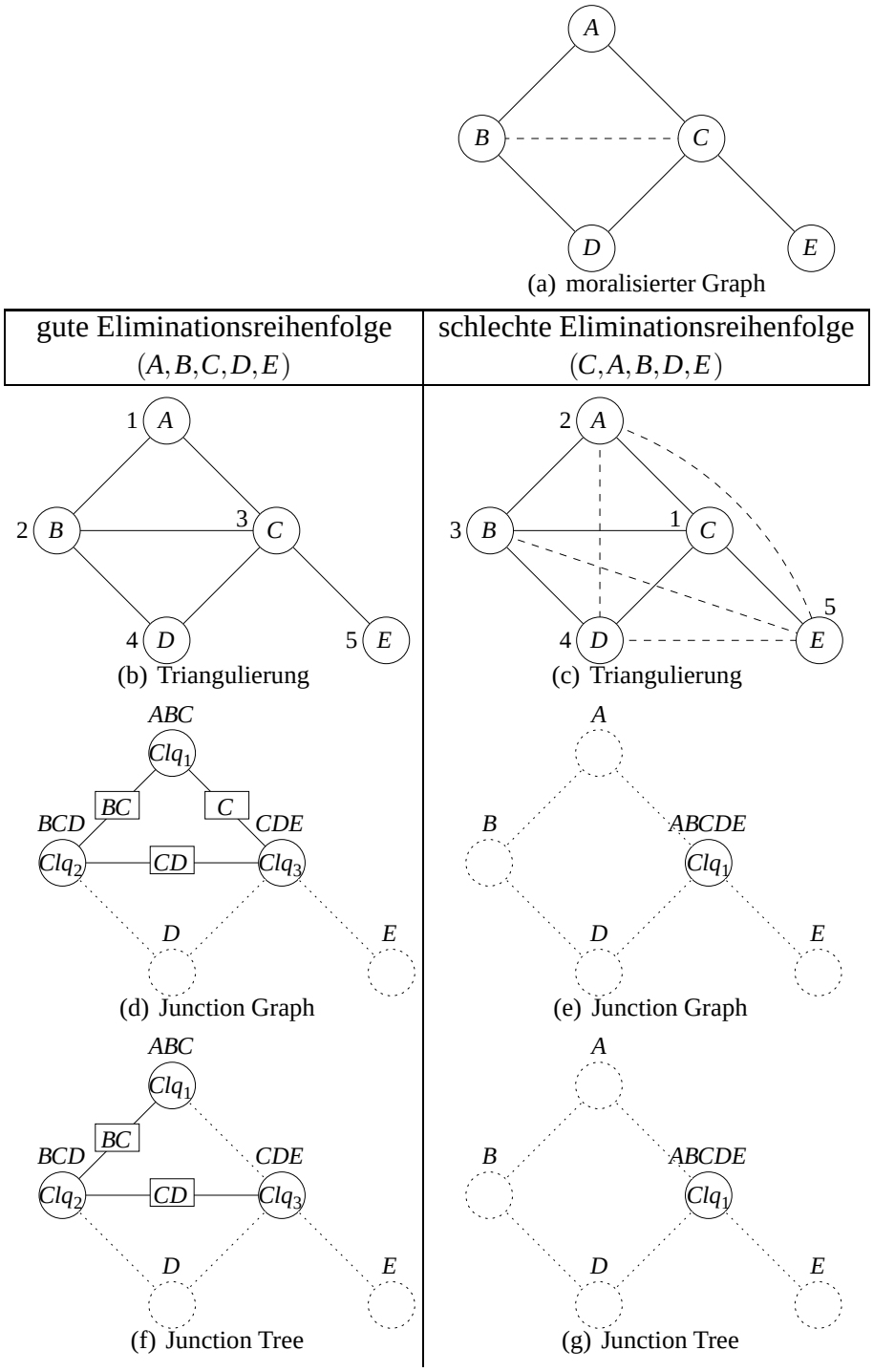
In der Clique, die einem Knoten zugeordnet wird, müssen auch die Eltern des Knotens liegen, d. h. $\{K\} \cup \mathbf{pa}(K) \subseteq f(K) = Clq_i$. (Die Elternknoten müssen der Clique aber nicht zugeordnet sein.) Durch die Moralisierung des Graphen des Bayesschen Netzes im Anfang der Konstruktion des Junction Trees wird sichergestellt, dass ein Knoten mit seinen Elternknoten in wenigstens einer Clique des Junction Trees zusammen zu liegen kommt.

Den einzelnen Cliquen $Clq_i$ des Junction Trees sind Tabellen $\theta_{Clq_i}$ zugeordnet, die im Anfang mit Einsen vorbelegt sind. Die Dimension einer Tabelle ist jeweils so bemessen, dass für jede Kombination von Knotenzuständen der Knoten, die in der Clique enthaltenen sind, ein Eintrag möglich ist. Dieser Tabelle werden die bedingten Wahrscheinlichkeiten und die A-priori-Wahrscheinlichkeiten der Knoten hinzumultipliziert, die der Clique durch die Funktion $f : \mathbb{V} \to \mathbb{C}$ zugeordnet sind.

Für diese Multiplikation werden im folgenden Abschnitt spezielle Tabellen über Knotenmengen eingeführt, die die bedingten Wahrscheinlichkeiten und A-priori-Wahrscheinlichkeiten der Knoten repräsentieren. Auf ihnen werden dann die Multiplikation und weitere Operationen definiert, die für die folgenden Inferenzalgorithmen notwendig sind. Welche Tabellen den Separatorknoten zugeordnet werden und welche zusätzlichen Tabellen den Cliquen dann hinzumultipliziert werden, wird in den beiden Abschnitten 2.3.3.1 und 2.3.3.2 der entsprechenden Inferenzalgorithmen besprochen, da diese unterschiedlich vorgehen.

### 2.3.2 Algebra von Wahrscheinlichkeitentabellen

Die A-priori- und bedingten Wahrscheinlichkeiten der Zufallsvariablen müssen dem Knoten zugeordnet werden, der mit der Zufallsvariablen assoziiert ist. In dieser Arbeit geschieht dies durch *Wahrscheinlichkeitentabellen*, die die A-priori-




(a) moralisierter Graph

(b) Triangulierung

(c) Triangulierung

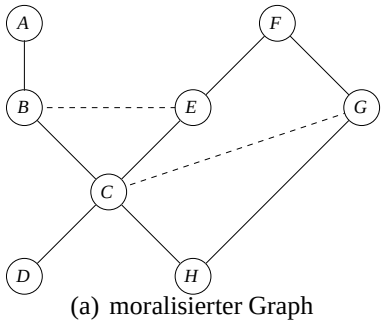
(d) Junction Graph

(e) Junction Graph

(f) Junction Tree

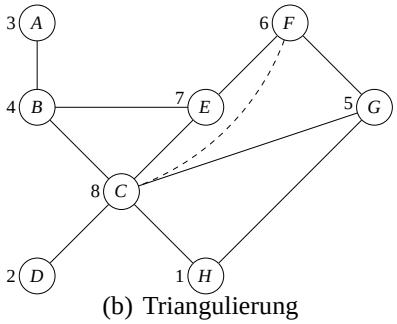
(g) Junction Tree

Abbildung 2.3: Bestimmung des Junction Trees am Beispielnetz Gehirntumor

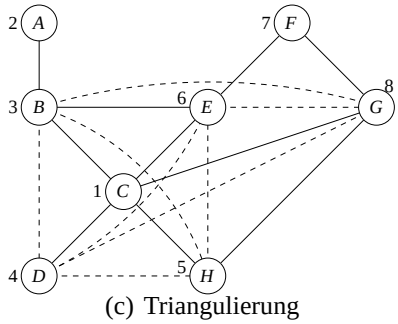



(a) moralisierter Graph

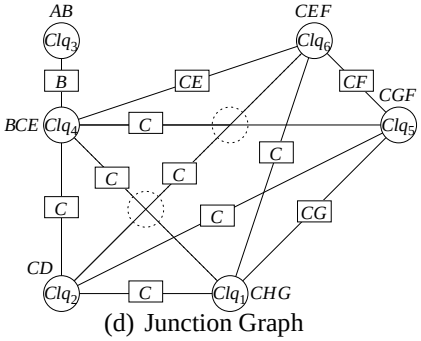
| gute Eliminationsreihenfolge $(H,D,A,B,G,F,E,C)$ | schlechte Eliminationsreihenfolge $(C,A,B,D,H,E,F,G)$ |
|---|---|



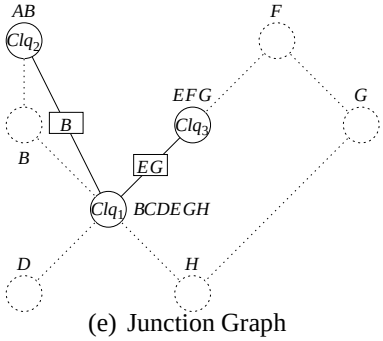

(b) Triangulierung



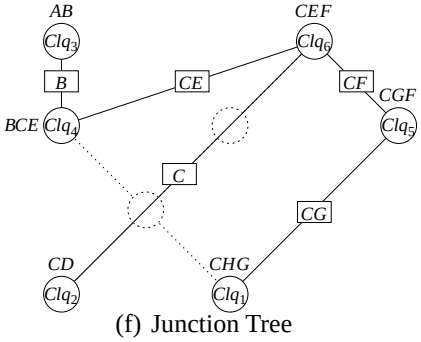

(c) Triangulierung



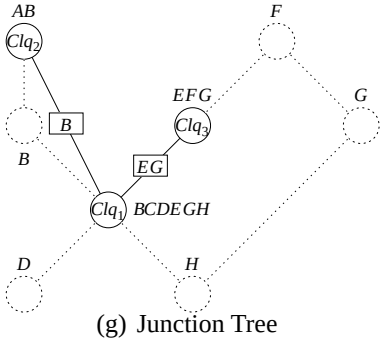

(d) Junction Graph




(e) Junction Graph




(f) Junction Tree




(g) Junction Tree

Abbildung 2.4: Bestimmung des Junction Trees am Beispielnetz Asienbesuch

oder bedingten Wahrscheinlichkeiten jeweils einer Zufallsvariablen des Bayesschen Netzes zusammenfasst. Wahrscheinlichkeitentabellen sind bestimmte Tabellen über Knotenmengen. Alle notwendigen Berechnungen können als Berechnungen von Tabellen über Knotenmengen angegeben werden. Im folgenden wird kurz beschrieben, welche Operationen mit Tabellen über Knotenmengen möglich sind und gebraucht werden, und welche besonderen Tabellen über Knotenmengen existieren. Genauere Informationen sind im Anhang A.1 angegeben. Hier wird beispielsweise darauf eingegangen, wie die Operationen ausgeführt werden und was beachtet werden kann, um die Operationen effizient (bzgl. Speicherverbrauch und Rechengeschwindigkeit) auszuführen.

Eine Tabelle über einer Knotenmenge $\mathbb{V} = \{K_1, \ldots, K_n\}$ ist eine Tabelle, die für jede Konfiguration $\langle k_1, \ldots, k_n \rangle$ einen Eintrag enthält. Sie wird mit $\theta_{(K_1,\ldots,K_n)}$ bezeichnet. Mit $\theta_{(K_1,\ldots,K_n)\langle k_1,\ldots,k_n\rangle}$ kann auf eine Konfiguration $\langle k_1, \ldots, k_n \rangle$ zugegriffen werden.

Mit $\theta_{1(K_1,\ldots,K_n)}$ wird die Tabelle über der Knotenmenge $\{K_1, \ldots, K_n\}$ bezeichnet, die nur Einsen als Einträge in der Tabelle besitzt.

Eine Tabelle bedingter Wahrscheinlichkeiten des Knotens $K$ ist eine Tabelle $\theta_{(K|\mathbf{pa}(K))}$ über der Knotenmenge $\{K\} \cup \mathbf{pa}(K)$, wobei man anstatt $\theta_{(K|\mathbf{pa}(K))}$ vereinfachend auch $\theta_{(K)}$ schreibt.

Eine Tabelle gemeinsamer Wahrscheinlichkeiten der Knoten $K_1, \ldots, K_n$ ist eine Tabelle $\theta_{(K_1,\ldots,K_n)}$ über der Knotenmenge $\mathbb{V} = \{K_1, \ldots, K_n\}$. Um bei einer Tabelle gemeinsamer Wahrscheinlichkeiten sicher zu stellen, dass sich die Tabelleneinträge insgesamt zu Eins addieren, wird die Tabelle mit einem Normalisierungsfaktor versehen. Dieser Normalisierungsfaktor wird auch als *Normalisierungskonstante* $\alpha$ bezeichnet.

Ein finding eines Knotens $K$ ist eine eindimensionale Tabelle $\lambda_K$ über dem Knoten $K$, die durch eine 0 an der entsprechenden Stelle angibt, dass diese Hypothese des Knotens $K$ nicht mehr eintreten wird. Hypothesen, die noch eintreten können, sind durch eine 1 an der entsprechenden Stelle der Tabelle markiert. Zum Beispiel gibt die Tabelle $\lambda_E = \left[\begin{array}{ccc} e_1 & e_2 & e_3 \\ \hline 0 & 1 & 1 \end{array}\right]$ an, dass nur noch die Hypothesen $e_2$ und $e_3$ des Knotens $E$ eintreten können. Die Hypothese $e_1$ des Knotens $E$ wird nicht mehr eintreten.

Die oben vorgestellten Tabellen werden zusammenfassend auch als *Wahrscheinlichkeitentabellen* bezeichnet.

Jetzt folgen die verschiedenen Operationen, die auf den Tabellen und die mit den Tabellen notwendig sind. Dazu zählen die *Multiplikation*, die *Division* sowie die *Marginalisation* von Tabellen.

- Die **Multiplikation** zweier Tabellen ist abelsch. Sind die Tabellen $\theta_{(A,B)}$ und $\theta_{(A,C)}$ über der Knotenmenge $\{A,B\}$ bzw. $\{A,C\}$ gegeben, so ist das

Produkt der beiden Tabellen eine Tabelle $\theta_{(A,B,C)}$ über der Knotenmenge $\{A,B\} \cup \{A,C\} = \{A,B,C\}$. Für eine Konfiguration $\langle a_i, b_j, c_k \rangle$ berechnet sich der Tabelleneintrag von $\theta_{(A,B,C)}$ zu:

$$\theta_{(A,B,C)\langle a_i,b_j,c_k\rangle} = \theta_{(A,B)\langle a_i,b_j\rangle} \; \theta_{(A,C)\langle a_i,c_k\rangle}.$$

- Die **Division** der Tabelle $\theta_{(A,B)}$ über der Knotenmenge $\{A,B\}$ durch die Tabelle $\theta_{(A,C)}$ über der Knotenmenge $\{A,C\}$ ist wie bei der Multiplikation dieser beiden Tabellen eine Tabelle $\theta_{(A,B,C)}$ über der Knotenmenge $\{A,B\} \cup \{A,C\} = \{A,B,C\}$. Für eine Konfiguration $\langle a_i, b_j, c_k \rangle$ berechnet sich der Tabelleneintrag von $\theta_{(A,B,C)}$ zu:

  $$\theta_{(A,B,C)\langle a_i,b_j,c_k\rangle} = \frac{\theta_{(A,B)\langle a_i,b_j\rangle}}{\theta_{(A,C)\langle a_i,c_k\rangle}}.$$

  Ist $\theta_{(A,C)\langle a_i,c_k\rangle} = 0$, so muss auch $\theta_{(A,B)\langle a_i,b_j\rangle}$ gleich 0 sein, und der Wert von $\theta_{(A,B,C)\langle a_i,b_j,c_k\rangle}$ wird auf 0 gesetzt, d. h. es gilt: $\frac{0}{0} = 0$.

- Um aus einer Tabelle $\theta_{(A,B,C,D)}$ über der Knotenmenge $\{A,B,C,D\}$ Information über eine Teilmenge von Knoten –zum Beispiel $\{B,C\}$– zu erhalten, wird über die restlichen Knoten der Tabelle addiert. Man erhält eine Tabelle $\theta_{(B,C)}$ über der Knotenmenge $\{B,C\}$:

  $$\theta_{(B,C)} = \sum_{\{A,D\}} \theta_{(A,B,C,D)}.$$

  Man bezeichnet dies auch als **Marginalisation**.

In den folgenden Abschnitten werden zwei verschiedene Architekturen vorgestellt, die die Algebra von Wahrscheinlichkeitentabellen für den Inferenzalgorithmus benötigen.

### 2.3.3 Inferenzverfahren

Berechnungen auf dem Junction Tree, um die Wahrscheinlichkeiten für das Eintreffen gewisser Hypothesen zu ermitteln, werden auch als *Inferenz* oder *Propagierung* bezeichnet. Es gibt verschiedene Inferenzverfahren, wie zum Beispiel die Aalborg-Architektur oder die Shafer-Shenoy-Architektur, die beide vorgestellt werden. Zunächst gehen wir auf die Gemeinsamkeiten der beiden Architekturen ein.

Für die Bestimmung der Cliquen und des *Junction Trees* sind dieselben Techniken anzuwenden, wie sie in Abschnitt 2.3.1 dargestellt sind. Sei nun der *Junction Tree* für das Bayessche Netz gegeben, das berechnet werden soll. Der Inferenzmechanismus gliedert sich bei beiden Architekturen in die zwei Phasen

1. *Collect Evidence* (1. Phase) und

2. *Distribute Evidence* (2. Phase),

die bei der *Initialisierung* des Junction Trees (*Init*) oder der *Auffrischung* des Junction Trees (*Update*) nach dem Eintragen neuer Beobachtungen hintereinander ausgeführt werden.

Bevor die 1. Phase *Collect Evidence* aufgerufen wird, wird noch der *aufrufende Knoten* bestimmt. Dazu wird ein Knoten $R$ zufällig als Wurzelknoten gewählt. Der Junction Tree $\mathscr{T}$ wird somit vom Knoten $R$ zu den Blattknoten gerichtet. Bis auf den Knoten $R$, der keine Elternknoten besitzt, haben alle anderen Knoten damit genau einen Elternknoten.

Die 1. Phase *Collect Evidence* läuft dabei wie folgt ab (siehe auch Abbildung 2.5(a)):

- Die Blattknoten schicken ihre Nachricht zum aufrufenden Knoten.
- Ein innerer Knoten schickt die Nachricht erst weiter, wenn er von allen Nachbarknoten bis auf einen die Nachricht erhalten hat. Die Nachricht wird an den Knoten[3] geschickt, von dem er noch keine Nachricht erhalten hat.
- Durch *Absorption* wird die Nachricht bestimmt, die verschickt wird.

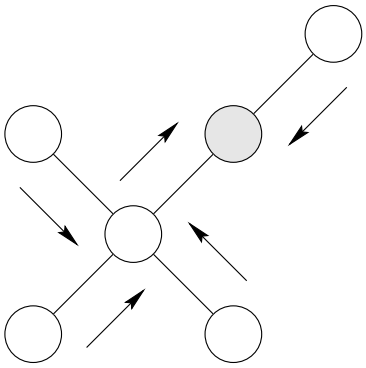
In der 2. Phase *Distribute Evidence* wird vom aufrufenden Knoten wie folgt eine Nachricht an die Blattknoten gesendet (siehe auch Abbildung 2.5(b)):

- Der aufrufende Knoten schickt seine Nachricht zu den Blattknoten des Baumes.
- Ein innerer Knoten erhält von einem Knoten die Nachricht. Die Nachricht wird an die Nachbarknoten weiter geschickt, von denen er keine Nachricht erhalten hat[4].
- Die Nachricht, die verschickt wird, wird durch *Absorption* bestimmt.

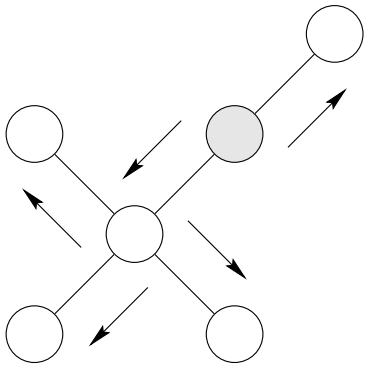
Die Aalborg- und die Shafer-Shenoy-Architektur unterscheiden sich in der *Absorption*, die in den Abschnitten 2.3.3.1 und 2.3.3.2 besprochen wird.

Nach einem vollen Durchlauf einer Initialisierung oder Auffrischung des Junction Trees sind die Informationen konsistent, die in den Cliquen des Junction Trees gespeichert sind: Ist man an der Einschätzung (BEL-Wert) über einer Knotenmenge $\mathbb{K}$ interessiert, so kann man diese aus jeder Clique des Junction Trees gewinnen,

[3] Dieser Knoten ist der Elternknoten des inneren Knotens.
[4] Diese Knoten sind die Kinderknoten des inneren Knotens.



(a) Collect Evidence: Die Blattknoten schicken ihre Nachricht zum aufrufenden Knoten.



(b) Distribute Evidence: Der aufrufende Knoten schickt seine Nachricht an die Blattknoten.

Abbildung 2.5: Die beiden Phasen Collect und Distribute Evidence. Der aufrufende Knoten ist grau schraffiert dargestellt.

in der die Knotenmenge enthalten ist. Man erhält immer dasselbe Ergebnis. Man sagt auch: Die Cliquen des Junction Trees sind zueinander *kalibriert*.

Ist man also an den BEL-Werten für alle Knoten interessiert, so müssen in der zweiten Phase (Distribute Evidence) alle Äste des *Junction Trees* besucht werden. Hat man jedoch nur ein Interesse an BEL-Werten gewisser Knoten, so kann man bei Distribute Evidence die Äste des *Junction Trees* aus Performanz-Gründen auslassen, in denen sich nur noch Knoten befinden, deren BEL-Werte von keinem Interesse mehr ist. (Dieses wird später bewusst beim *Rollup* (siehe Kapitel 3.3) ausgenutzt, da hier die alten Cliquen, die den alten Zeitscheiben zugeordnet sind, bei Distribute Evidence ausgelassen werden.).

#### 2.3.3.1 Aalborg-Architektur

Im folgenden Abschnitt wird gezeigt, wie sich die Informationen in der Aalborg-Architektur berechnen, die bei Collect Evidence und Distribute Evidence von Clique zu Clique verschickt werden.

Damit jede Clique im Junction Tree konsistente Information besitzt, werden Nachrichten im Junction Tree verschickt. Das Versenden einer Nachricht von einer Clique zu einer anderen Clique wird *Absorption* genannt. In der *Aalborg-Architektur* (siehe [Andersen et al. 89] und [Jensen 96]) wird die *Absorption* wie folgt durchgeführt.

***Definition 2.3.1 (Absorption (Aalborg))***
Seien $Clq_i$ und $Clq_\ell$ benachbarte Cliquen mit dem Separator $Sep_{i,\ell}$, und seien $\theta_{Clq_i}$, $\theta_{Clq_\ell}$ und $\theta_{Sep_{i,\ell}}$ die zugehörigen Wahrscheinlichkeitentabellen über den entsprechenden Knotenmengen.

1. Berechne Tabelle $\theta^*_{Sep_{i,\ell}} = \sum_{Clq_i \setminus Sep_{i,\ell}} \theta_{Clq_i}$. (Marginalisation)
2. Berechne Tabelle $\theta^*_{Clq_\ell} = \theta_{Clq_\ell} \cdot \frac{\theta^*_{Sep_{i,\ell}}}{\theta_{Sep_{i,\ell}}}$.
3. Weise dem Separator $Sep_{i,\ell}$ die Tabelle $\theta^*_{Sep_{i,\ell}}$ zu.
4. Weise der Clique $Clq_\ell$ die Tabelle $\theta^*_{Clq_\ell}$ zu.

Man sagt, dass die Clique $Clq_\ell$ von der Clique $Clq_i$ *absorbiert* hat. **D**

Die Absorption der Clique $Clq_\ell$ von der benachbarten Clique $Clq_i$ ist in Abbildung 2.6 grafisch dargestellt. Dieser Abbildung und auch der Definition 2.3.1 ist zu entnehmen, welche Tabellen den Cliquen $Clq \in \mathbb{C}$ und Separatorknoten $Sep \in \mathbb{S}$ des Junction Trees $\mathscr{T} = (\mathbb{C}, \mathbb{S})$ zugeordnet sind.

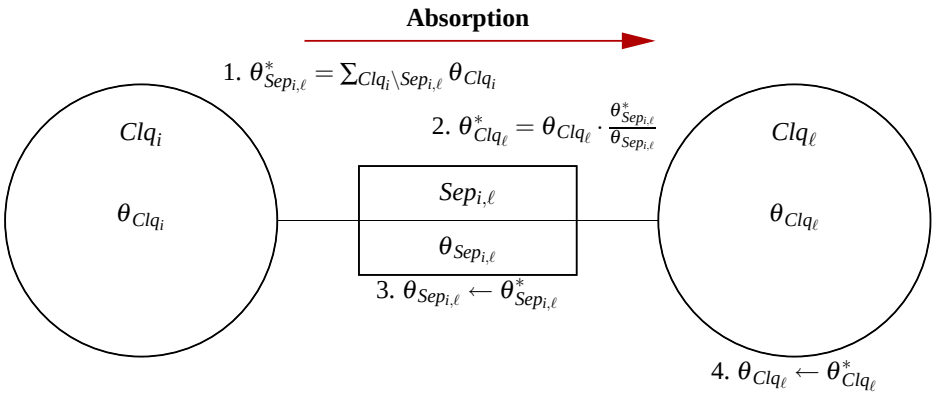



Abbildung 2.6: Aalborg: Absorption der Clique $Clq_\ell$ von der Clique $Clq_i$

Der Clique $Clq_i$ wird eine Tabelle über der Knotenmenge $Clq_i$ zugewiesen, die im Anfang nur Einsen als Einträge besitzt. Dieser Tabelle werden nun die Tabellen, die die bedingten Wahrscheinlichkeiten bzw. A-priori-Wahrscheinlichkeiten ihrer ihr zugeordneten Knoten repräsentieren, hinzumultipliziert. Die der Clique $Clq_i$ zugeordnete Tabelle wird mit $\theta_{Clq_i}$ bezeichnet. Dem Separatorknoten $Sep_{i,\ell}$, der die Knotenmenge $Clq_i \cap Clq_\ell$ enthält, wird eine Tabelle über der Knotenmenge $Clq_i \cap Clq_\ell$ zugeordnet, die wie vor im Anfang nur Einsen als Einträge enthält und mit $\theta_{Sep_{i,\ell}}$ bezeichnet wird. Die Einträge dieser Tabelle ändern sich nur durch Absorption. Liegen für Knoten *findings* vor, d. h. dass einige Zustände der dem Knoten zugehörige Zufallsvariable unmöglich geworden sind, werden diese *findings* in der Clique eingetragen, die dem Knoten zugeordnet ist. Ein finding eines Knotens $K$ ist eine eindimensionale Tabelle $\lambda_K$ über dem Knoten $K$, die durch eine 0 an der entsprechenden Stelle angibt, dass diese Hypothese des Knoten $K$ nicht mehr eintreten wird. Hypothesen, die noch eintreten können,

sind durch eine 1 an der entsprechenden Stelle der Tabelle markiert. Zum Beispiel gibt die Tabelle $\lambda_E = \left[\begin{array}{ccc} e_1 & e_2 & e_3 \\ \hline 0 & 1 & 1 \end{array}\right]$ an, dass nur noch die Hypothesen $e_2$ und $e_3$ des Knotens $E$ eintreten können. Die Hypothese $e_1$ des Knotens $E$ wird nicht mehr eintreten.

Im Abschnitt B.1 werden an einem Beispiel die beiden Phasen Collect Evidence und Distribute Evidence beim Init und beim Update eines Junction Trees ausführlich vorgestellt.

#### 2.3.3.2 Shafer-Shenoy-Architektur

Bei der Shafer-Shenoy-Architektur sind nicht wie bei der Aalborg-Architektur die an die Clique verschickten Nachrichten in der Clique selbst sondern in den Separatoren gespeichert, die sich in der Nachbarschaft der Clique befinden. Die sich daraus ergebenden Änderungen für die Absorption werden im folgenden Abschnitt vorgestellt.

Bei der *Shafer-Shenoy-Architektur* sind im Gegensatz zur *Aalborg-Architektur* in jedem Separator zwei Tabellen gespeichert, nämlich jeweils eine für jede Richtung. In der einen Tabelle wird die Nachricht gespeichert, die die erste Clique zur zweiten Clique schickt, und in der anderen Tabelle wird die Nachricht gespeichert, die die zweite Clique zur ersten Clique zurückschickt. In jeder Clique ist eine Tabelle gespeichert.

Die hier vorgestellte *Shafer-Shenoy-Architektur* wurde im Vergleich zur Shafer-Shenoy-Architektur, wie sie in [Shafer 96] vorgestellt wird, vom Autor so abgewandelt, dass die Tabellen der Cliquen nicht explizit als Tabellen sondern nur als Berechnungsvorschriften (Skripte) gespeichert sind. Dies ist auch ein Unterschied zur Aalborg-Architektur, bei der die Tabelle explizit ausgerechnet werden. Dadurch ist bei der Absorption in der Shafer-Shenoy-Architektur im Gegensatz zur *Aalborg-Architektur* keine Division notwendig. Als weiterführende Literatur für die Shafer-Shenoy-Architektur und den Vergleich mit anderen Algorithmen bieten sich unter anderem [Shafer 96, Madsen & Jensen 98, Xiang & Jensen 99] an.

Aus den Cliquen und Separatorknoten lassen sich nach erfolgreicher Propagierung die BEL-Werte der interessierenden Knoten berechnen. Dieses ist in Abbildung 2.7 grafisch dargestellt. In den Separatoren sind die Informationen gespeichert, die die benachbarten Cliquen an Clique $Clq_i$ geschickt haben ($\theta_{Clq \triangleright Clq_i}$). Heißt der Separatorknoten $Sep_{i,\ell}$, so stehen in ihm die beiden Tabellen $\theta_{Clq_i \triangleright Clq_\ell}$ und $\theta_{Clq_\ell \triangleright Clq_i}$. Die Informationen, die die Clique $Clq_i$ an die Clique $Clq_\ell$ schicken, stehen in einer Tabelle $\theta_{Clq_i \triangleright Clq_\ell}$ über der Knotenmenge $Clq_i \cap Clq_\ell$. Die Clique $Clq_\ell$ wird dabei von der Clique $Clq_i$ über die Knoten informiert, die beide Cliquen gemeinsam haben (nämlich $Clq_i \cap Clq_\ell$). In der Clique $Clq_i$ sind die Informationen ($\theta_{(.)}$ und $\lambda_{(.)}$) der Knoten gespeichert, die der Clique durch die

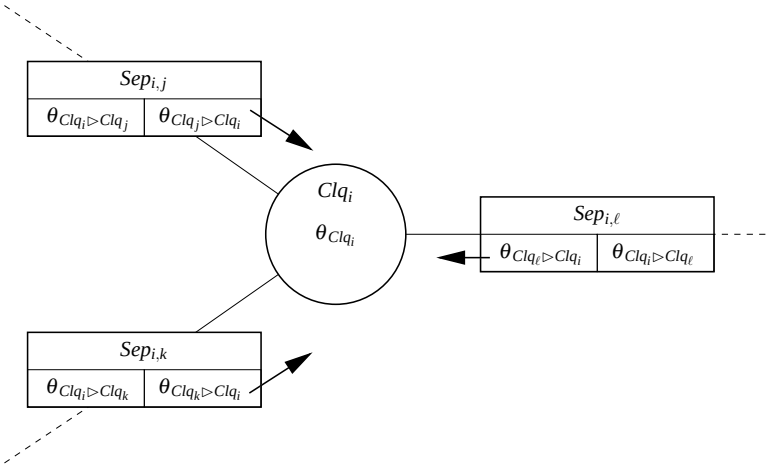



Abbildung 2.7: Shafer-Shenoy: Berechnung der Cliqueninfo nach erfolgreicher Propagierung.

Funktion $f : \mathbb{V} \rightarrow \mathbb{C}$ zugeordnet sind.

Somit ergibt sich als Tabelle für die Clique $Clq_i$:

$$\theta_{Clq_i} = \theta_{1(Clq_i)} \prod_{K \in \{K | f(K) = Clq_i\}} \left(\theta_{(K)}\ \lambda_K\right) \prod_{Clq \in \mathrm{adj}(Clq_i)} \theta_{Clq \triangleright Clq_i}.$$

Für die Tabelle der Clique $Clq_i$ ohne die Nachricht der benachbarten Clique $Clq_\ell$ schreibt man auch $\theta_{Clq_i \setminus Clq_\ell}$ und meint damit:

$$\theta_{Clq_i \setminus Clq_\ell} = \theta_{1(Clq_i)} \prod_{K \in \{K | f(K) = Clq_i\}} \left(\theta_{(K)}\ \lambda_K\right) \prod_{Clq \in \mathrm{adj}(Clq_i) \setminus Clq_\ell} \theta_{Clq \triangleright Clq_i}.$$

Ist der Clique $Clq_i$ kein Knoten zugeordnet, so gilt:

$$\prod_{K \in \{K | f(K) = Clq_i\}} \left(\theta_{(K)}\ \lambda_K\right) = \prod_{K \in \{\}} \left(\theta_{(K)}\ \lambda_K\right) = 1.$$

In der Shafer-Shenoy-Architektur wird die Absorption von Clique $Clq_i$ zu Clique $Clq_\ell$ wie folgt durchgeführt.

***Definition 2.3.2 (Absorption (Shafer-Shenoy))***
Seien $Clq_i$ und $Clq_\ell$ zwei benachbarte Cliquen mit dem Separator $Sep_{i,\ell}$. Die benachbarten Cliquen von $Clq_i$ seien mit adj($Clq_i$) bezeichnet. Die verschiedenen Wahrscheinlichkeitentabellen seien wie üblich definiert. Dann berechnet sich die

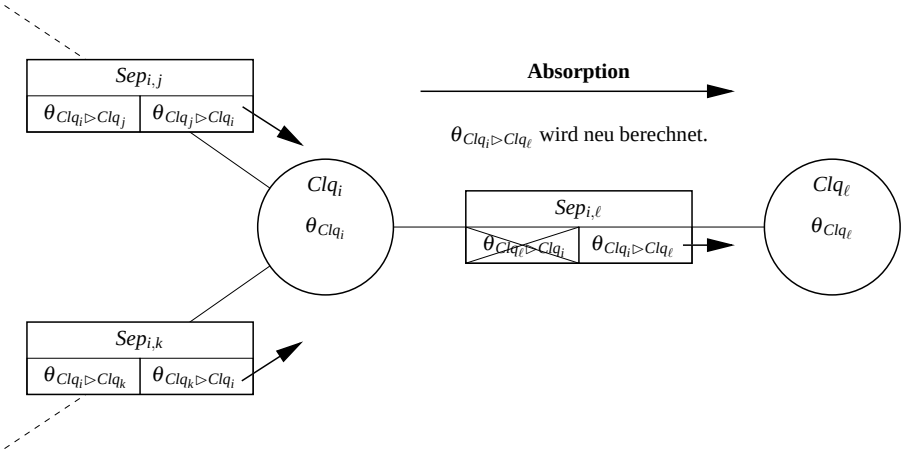



Abbildung 2.8: Shafer-Shenoy: Absorption der Clique $Clq_\ell$ von der Clique $Clq_i$

Nachricht von $Clq_i$ zu $Clq_\ell$ wie folgt:

$$\begin{aligned}\theta_{Clq_i \triangleright Clq_\ell} &= \sum_{Clq_i \setminus Clq_\ell} \theta_{Clq_i \setminus Clq_\ell} = \\ &= \sum_{Clq_i \setminus Clq_\ell} \Bigg( \theta_{1(Clq_i)} \prod_{K \in \{K \mid f(K) = Clq_i\}} \left(\theta_{(K)}\ \lambda_K\right) \\ &\qquad \prod_{Clq \in \mathrm{adj}(Clq_i) \setminus Clq_\ell} \theta_{Clq \triangleright Clq_i} \Bigg).\end{aligned}$$

D

Man beachte, dass im Vergleich zur Aalborg-Propagierung keine Division erfolgt. Die Tabelle, durch die bei der Aalborg-Propagierung dividiert wird, wird in der Shafer-Shenoy-Architektur gar nicht erst hinzu multipliziert.

In Kapitel B.2 wird eine ausführliche Beispielrechnung für die Shafer-Shenoy-Architektur durchgeführt. Um dabei die Shafer-Shenoy-Architektur mit der Aalborg-Architektur vergleichen zu können, wurde dasselbe Beispiel wie bei der Beispielrechnung für die Aalborg-Architektur (siehe Kapitel B.1) herangezogen.

In [Brandherm 00] wird vom Autor die Shafer-Shenoy-Architektur zu Shafer-Shenoy-Skripten umgeformt und zur Polynom-Propagierung erweitert, die nur für bestimmte Anfragen, d. h. für Cliquen im Junction Tree, jeweils ein multivariates Polynom in Abhängigkeit der Evidenz-Variablen erzeugt.

Im folgenden Abschnitt werden wir nun bezüglich Polynomen ein ganzheitliches Verfahren vorstellen, das ein Bayessches Netz in seiner Gesamtheit als Polynom (zunächst in kanonischer und dann in faktorisierter Form) darstellt und auswertet. Dabei wird der Junction Tree, der für die beiden zuvor eingeführten

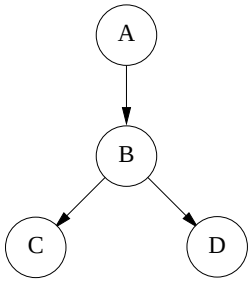



Abbildung 2.9: Beispiel für ein Bayessches Netz mit vier binären Knoten [$\Pr(A,B,C,D) = \Pr(A)\ \Pr(B \mid A)\ \Pr(C \mid B)\ \Pr(D \mid B)$]

Verfahren als Rechengrundlage dient, herangezogen werden, um ein faktorisiertes Polynom zu erhalten, welches im allgemeinen weniger komplex ist als in seiner kanonischen Darstellung.

## 2.4 Differentieller Ansatz für Bayessche Netze

Der differentielle Ansatz für Bayessche Netze ist ein Verfahren, das ein Bayessches Netz in ein multivariates Polynom kompiliert, welches effizient verarbeitet werden kann: Nachdem das Polynom und seine partiellen Ableitungen bestimmt wurde, lässt sich eine Vielzahl von Wahrscheinlichkeitsberechnungen wie beispielsweise die klassische Inferenz oder eine Sensitivitätsanalyse in konstanter Zeit durchführen (siehe [Darwiche 99, Darwiche 00, Darwiche 03]).

In diesem Abschnitt fassen wir den differentiellen Ansatz für Bayessche Netze zusammen, wobei wir auf die *kanonische* und die *faktorisierte* Darstellung der multivariaten Polynome eingehen, die das Bayessche Netz repräsentieren. Wir zeigen, welche Anfragen an das Bayessche Netz mit dem Polynom beantwortet werden können.

Die in [Brandherm 00] entwickelten Verfahren *Polynompropagierung* und *Shafer-Shenoy-Skripte* generieren nur auf bestimmte Anfragen spezialisierte Polynome und werden somit durch diesen Ansatz subsumiert, der das Bayessche Netz ganzheitlich betrachtet.

Der differentielle Ansatz für Bayessche Netze wird in Abschnitt 4 für dynamische Bayessche Netz erweitert werden. In Abschnitt 6 stellen wir die Anwendung *JavaDBN* vor, die dynamische Bayessche Netze in selbstausführbaren Programmcode mit Rollup alter Zeitscheiben erzeugt, und sozusagen eine integrierte Lösung dynamischer Bayesscher Netze für den Einsatz auf eingebetteten Systemen mit geringen Ressourcen anbietet.

### 2.4.1 Kanonische Darstellung des Polynoms

Jedes Bayessche Netz mit diskreten Variablen kann als Polynom dargestellt werden. In Abbildung 2.9 betrachten wir z. B. erst nur die beiden Knoten $A$ und $B$ und deren assoziierten Tabellen bedingter Wahrscheinlichkeiten $\theta_{(A)}$ und $\theta_{(B|A)}$. Die beiden möglichen Werte von $A$ werden mit $a_1$ und $a_2$ bezeichnet, während die beiden möglichen Werte von $B$ mit $b_1$ und $b_2$ bezeichnet werden. Nehmen wir beispielsweise an, dass wir Evidenz $\mathbf{e} = a_1$ vorliegen haben und an $\Pr(\mathbf{e})$ interessiert sind. Indem wir die beiden Tabellen bedingter Wahrscheinlichkeiten miteinander multiplizieren, erhalten wir eine Gesamtwahrscheinlichkeitentabelle $\theta_{(A,B)}$, die die gemeinsame Wahrscheinlichkeitsverteilung der beiden Knoten $A$ und $B$ darstellt:

$$\theta_{(A,B)} = \theta_{(A)}\theta_{(B|A)} = \begin{bmatrix} \theta_{a_1} \\ \theta_{a_2} \end{bmatrix} \begin{bmatrix} \theta_{b_1|a_1} & \theta_{b_1|a_2} \\ \theta_{b_2|a_1} & \theta_{b_2|a_2} \end{bmatrix} = \begin{bmatrix} \theta_{a_1}\theta_{b_1|a_1} & \theta_{a_2}\theta_{b_1|a_2} \\ \theta_{a_1}\theta_{b_2|a_1} & \theta_{a_2}\theta_{b_2|a_2} \end{bmatrix}.$$

Nun müssen wir nur noch die Tabelleneinträge (Wahrscheinlichkeiten) aufaddieren, die mit $a_1$ konsistent sind, und wir erhalten das gewünschte Ergebnis:

$$\Pr(\mathbf{e}) = \theta_{a_1}\theta_{b_1|a_1} + \theta_{a_1}\theta_{b_2|a_1}.$$

Für jede Variable im Bayesschen Netz ist es möglich, dass wir Evidenz eintragen. Solche Evidenz kann als *Evidenz-Vektor* $\lambda_{X_i}$ für den fraglichen Knoten dargestellt werden. Der Vektor enthält an der Stelle eine 1, dessen Zustand festgestellt wurde und eine 0 für alle anderen Zustände der Variablen. (Wenn für eine Variable keine Evidenz vorliegt, so ist jedes Element des Vektors 1.)

Wir können den Evidenz-Vektor wie folgt verwenden, um die Wahrscheinlichkeitentabelle zu parametrisieren, so dass sie vorhandene Evidenz berücksichtigen kann:

$$\theta_{(A,B)}\lambda_A\lambda_B = \theta_{(A)}\lambda_A\ \theta_{(B|A)}\lambda_B = \begin{bmatrix} \theta_{a_1}\theta_{b_1|a_1}\lambda_{a_1}\lambda_{b_1} & \theta_{a_2}\theta_{b_1|a_2}\lambda_{a_2}\lambda_{b_1} \\ \theta_{a_1}\theta_{b_2|a_1}\lambda_{a_1}\lambda_{b_2} & \theta_{a_2}\theta_{b_2|a_2}\lambda_{a_2}\lambda_{b_2} \end{bmatrix}.$$

Indem wir alle Tabelleneinträge aufsummieren, erhalten wir ein multivariates Polynom:

$$\begin{aligned} \mathscr{F}(\lambda_A, \lambda_B, \theta_{(A)}, \theta_{(B|A)}) = {} & \theta_{a_1}\theta_{b_1|a_1}\lambda_{a_1}\lambda_{b_1} + \theta_{a_2}\theta_{b_1|a_2}\lambda_{a_2}\lambda_{b_1} + \\ & \theta_{a_1}\theta_{b_2|a_1}\lambda_{a_1}\lambda_{b_2} + \theta_{a_2}\theta_{b_2|a_2}\lambda_{a_2}\lambda_{b_2}\ . \end{aligned}$$

Nun können wir über die Evidenz-Vektoren bestimmen, welche Tabelleneinträge aufsummiert werden sollen. Die $\lambda_{x_i}$, die mit der Evidenz $\mathbf{e}$ konsistent sind, werden auf 1 gesetzt, während die anderen $\lambda_{x_i}$ auf 0 gesetzt werden. In unserem Beispiel erhalten wir somit $\lambda_{a_2} = 0$ und $\lambda_{a_1} = \lambda_{b_1} = \lambda_{b_2} = 1$.

Mit diesem Polynom lässt sich beispielsweise der BEL-Wert für bestimmte Knoten im Netz berechnen, und wir können verschiedene Sensitivitätsanalysen durchführen (siehe [Darwiche 03]).

Im allgemeinen berechnet sich das multivariate Polynom wie folgt: Als erstes werden die Tabellen bedingter Wahrscheinlichkeiten aller Knoten des Bayesschen Netzes und ihre Evidenz-Vektoren $\lambda_{X_i}$ miteinander multipliziert, und man erhält eine Tabelle **T**, die die Wahrscheinlichkeiten der einzelnen Hypothesenkombinationen enthält:

$$\mathbf{T} = \prod_{i=1}^{n} \theta_{(X_i|\mathbf{pa}(X_i))} \lambda_{X_i} \ .$$

Indem wir alle Tabelleneinträge von **T** aufaddieren, erhalten wir das multivariate Polynom in kanonischer Form (siehe auch [Darwiche 00]):

$$\mathscr{F}(\lambda_{X_i}, \theta_{(X_i|\mathbf{pa}(X_i))}) = \sum_{\mathbf{X}} \prod_{i=1}^{n} \theta_{(X_i|\mathbf{pa}(X_i))} \lambda_{X_i} \ .$$

Die Größe des multivariaten Polynoms in kanonischer Form ist immer exponentiell in der Anzahl seiner Variablen. Eine Möglichkeit, diese sich ergebende Komplexität zu verhindern, besteht darin, das Polynom faktorisiert darzustellen, wie es im nachfolgenden Abschnitt beschrieben wird.

### 2.4.2 Faktorisierte Darstellung des Polynoms

Wir wollen bei der Bestimmung der Faktorisierung des multivariaten Polynoms die Struktur des Bayesschen Netzes berücksichtigen und die faktorisierte Darstellung der Gesamtwahrscheinlichkeitsverteilung dazu ausnutzen, die Komplexität des multivariaten Polynoms zu verringern. Wir werden nicht einfach das Produkt aller Tabellen bedingter Wahrscheinlichkeiten und ihrer Evidenz-Vektoren berechnen, sondern zuerst eine Reihenfolge der Knoten bestimmen, in der die Tabellen bedingter Wahrscheinlichkeiten und ihre dazugehörigen Evidenz-Vektoren $\lambda_{X_i}$ miteinander multipliziert werden sollen. Frühzeitig werden wir in diesem Prozess so viele Knoten wie möglich aus dem Produkt herausmarginalisieren, so dass wir das multivariate Polynom in faktorisierter anstatt kanonischer Form erhalten.

Wir wollen das noch einmal am Beispiel der Knoten $A$ und $B$ veranschaulichen. Angenommen, dass die Abarbeitungsreihenfolge der Knoten lautet “Erst $B$, dann $A$”. Diese Reihenfolge wird auch als *Eliminationsreihenfolge* $\pi = \langle B, A \rangle$ bezeichnet. Wenn wir die Knoten so früh wie möglich herausmarginalisieren, so

erhalten wir:

$$\mathscr{F}_{\langle B,A\rangle}(\lambda_A,\lambda_B,\theta_{(A)},\theta_{(B|A)}) = \sum_A \theta_{(A)}\lambda_A \left(\sum_B \theta_{(B|A)}\lambda_B\right) =$$
$$\theta_{a_1}\lambda_{a_1}(\theta_{b_1|a_1}\lambda_{b_1} + \theta_{b_2|a_1}\lambda_{b_2}) + \theta_{a_2}\lambda_{a_2}(\theta_{b_1|a_2}\lambda_{b_1} + \theta_{b_2|a_2}\lambda_{b_2}) \,.$$

Wenn wir die faktorisierte Form des Polynoms mit seiner kanonischer Form vergleichen, so stellen wir fest, dass es in der faktorisierten Form einige Multiplikationen nicht berechnet werden müssen. In Verbindung mit der Schreibweise der Eliminationsreihenfolge wollen wir bemerken, dass das folgende gilt:

$$\prod_{\langle B,A\rangle} \theta_{(X|\mathbf{pa}(X))}\lambda_X = \theta_{(A)}\lambda_A\ \theta_{(B|A)}\lambda_B$$

und

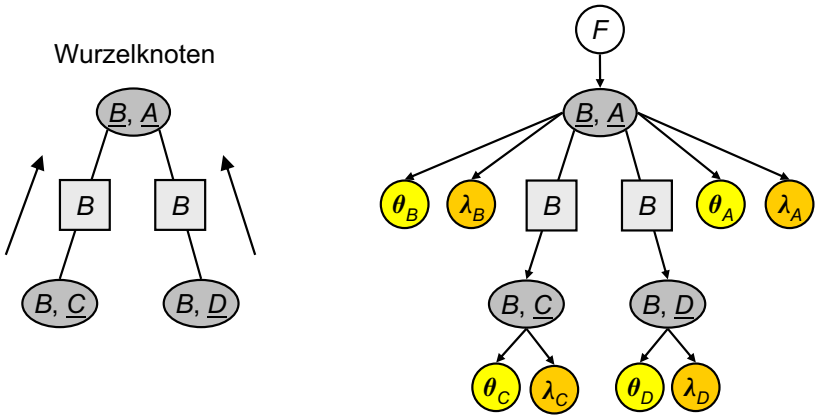
$$\sum_{\langle B,A\rangle} \theta_{(A)}\lambda_A\ \theta_{(B|A)}\lambda_B = \sum_{\langle A\rangle}\sum_{\langle B\rangle} \theta_{(A)}\lambda_A\ \theta_{(B|A)}\lambda_B = \sum_A \theta_{(A)}\lambda_A \left(\sum_B \theta_{(B|A)}\lambda_B\right).$$

Wir wollen die folgende Kurzschreibweise für das multivariate Polynom in faktorisierter Form verwenden:

$$\mathscr{F}_{\mathrm{elim}(\{X\in\mathscr{N}\})}(\lambda_X,\theta_{(X|\mathbf{pa}(X))}) =$$
$$\sum_{\mathrm{elim}(\{X\in\mathscr{N}\})} \prod_{\mathrm{elim}(\{X\in\mathscr{N}\})} (\theta_{(X|\mathbf{pa}(X))}\lambda_X) = \sum_{\langle X_1,\ldots,X_n\rangle} \prod_{\langle X_1,\ldots,X_n\rangle} (\theta_{(X|\mathbf{pa}(X))}\lambda_X) \,.$$

Hier ist $\mathrm{elim}(\{X \in \mathscr{N}\})$ eine Funktion, die die Eliminationsreihenfolge der Variablen berechnet, wobei es an dieser Stelle nicht von Interesse ist, welche Funktion nun genau dafür verwendet wird. Für mehr Hintergrundinformationen möchten wir auf [Kjærulff 95] verweisen, der verschiedene Heuristiken zur Bestimmung einer Eliminationreihenfolge vorstellt.

In Abbildung 2.10(a) ist ein möglicher Junction Tree des Bayesschen Netzes aus Abbildung 2.9 mit bereits bestimmtem Wurzelknoten gezeigt. Der Junction Tree besteht aus drei Cliquen und zwei Separatorknoten. Die unterstrichenen Knoten einer Clique wurden derjenigen Clique zugeordnet. Der Wurzelknoten gibt die Richtung der Inferenz vor, so dass wir nun das dazugehörige faktorisierte Polynom bestimmen können, das das Bayessche Netz berechnet. Die graphische Darstellung des faktorisierten Polynoms ist in Abbildung 2.10(b) zu sehen. Für die der Clique zugeordneten Knoten wurde jeweils ein Knoten mit der Tabelle bedingter Wahrscheinlichkeiten und ein Knoten mit dem Evidenzvektor angehängt. Aus dieser graphischen Darstellung lässt sich der arithmetische Schaltkreis ablesen (bei Cliquen wird multipliziert und bei Separatorknoten wird addiert), der in Abbildung 2.11 dargestellt ist. Dieser Schaltkreis berechnet das




(a) Junction Tree (b) Graphische Darstellung des Polynoms

Abbildung 2.10: Ein möglicher Junction Tree des Bayesschen Netzes in Abbildung 2.9 und das dazugehörige Polynom in graphischer Darstellung.

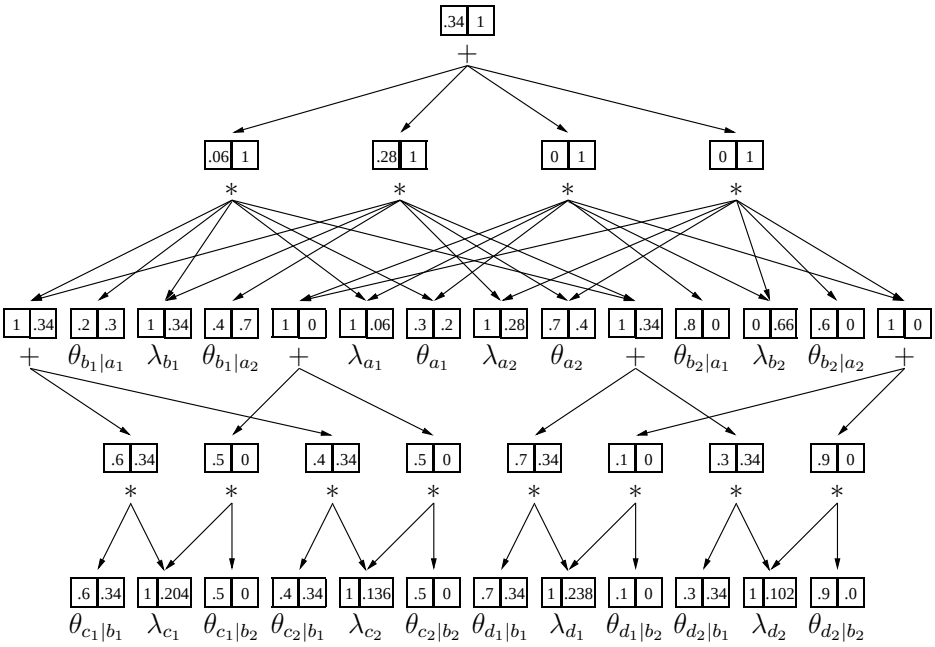



Abbildung 2.11: Ein arithmetischer Schaltkreis, der ein Polynom berechnet, das das Bayessche Netz in Abbildung 2.9 repräsentiert. Es wurde unter der Evidenz $b_1$ ausgewertet (Aufwärtsdurchgang) und differenziert (Abwärtsdurchgang). Die Register *vr* sind auf der linken Seite dargestellt, und die Register *dr* sind auf der rechten Seite dargestellt.

Polynom $\sum_{A,B} \theta_A \lambda_A \ \theta_{B|A} \lambda_B \ (\sum_C \theta_{C|B} \lambda_C) \ (\sum_D \theta_{D|B} \lambda_D)$, das das Bayessche Netz in Abbildung 2.9 repräsentiert.

Diese besondere Darstellung des Polynoms vereinfacht seine Auswertung und Differentiation. Wurde der arithmetische Schaltkreis einmal ausgewertet und differenziert, so lässt sich eine große Gruppe von Anfragen umgehend abfragen (siehe [Darwiche 03]).

Wir stellen hier nun einen einfachen Algorithmus für die Auswertung und Differentiation eines arithmetischen Schaltkreises vor, der auch im Fall der dynamischen Bayesschen Netze verwendet werden wird. Den technischen Hintergrund und zusätzliche Informationen über Algorithmen zur Auswertung und Differentiation findet man in [Park & Darwiche 04, Darwiche 03, Park & Darwiche 03]. Wir haben diesen Algorithmus bewusst gewählt, da er leicht verständlich ist und auch, wie wir in Kapitel 6 sehen werden, von Hand einfach an andere Gegebenheiten angepasst werden kann.

Jedem Schaltkreisknoten $v$ werden zwei Register $vr(v)$ und $dr(v)$ zugeordnet. Im Aufwärtsdurchgang werden die Werte der Register $vr(v)$ berechnet und der Schaltkreis somit ausgewertet. Im Abwärtsdurchgang werden die Register $dr(v)$ berechnet und der Schaltkreis dabei differenziert.

- Initialisierung: Setze für den Wurzelknoten $v$ $dr(v)$ auf 1; für alle anderen $v$ setze $dr(v)$ auf 0.
- Aufwärtsdurchgang:
  - Wenn der Knoten $p$ mit einem Addition-Zeichen beschriftet ist: Setze das Register $vr$ des Knoten $p$ auf $\sum_v vr(v)$, wobei die Knoten $v$ die Kinder von $p$ sind.
  - Wenn der Knoten $p$ mit einem Multiplikation-Zeichen beschriftet ist: Setze das Register $vr$ des Knoten $p$ auf $\prod_v vr(v)$, wobei die Knoten $v$ die Kinder von $p$ sind.
- Abwärtsdurchgang:
  - Wenn der Knoten $p$ mit einem Addition-Zeichen beschriftet ist: Für alle Kinder $v$ von $p$: Erhöhe $dr(v)$ um $dr(p)$.
  - Wenn der Knoten $p$ mit einem Multiplikation-Zeichen beschriftet ist: Für alle Kinder $v$ von $p$: Erhöhe $dr(v)$ um $dr(p) \prod_{v'} vr(v')$, wobei $v'$ die anderen Kinder von $p$ sind.

In Abbildung 2.11 sind die Werte der Register $vr$ und $dr$ nach dem Auf- und Abwärtsdurchgang unter der Evidenz $b_1$ dargestellt. Mit diesen Werten können

wir nun den BEL-Wert eines Knotens berechnen. Beispielsweise erhalten wir für den Knoten $C$ den BEL-Wert:

$$\mathrm{BEL}(C) = \left(\frac{0.204}{0.34}, \frac{0.136}{0.34}\right) = (0.6, 0.4)\,.$$

## 2.5 Zusammenfassung

Anhand von zwei kleinen Beispielen wurden die mehrfach verbundenen Bayesschen Netze und einige Begriffe informell eingeführt. Nach dieser Motivation wurden die notwendigen Notationen und Begriffe für Bayessche Netze definiert. Dann wurde der für die Inferenzverfahren notwendige Junction Tree eingeführt. Dabei wurde anhand der in der Motivation vorgestellten Beispielnetze gezeigt, welche Elementarschritte für die Bestimmung eines Junction Trees ausgeführt werden müssen. Man erkennt, wie wichtig eine optimale Eliminationsreihenfolge ist. Eine schlechte Eliminationsreihenfolge induziert eine große Anzahl von Triangulationskanten im moralisierten Graphen, was zur Folge einen Junction Tree haben kann, von dem einige Clique so groß werden, dass sie bezüglich Speicherplatzverbrauch und Berechnungsaufwand nicht mehr verarbeitet werden können. Desweiteren wurde eine Algebra von Wahrscheinlichkeitentabellen eingeführt, die eine Schreibweise benutzt, die sich auf die wesentlichen Zusammenhänge bezieht. So wird nicht zwischen bedingten und A-priori-Wahrscheinlichkeiten unterschieden und auch für die Instantiierung von Knoten mit bestimmten Werten werden Wahrscheinlichkeitentabellen benutzt. Als Inferenzverfahren wurden die Aalborg-Architektur und die Shafer-Shenoy-Architektur vorgestellt, die beide auf einem Junction Tree arbeiten und exakte Verfahren sind. Im Anschluss daran wurde der differentielle Ansatz von Darwiche vorgestellt, der ein Bayessches Netz in ein multivariates Polynom umwandelt. Dabei kann der für die beiden zuvor genannten anderen Inferenzverfahren erzeugte Junction Tree bei der Erzeugung des faktorisierten Polynoms herangezogen werden.

# Kapitel 3

# Dynamische Bayessche Netze

Dynamische Bayessche Netze sind spezielle Bayessche Netze, die sich aus sogenannten Zeitscheiben, d. h. Bayesschen Netzen mit spezieller Zusatzinformation, aufbauen. Als neue Zeitscheibe wird ein Bayessches Netz aus einer vorgegebenen Menge von Bayesschen Netzen an die letzte Zeitscheibe des dynamischen Bayesschen Netzes angehängt. Dabei lassen sich beim Anhängen von Zeitscheiben zwei Verfahren unterscheiden, die wir an einem Beispielnetz gegenüberstellen werden.

Eines dieser beiden Verfahren für das Anhängen von Zeitscheiben bildet die Grundlagen für einige der Rollup-Verfahren, die in Abschnitt 3.3 vorgestellt werden. Die Idee bei diesem Verfahren ist, dass sich der Aufbau des dynamischen Bayesschen Netzes aus Zeitscheiben im Junction Tree (siehe Abschnitt 2.3) widerspiegelt.

Schon mit Hidden Markov Modellen und Kalman Filtern lassen sich stochastische temporale Prozesse modellieren. So werden Hidden Markov Modelle zur Spracherkennung und Biosequenzanalyse verwendet. Kalman Filter werden eingesetzt, um die Flugbahn von Flugzeugen oder Raketen einzuschätzen. Jedoch sind Hidden Markov Modelle und Kalman Filter in ihrer Ausdrucksstärke eingeschränkt. Gegenüber Kalman Filtern erlauben dynamische Bayessche Netze eine beliebige anstatt einer (unimodalen) linearen Gaußschen Wahrscheinlichkeitsverteilung. Die Hidden Markov Modelle werden von den dynamischen Bayesschen Netzen verallgemeinert, da der Zustandsraum faktorisiert und nicht nur als eine einzelne, diskrete Zufallsvariable dargestellt werden kann. So lassen sich mit dynamischen Bayesschen Netzen komplexere Zusammenhänge als mit Hidden Markov Modellen modellieren und verarbeiten. [Zweig & Russell 98] setzen zur Spracherkennung (einem klassischen Gebiet für Hidden Markov Modelle) dynamische Bayessche Netze ein, um zusätzlich zu den phonetischen Zuständen, wie sie schon im Hidden Markov Modell modelliert wurden, auch die Artikulation und den akustischen Kontext zu modellieren und zu verarbeiten, was erst

durch die faktorisierte Darstellung des Zustandsraums ermöglicht wird. Ein anderes Beispiel aus der Spracherkennung wird in [Nefian et al. 02] beschrieben. Es werden dynamische Bayessche Netze für eine kontinuierliche audio-visuelle Spracherkennung vorgestellt, die sprecherunabhängig ist. Für eine bessere Spracherkennung werden bei diesem Verfahren zusätzlich die Lippen abgelesen und ihre Bewertung in der Spracherkennung berücksichtigt. Dynamische Bayessche Netze haben aber auch in anderen Bereichen als der Spracherkennung ihren Einsatz gefunden. Beispielsweise wird in [Dagum et al. 92] ein dynamisches Bayessches Netz verwendet, um den Verkauf amerikanischer Wagen auf dem japanischen Markt vorherzusagen. Im Projekt BAT[1] (siehe [Forbes et al. 95]) werden dynamische Bayessche Netze eingesetzt, um das Fahrzeug zu lokalisieren. Eine Anwendung aus dem Bereich der *Computer Vision* wird in [Gong & Xiang 03] beschrieben. Es werden dynamische Bayessche Netze eingesetzt, um Gruppenaktivitäten an einer Flughafenrampe im Freien zu erkennen, wie z. B. das Ent- und Beladen der Fracht eines Flugzeuges, wobei die auszuwertenden Bilder, die von den zur Überwachung typischerweise eingesetzten Videokameras geliefert werden, nur in geringer Auflösung vorliegen und teilweise stark verrauscht sind.

Mit *dynamischen Bayesschen Netzen* lassen sich stochastische temporale Prozesse modellieren (vgl. [Nicholson & Brady 92, Kjærulff 92]). Typischerweise repräsentiert eine *Zeitscheibe* einen Schnappschuss des temporalen Prozesses. Die Wahrscheinlichkeiten, die die Abhängigkeiten zwischen den Zeitscheiben beschreiben, werden auch als *Zustandsentwicklungsmodell* bezeichnet.

Im allgemeinen bleibt die Struktur des Teilnetzes in einer Zeitscheibe und die quantitativen Abhängigkeiten zwischen den Knoten dieser Teilnetze gleich. In [Dagum et al. 92] wird darauf hingewiesen, dass jedoch in einigen Domänen nicht modellierte externe Umstände zu einem Verfall des Modells führen können und eine Aktualisierung der bedingten Wahrscheinlichkeiten oder sogar der Struktur des Netzes verlangen. Ähnlich wird in [Tawfik & Neufeld 94] auf Probleme hingewiesen, die aufgrund von zeitlichen Abläufen entstehen können. So kann beispielsweise die gleiche Evidenz zu verschiedenen Zeitpunkten verschiedene Bedeutungen haben. In diesem Fall ist es notwendig, die bedingten Wahrscheinlichkeiten oder die Struktur der Teilnetze in den entsprechenden Zeitscheiben neu zu definieren.

In einigen Fällen sind die bedingten Wahrscheinlichkeiten sogar zeitabhängig (vergleiche [Tawfik & Neufeld 94]). In [Horvitz & Barry 95] wird ein Beispiel in der Domäne der Weltraumfähre der NASA beschrieben. Wenn es einen Fehler in der Treibstoffversorgung gibt, während das Triebwerk feuert, so ist die Wahrscheinlichkeit, dass die Maschine beschädigt wird, um so höher, je länger das Triebwerk feuert. Gerade bei Krankheiten kann die Wahrscheinlichkeit des Auf-

[1] BAT ist das Akronym für **B**ayesian **A**utomated **T**axi.

tretens von Symptomen stark mit dem Fortschreiten der Krankheit schwanken. Zur Behandlung dieses Falls kann man die bedingten Wahrscheinlichkeiten zeitabhängig definieren, beispielsweise indem man sie durch eine Funktion ausdrückt, die von der Zeit abhängt.

In Kapitel 6 wird man sehen, wie man mit dem in dieser Arbeit vorgestellten Verfahren die bedingten Wahrscheinlichkeiten durch Funktionen parametrisieren und somit auch zeitabhängig definieren kann.

Ein dynamisches Bayessches Netz kann durch *Aufrollen* (Rollup) vereinfacht werden. Einige Rollup-Verfahren werden dazu in Kapitel 3.3 vorgestellt. Durch den Rollup kann das Netz so vereinfacht werden, dass höchstens zwei Zeitscheiben gleichzeitig existieren. Auf diese Weise kann die Vorhersage und Interpretation beliebig vieler Beobachtungen erfolgen.

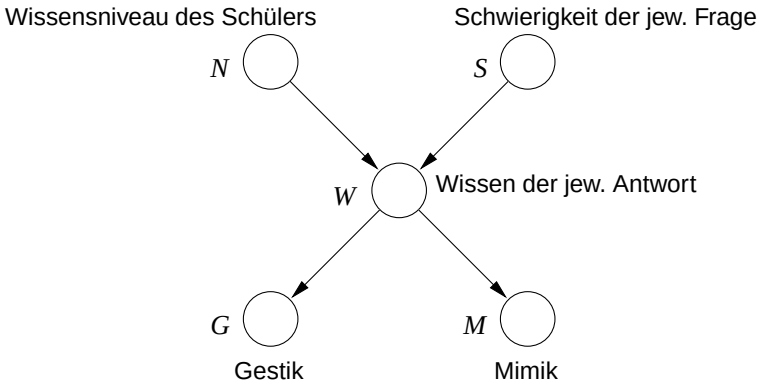
## 3.1 Motivation

Bayessche Netze sind durch ihre statische Natur in ihrem Einsatz stark eingeschränkt und für gewisse Anwendungen nicht sinnvoll. Zur Veranschaulichung wollen wir das folgende Beispiel betrachten, das aus [Kipper et al. 95] entnommen wurde.

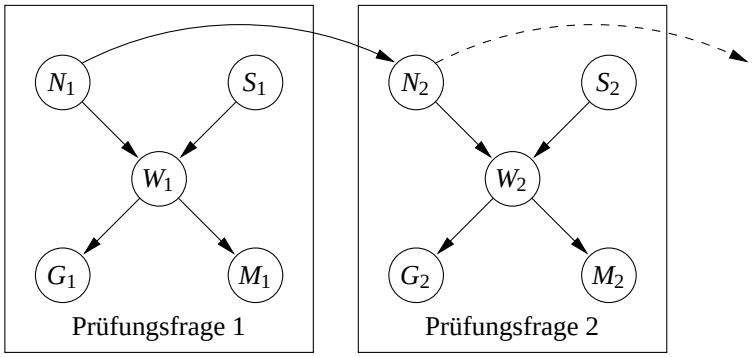
***Beispiel 3.1.1 (Prüfungssituation)***
Dieses Beispiel, das durch die Überlegungen in [Jameson 90] angeregt wurde, ist durch eine Prüfungssituation gegeben. Dem Prüfling wird eine Frage gestellt, die er je nach seinem Wissensniveau und der Schwierigkeit der Frage beantworten kann oder nicht. Je höher das Wissensniveau des Prüflings ist bzw. je leichter die Frage ist, die ihm gestellt wurde, desto größer ist die Wahrscheinlichkeit, dass er die Antwort auf die Frage kennt. Ob er die Antwort auf die Frage kennt, lässt sich anhand seiner Gestik oder Mimik erahnen. Kennt er die Antwort, so wird seine Gestik viel eher ruhig und seine Mimik viel eher gelassen sein, als wenn er sie nicht kennen würde. Die Strukur der soeben angegebenen Sachzusammenhänge, der sogenannte qualitative Teil, spiegelt sich in Abbildung 3.1(a) wider, wobei sich der quantitative Teil, d. h. die A-priori-Wahrscheinlichkeiten und die bedingten Wahrscheinlichkeiten, in Tabelle 3.1 ablesen lässt. Wie wir an den Wahrscheinlichkeiten ablesen können, gehen wir in diesem Beispiel von einem besseren Schüler aus, der geprüft wird. Um einen schlechteren Schüler zu modellieren, müssten die bedingten Wahrscheinlichkeiten entsprechend angepasst werden.

Mit diesem Bayesschen Netz lässt sich nun das Wissensniveau eines Prüflings bezüglich *einer* Frage einschätzen.

Werden dem Prüfling allerdings mehrere Prüfungsfragen gestellt, um das Wissensniveau des Prüflings differenzierter einzuschätzen, so müsste für jede Prü-




(a) statisches Bayessches Netz




(b) dynamisches Bayessches Netz

Abbildung 3.1: Beispielnetze Prüfungsfrage

fungsfrage die entsprechende Netzstruktur vorhanden sein, d. h. das vorhandene Bayessche Netz muss um Knoten erweitert werden. Wird dies nicht getan, so stellt sich die Situation wie folgt dar: Um das Wissensniveau des Prüflings einschätzen zu können, werden ihm nacheinander mehrere Prüfungsfragen gestellt, deren Schwierigkeitsgrade bekannt sind. Die Antwort des Prüflings wird abgewartet, so dass sie als richtig oder falsch eingestuft werden kann. Im selben Bayesschen Netz werden nun sukzessive die Hypothesen der Knoten *S* (Schwierigkeit der jew. Frage) und *W* (Wissen der jew. Antwort) instantiiert, die den eingetroffenen Ereignissen entsprechen.

Wie man Tabelle 3.2 entnehmen kann, wird der Verlauf der Prüfungssituation bei der Einschätzung des Wissensniveaus des Prüflings nicht berücksichtigt. Selbst mehrere falsche Antworten auf leichte Fragen belassen die Einschätzung des Wissensniveaus des Prüflings unverändert.

Wie schon erwähnt ist für solche Domänen ein statisches Modell nicht ausreichend oder sinnvoll. Vielmehr wird ein Modell benötigt, das eine Historie berück-

| | | |
|---|---|---|
| Wissensniveau des Schülers ($N$ : $n_1 =$ hoch, $n_2 =$ niedrig) | | |
| statisches Bayessches Netz | | |
| $\Pr(n)$: | $\Pr(n_1) = 0.8$ | $\Pr(n_2) = 0.2$ |
| dynamisches Bayessches Netz für Zeitscheibe 1 | | |
| $\Pr(n_1)$: | $\Pr(n_{1,1}) = 0.8$ | $\Pr(n_{1,2}) = 0.2$ |
| dynamisches Bayessches Netz für Zeitscheibe $t$ ($t > 1$) | | |
| $\Pr(n_t \mid n_{t-1})$: | $\Pr(n_{t,1} \mid n_{t-1,1}) = 1$ | $\Pr(n_{t,2} \mid n_{t-1,1}) = 0$ |
| | $\Pr(n_{t,1} \mid n_{t-1,2}) = 0$ | $\Pr(n_{t,2} \mid n_{t-1,2}) = 1$ |
| Schwierigkeit der jew. Frage ($S$ : $s_1 =$ schwer, $s_2 =$ mittel, $s_3 =$ leicht) | | |
| $\Pr(s)$: | $\Pr(s_1) = 0.6$ | $\Pr(s_2) = 0.3$ $\Pr(s_3) = 0.1$ |
| Wissen der jew. Antwort ($W$ : $w_1 =$ richtig, $w_2 =$ falsch) | | |
| $\Pr(w \mid n,s)$: | $\Pr(w_1 \mid n_1,s_1) = 0.5$ | $\Pr(w_2 \mid n_1,s_1) = 0.5$ |
| | $\Pr(w_1 \mid n_1,s_2) = 0.6$ | $\Pr(w_2 \mid n_1,s_2) = 0.4$ |
| | $\Pr(w_1 \mid n_1,s_3) = 0.8$ | $\Pr(w_2 \mid n_1,s_3) = 0.2$ |
| | $\Pr(w_1 \mid n_2,s_1) = 0.3$ | $\Pr(w_2 \mid n_2,s_1) = 0.7$ |
| | $\Pr(w_1 \mid n_2,s_2) = 0.4$ | $\Pr(w_2 \mid n_2,s_2) = 0.6$ |
| | $\Pr(w_1 \mid n_2,s_3) = 0.5$ | $\Pr(w_2 \mid n_2,s_3) = 0.5$ |
| Gestik ($G$ : $g_1 =$ hektisch, $g_2 =$ ruhig) | | |
| $\Pr(g \mid w)$: | $\Pr(g_1 \mid w_1) = 0.3$ | $\Pr(g_2 \mid w_1) = 0.7$ |
| | $\Pr(g_1 \mid w_2) = 0.9$ | $\Pr(g_2 \mid w_2) = 0.1$ |
| Mimik ($M$ : $m_1 =$ entsetzt, $m_2 =$ gelassen) | | |
| $\Pr(m \mid w)$: | $\Pr(m_1 \mid w_1) = 0.2$ | $\Pr(m_2 \mid w_1) = 0.8$ |
| | $\Pr(m_1 \mid w_2) = 0.6$ | $\Pr(m_2 \mid w_2) = 0.4$ |

Tabelle 3.1: Bedingte Wahrscheinlichkeiten bzw. A-priori-Wahrscheinlichkeiten für die Knoten der Beispielnetze in Abbildung 3.1

sichtigt. Das bestehende Bayessche Netz muss also bei Bedarf um notwendige Knoten bzw. Netzstrukturen erweitert werden können.

Das vorhergehende Bayessche Netz wird nun so erweitert, dass frühere Prüfungsfragen mit berücksichtigt werden. Es wird für jede Prüfungsfrage ein Bayessches Netz angehängt, wobei die Einschätzung über das Wissensniveau des Prüflings an die nächste Instanz weitergereicht wird. Dazu sind die Knoten, die die Zufallsvariable Wissensniveau des Schülers repräsentieren, über eine gerichtete Kante von der alten zur neuen Zeitscheibe miteinander verbunden.

Die A-priori- (für Zeitscheibe 1) und bedingten Wahrscheinlichkeiten (für Zeitscheibe $t$, $t > 1$) für die Zufallsvariable Wissensniveau des Schülers sind wiederum aus Tabelle 3.1 zu entnehmen. Die bedingten Wahrscheinlichkeiten der

anderen Zufallsvariablen bleiben unverändert.

| *Zeit-scheibe* | *Prüfungsfrage* | | *Einschätzung d. Wissensniveaus* | |
|---|---|---|---|---|
| | *Schwierigk.* | *Antwort* | *statisch* | *dynamisch* |
| 1 | - | - | $(0.800, 0.200)$ | $(0.800, 0.200)$ |
| 1 | leicht | falsch | $(0.615, 0.385)$ | $(0.615, 0.385)$ |
| 2 | leicht | falsch | $(0.615, 0.385)$ | $(0.390, 0.610)$ |
| 3 | leicht | falsch | $(0.615, 0.385)$ | $(0.204, 0.796)$ |
| 4 | schwer | richtig | $(0.870, 0.130)$ | $(0.299, 0.701)$ |
| 5 | mittel | richtig | $(0.857, 0.143)$ | $(0.390, 0.610)$ |
| 6 | schwer | richtig | $(0.870, 0.130)$ | $(0.516, 0.484)$ |
| 7 | schwer | falsch | $(0.741, 0.259)$ | $(0.432, 0.568)$ |
| 8 | leicht | richtig | $(0.865, 0.135)$ | $(0.549, 0.451)$ |
| 9 | leicht | richtig | $(0.865, 0.135)$ | $(0.661, 0.339)$ |
| 10 | leicht | richtig | $(0.865, 0.135)$ | $(0.757, 0.243)$ |

Tabelle 3.2: Vergleich der Einschätzungen des Wissensniveaus durch das Beispielnetz Prüfungsfrage statisch mit den Einschätzungen des Wissensniveaus durch das Beispielnetz Prüfungsfrage dynamisch

Mit jeder Prüfungsfrage wächst das Bayessche Netz um eine sogenannte *Zeitscheibe*. Knoten wie der Knoten der Zufallsvariablen Wissensniveau des Schülers bezeichnet man auch als dynamische Knoten. Treten in einer Zeitscheibe mehrere solcher dynamischen Knoten auf, so ist der Rollup nicht so einfach durchzuführen. Diese Problematik wird in Kapitel 3.3 beschrieben.

Der Vergleich von Tabelle 3.2 mit Tabelle 3.2 in Tabelle 3.2 zeigt, wie wichtig es ist, den Verlauf der Prüfungssituation bei der Einschätzung des Wissensniveaus des Prüflings mit zu berücksichtigen. Der BEL-Wert für die Zufallsvariable Wissensniveau des Schülers im statischen Bayesschen Netz unterscheidet sich im Verlauf der Prüfungssituation ganz erheblich vom entsprechenden BEL-Wert für die Zufallsvariable Wissensniveau des Schülers im dynamischen Bayesschen Netz.

**Bsp**

Wie wir sehen können, verändern sich Dynamische Bayessche Netze also über die Zeit. Bei Bedarf werden neue Zeitscheiben angehängt und alte Zeitscheiben abgeschnitten. Im nachfolgenden Abschnitt werden dazu die notwendigen Begriffe eingeführt und die genaue Vorgehensweise beim Anhängen und Abschneiden von Zeitscheiben erläutert.

## 3.2 Struktur dynamischer Bayesscher Netze

Ein dynamisches Bayessches Netz besteht aus einer variablen endlichen Anzahl $n$ von Zeitscheiben (siehe Abbildung 3.2). Sei $\mathscr{G} = (\mathbb{V}, \mathbb{E})$ der DAG (*gerichtete azyklische Graph*), der die Struktur des dynamischen Modells zu einem bestimmten Zeitpunkt beschreibt. Wenn $t'$ die erste Zeitscheibe des DAGs ist, dann besteht $\mathbb{V}$ aus den disjunkten Teilmengen $\mathbb{V}(t'), \ldots, \mathbb{V}(t'+n-1)$, d. h.

$$\mathbb{V} = \mathbb{V}(t', n) = \bigcup_{t=t'}^{t'+n-1} \mathbb{V}(t).$$

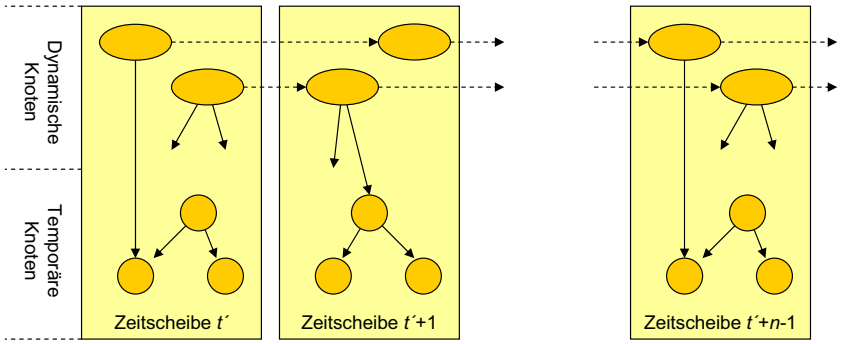



Abbildung 3.2: Prototypische Darstellung eines dynamischen Bayesschen Netzes.

Als Vorlage für die Instantiierung von Zeitscheiben dienen Bayessche Netze mit speziellen Zusatzinformationen, die auch als *Zeitscheibenschemata* bezeichnet werden. Einige Knoten eines Zeitscheibenschemas übernehmen dabei eine besondere Rolle. Sie sind von Zufallsvariablen abhängig, die mit Knoten in der vorhergehenden Zeitscheibe assoziiert sind. Diese Knoten werden als *dynamische Knoten* und die anderen Knoten des Netzes werden als *temporäre Knoten* bezeichnet (siehe Abbildung 3.2). Die Abhängigkeiten werden durch Kanten von Zeitscheibe $t-1$ zu Zeitscheibe $t$ modelliert, sogenannten *temporale Kanten* oder auch *relationale Kanten*. In dynamischen Bayesschen Netzen sind aber keine Kanten erlaubt, die über eine Zeitscheibe gehen. (In Kapitel 5 werden Abhängigkeiten über mehrere Zeitscheiben hinweg und andere Besonderheiten vorgestellt.) Jede neue Zeitscheibe wird also durch temporale Kanten mit der letzten Zeitscheibe des alten Netzes verbunden.

Ein dynamisches Bayessches Netz ist somit eine endliche Aneinanderreihung von Zeitscheibenschemata, die als Zeitscheiben instantiiert wurden (siehe Abbildung 3.3). In Abbildung 3.3 sind die Zeitscheibenschemata, die als Zeitscheiben für das dynamische Bayessche Netz instantiiert werden können, mit $\mathscr{S}_1, \ldots, \mathscr{S}_s$

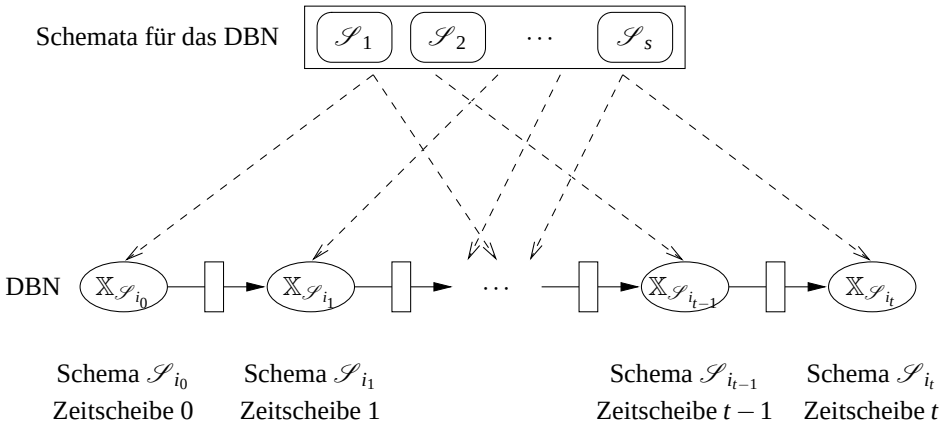



Abbildung 3.3: Die Zeitscheiben sind Instanzen der Schemata für das dynamische Bayessche Netz

bezeichnet. Mit $\mathbb{X}_{\mathscr{S}_{i_\ell}}$ sind die Mengen der Zufallsvariablen bezeichnet, aus denen sich das dynamische Bayessche Netz zusammensetzt. Die Zufallsvariablen für das Schema $\mathscr{S}_i$ ($1 \leq i \leq s$) werden durch $\mathbb{X}_{\mathscr{S}_i}$ symbolisiert. Als Zeitscheibe $k$ ($0 \leq k \leq t$) ist also somit das Schema $\mathscr{S}_{i_k}$ ($1 \leq i_k \leq s$) mit der Menge der Zufallsvariablen $\mathbb{X}_{\mathscr{S}_{i_k}}$ instantiiert.

Durch rechteckige Kästen zwischen zwei jeweils benachbarten Zeitscheiben sind in Abbildung 3.3 die sogenannten *Zustandsentwicklungsmodelle* dargestellt. Ein Zustandsentwicklungsmodell beschreibt die Abhängigkeiten der Zustandsvariablen einer Zeitscheibe $k$ von den Zustandsvariablen der Vorgängerzeitscheibe $k-1$ ($1 \leq k \leq t$). Dieses Zustandsentwicklungsmodell wird im Falle des Rollups in Kapitel 3.3 noch eine große Rolle spielen, und hier wird dann auch seine genaue Bedeutung klar werden. Analog zur Wahrscheinlichkeitsverteilung aller Zufallsvariablen eines Bayesschen Netzes lässt sich die Wahrscheinlichkeitsverteilung für dynamische Bayessche Netze als Produkt ihrer bedingten Wahrscheinlichkeiten schreiben:

$$\Pr(\mathbb{X}_{\mathscr{S}_{i_0}}, \ldots, \mathbb{X}_{\mathscr{S}_{i_t}}) = \prod_{V \in \bigcup_{k=0}^{t} \mathbb{X}_{\mathscr{S}_{i_k}}} \Pr(V \mid \mathbf{pa}(V)).$$

Ist $V$ ein Wurzelknoten, so reduzieren sich die bedingten Wahrscheinlichkeiten zu $\Pr(V)$.

In Abbildung 3.4 ist ein dynamisches Bayessches Netz dargestellt, das sich aus den vier verschiedenen Schemata $\mathscr{A}$, $\mathscr{B}$, $\mathscr{C}$ und $\mathscr{D}$ aufbaut. Es sind dabei fünf Zeitscheiben (Zeitscheibe 0 bis Zeitscheibe 4) instantiiert. Wie in der Abbildung zu sehen ist, sind im Schema $\mathscr{B}$ Knoten enthalten, die Elternknoten im Schema $\mathscr{A}$ und Elternknoten im Schema $\mathscr{D}$ haben. Diese Elternknoten der beiden Schemata

müssen dieselben Namen besitzen oder über denselben *Alias* referenziert werden können. Ein Knoten in einem dynamischen Bayesschen Netz kann also neben seinem *eindeutigen* Namen, den kein weiterer Knoten des dynamischen Bayesschen Netzes besitzt, noch mit mehreren anderen Namen (*Aliase*) benannt sein, die er sich mit anderen Knoten des dynamischen Bayesschen Netzes teilt. Zwei Knoten eines Schemas haben aber nie denselben Alias. Ein Alias kommt also in einem Schema höchstens einmal vor.

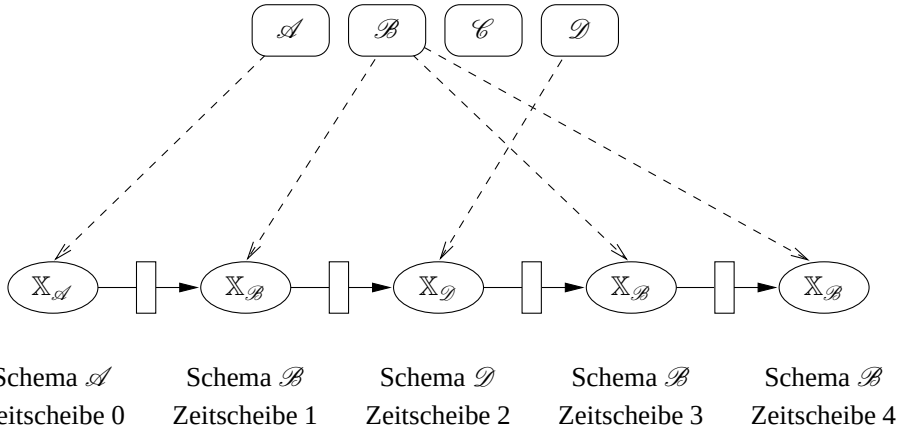



Abbildung 3.4: Fünf Zeitscheiben eines dynamischen Bayesschen Netzes mit vier Schemata.

Ein Schema hat keine Elternknoten in vorhergehenden Zeitscheiben, wenn es als Zeitscheibe 0 instantiiert wird. Würde das Schema $\mathscr{A}$ zum Zeitpunkt $t \neq 0$ als Zeitscheibe $t$ instantiiert werden, so hätten einige Knoten der Zeitscheibe $t$ Elternknoten in Zeitscheibe $t-1$. Diese Knoten bezeichnet man als *dynamische Knoten*. Die gerichteten Kanten zwischen den Zeitscheiben werden *temporale Kanten* genannt. Die Menge der temporalen Kanten der Zeitscheibe $t$ wird mit $\mathbb{E}^{tmp}(t) = \{(A,B) \in \mathbb{E} \mid A \in \mathbb{V}(t-1), B \in \mathbb{V}(t)\}, t' \leq t \leq t'+n-1$ bezeichnet. In Abbildung 3.2 sind die *temporalen Kanten* gestrichelt eingezeichnet.

Mit den temporalen Kanten sind bedingte Wahrscheinlichkeiten verbunden, die beschreiben, wie sich die Wahrscheinlichkeitsverteilung der Zufallsvariablen in Zeitscheibe $t-1$ auf die Wahrscheinlichkeitsverteilung der Zufallsvariablen in Zeitscheibe $t$ auswirkt.

Implizit wird vorausgesetzt, dass die Zeitscheiben des dynamischen Bayesschen Netzes die ***Markov-Eigenschaft*** einhalten, d. h.

$$\mathbb{X}_{\mathbb{V}(0)}, \ldots, \mathbb{X}_{\mathbb{V}(t-1)} \perp\!\!\!\perp \mathbb{X}_{\mathbb{V}(t+1)}, \ldots, \mathbb{X}_{\mathbb{V}(t+k)} \mid \mathbb{X}_{\mathbb{V}(t)}, \qquad \forall t, k > 0.$$

Dies bedeutet, dass eine Zufallsvariable $X_K$, die einem Knoten $K \in \mathbb{V}(\ell)$ zugeordnet ist, von keiner Zufallsvariablen $X_A$ direkt beeinflusst wird, die mit einem

beliebigen Knoten $A \in \mathbb{V}(i)$ assoziiert ist ($i \leq \ell - 2$). Graphentheoretisch bedeutet dies, dass nur Kanten erster Ordnung erlaubt sind. Es sind keine Kanten erlaubt, die zwei nicht benachbarte Zeitscheiben miteinander verbinden. Informell bedeutet dies, dass die Zukunft unabhängig von der Vergangenheit ist, wenn die Gegenwart bekannt ist.

Im folgenden Abschnitt zeigen wir, wie sich durch Aufspalten oder Löschen von Knoten im Bayesschen Netz die BEL-Werte der anderen Knoten im Bayesschen Netz ändern, so dass sich ein einfaches Abschneiden alter Zeitscheiben im dynamischen Bayesschen Netz nicht als Rollup eignet. Weiterhin zeigen wir, wie sich Zeitscheiben effektiv anhängen lassen, so dass sich ein Teil der vorhergehenden Berechnungen wiederverwenden lässt, was uns dann direkt zum Rollup alter Zeitscheiben ohne Informationsverlust führt.

## 3.3 Rollup für dynamische Bayessche Netze

Durch das Anhängen von Zeitscheiben wächst bei dynamischen Bayesschen Netzen der Bedarf an *Ressourcen* wie Rechenzeit und Arbeitsspeicher. Zu gewissen Zeitpunkten wird es zwingend erforderlich, dynamische Bayessche Netz an die vorhandene Rechenzeit und/oder den vorhandenen Arbeitsspeicher anzupassen. Diese Anpassung kann durch ein Rollup-Verfahren geschehen. Im allgemeinen werden dabei ganze Zeitscheiben des dynamischen Bayesschen Netzes aufgerollt[2] und maximal zwei Zeitscheiben des dynamischen Bayesschen Netzes verwaltet.

Im folgenden werden für die verschiedensten Bedürfnisse Rollup-Verfahren vorgestellt, die die Inferenzverfahren nutzen, die in dieser Arbeit vorgestellt werden (siehe Kapitel 2). Durch diese Rollup-Verfahren werden dann Vorhersagen und Interpretationen beliebig vieler Beobachtungen ermöglicht.

Wie man sehen wird, lässt sich der Rollup von dynamischen Bayesschen Netzen nicht so einfach durch das Abschneiden von Knoten (oder ganzen Zeitscheiben) realisieren. Der hierdurch verursachte Fehler könnte so groß sein oder durch Akkumulation (beim Abschneiden mehrerer Zeitscheiben nacheinander) so groß werden, dass die durch die Propagierung gewonnenen Einschätzungen unbrauchbar sind.

### 3.3.1 Löschen von Netzstrukturen

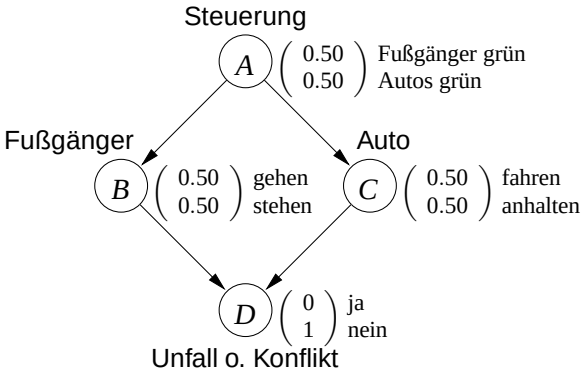
In diesem Abschnitt wird gezeigt, wie sich schon in einem kleinen Bayesschen Netz die BEL-Werte der Knoten ändern, wenn eine Netzstruktur aufgebrochen

[2] Eine aufgerollte Zeitscheibe kann man auch als *abgeschlossen* bezeichnen. Dieser Ausdruck verdeutlicht sehr anschaulich, dass keine Einschätzungen für die Knoten dieser Zeitscheibe mehr vorgenommen werden können.

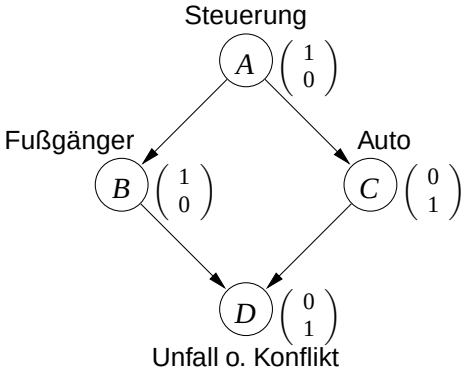
oder ein Knoten des Bayesschen Netzes gelöscht wird.

Das Abschneiden eines Knotens $K$ in einem Bayesschen Netz kann als Aufspalten des Netzes an der Stelle des Knotens $K$ mit anschließendem Löschen des Knotens $K$ betrachtet werden. Im folgenden Beispiel wird man sehen, wie schon ein einfaches Aufspalten der Netzstruktur eines Bayesschen Netzes die BEL-Werte der Knoten des Bayesschen Netzes verändert.

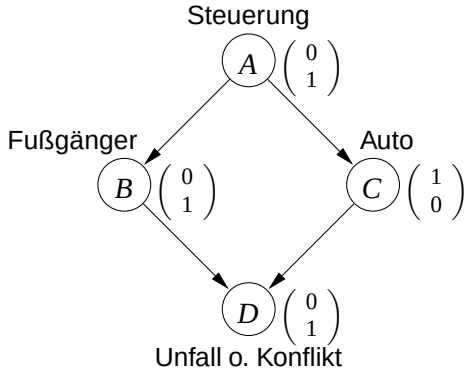
***Beispiel 3.3.1 (Aufspalten bzw. Löschen eines Knotens)***
Sei der Graph aus Abbildung 3.5 mit den A-priori- und bedingten Wahrscheinlichkeiten in Tabelle 3.3 gegeben.




(a) Anfangswahrscheinlichkeiten




(b) BEL-Werte bei der Instantiierung des Knotens $A$ mit $a_1$




(c) BEL-Werte bei der Instantiierung des Knotens $A$ mit $a_2$

Abbildung 3.5: Beispielnetz Ampelsteuerung [$\Pr(A)$ $\Pr(B \mid A)$ $\Pr(C \mid A)$ $\Pr(D \mid B,C)$]

Es ist ein einfaches Bayessches Netz zur Darstellung des Sachverhaltes “Ampelsteuerung” an einer einfachen Straße zur Straßenüberquerung. Eine zentrale Steuerung gibt an, ob die Autos fahren oder die Fußgänger die Straße überque-

| | | |
|---|---|---|
| Steuerung ($A$ : $a_1$ = Fußgänger grün, $a_2$ = Autos grün) | | |
| $\Pr(a)$: | $\Pr(a_1) = 0.50$ | $\Pr(a_2) = 0.50$ |
| Fußgänger ($B$ : $b_1$ = gehen, $b_2$ = stehen) | | |
| $\Pr(b \mid a)$: | $\Pr(b_1 \mid a_1) = 1.00$ | $\Pr(b_2 \mid a_1) = 0.00$ |
| | $\Pr(b_1 \mid a_2) = 0.00$ | $\Pr(b_2 \mid a_2) = 1.00$ |
| Auto ($C$ : $c_1$ = fahren, $c_2$ = anhalten) | | |
| $\Pr(c \mid a)$: | $\Pr(c_1 \mid a_1) = 0.00$ | $\Pr(c_2 \mid a_1) = 1.00$ |
| | $\Pr(c_1 \mid a_2) = 1.00$ | $\Pr(c_2 \mid a_2) = 0.00$ |
| Unfall o. Konflikt ($D$ : $d_1$ = ja, $d_2$ = nein) | | |
| $\Pr(d \mid b,c)$: | $\Pr(d_1 \mid b_1,c_1) = 1.00$ | $\Pr(d_2 \mid b_1,c_1) = 0.00$ |
| | $\Pr(d_1 \mid b_1,c_2) = 0.00$ | $\Pr(d_2 \mid b_1,c_2) = 1.00$ |
| | $\Pr(d_1 \mid b_2,c_1) = 0.00$ | $\Pr(d_2 \mid b_2,c_1) = 1.00$ |
| | $\Pr(d_1 \mid b_2,c_2) = 1.00$ | $\Pr(d_2 \mid b_2,c_2) = 0.00$ |

Tabelle 3.3: Bedingte Wahrscheinlichkeiten bzw. A-priori-Wahrscheinlichkeiten für die Knoten des Bayesschen Netzes Ampelsteuerung (siehe Abbildung 3.5)

ren dürfen. Dürfen die Autos fahren, so zeigt die Steuerung für die Autos grün und die Fußgänger rot an. Geht man davon aus, dass sich alle Verkehrsteilnehmer an die Straßenverkehrsordnung halten, so ist die Wahrscheinlichkeit für einen Unfall gleich Null, selbst wenn man nicht weiß, ob die Ampel jetzt für die Fußgänger oder die Autos grün anzeigt (siehe Abbildung 3.5(a)). Für die verschiedenen Instantiierungen des Knotens $A$ sind die BEL-Werte der anderen Knoten in den Abbildungen 3.5(a), (b) und (c) dargestellt.

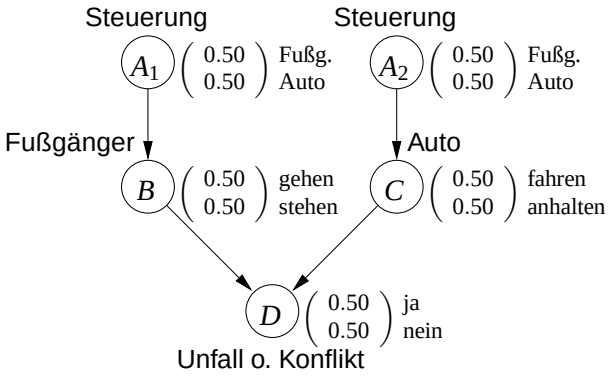
Jetzt wird das obige Beispiel leicht abgeändert (siehe Abbildung 3.6).

Anstatt einer zentralen Steuerung sind zwei separate Steuerungen (Knoten $A_1$ und Knoten $A_2$) modelliert. Die A-priori-Wahrscheinlichkeiten der neuen Knoten sind in Tabelle 3.4 gegeben. Die Wahrscheinlichkeiten der anderen Knoten bleiben erhalten. Man erhält ein einfaches Bayessches Netz zur Darstellung des

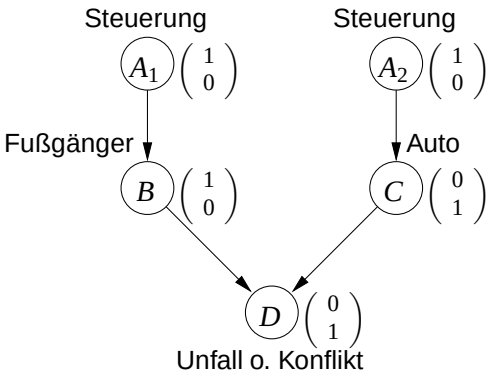
| | | |
|---|---|---|
| Steuerung ($A_1, A_2$ : $a_1$ = Fußgänger, $a_2$ = Auto) | | |
| $\Pr(a)$: | $\Pr(a_1) = 0.50$ | $\Pr(a_2) = 0.50$ |

Tabelle 3.4: A-priori-Wahrscheinlichkeiten für die beiden Knoten $A_1$ und $A_2$ des aufgespaltenen Knotens $A$ des Bayesschen Netzes Zebrastreifen (siehe Abbildung 3.6)

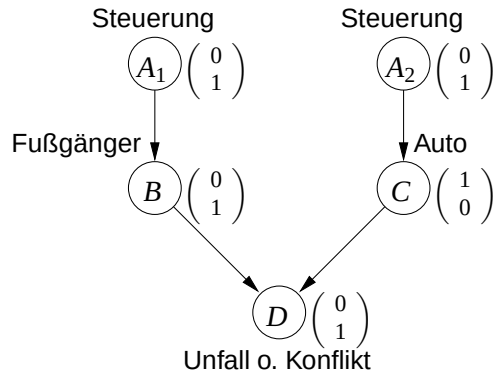
Sachverhaltes “Zebrastreifen” an einer einfachen Straße zur Straßenüberquerung. Die beiden Verkehrsteilnehmer übernehmen die Ampelsteuerung. Signalisiert ein




(a) Anfangswahrscheinlichkeiten




(b) BEL-Werte bei der Instantiierung des Knotens $A$ mit $a_1$



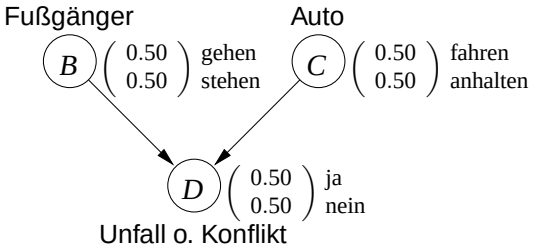

(c) BEL-Werte bei der Instantiierung des Knotens $A$ mit $a_2$

Abbildung 3.6: Beispielnetz Zebrastreifen $[\Pr(A_1)\ \Pr(A_2)\ \Pr(B \mid A_1)\ \Pr(C \mid A_2)\ \Pr(D \mid B,C)]$

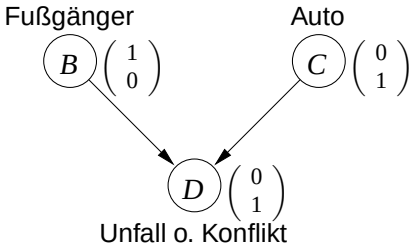
Fußgänger, dass er die Straße überqueren möchte, so wird der Autofahrer am Zebrastreifen halten und ihn auf die andere Straßenseite lassen. Geht man davon aus, dass sich alle Verkehrsteilnehmer an die Straßenverkehrsordnung halten, so ist die Wahrscheinlichkeit für einen Unfall gleich Null, wenn sich die Verkehrsteilnehmer miteinander (z. B. durch Handzeichen) verständigt haben (siehe Abbildungen 3.6(b) und (c)). Hat diese Kommunikation versagt (z. B. tritt der Fußgänger unerwartet für den Autofahrer an den Zebrastreifen heran und überquert die Straße, ohne auf die Autos zu achten.), so steigt die Wahrscheinlichkeit in diesem Beispiel auf 50% (siehe Abbildung 3.6(a)). Man kann in diesem Beispiel davon ausgehen, dass die beiden Knoten $A_1$ und $A_2$ immer gleich instantiiert sind (bei Kommunikation beide auf die erste oder beide auf die zweite Hypothese der beiden Knoten oder bei keiner Kommunikation keine von beiden Hypothesen).

Das neue Bayessche Netz Zebrastreifen kann man damit vergleichen, dass der

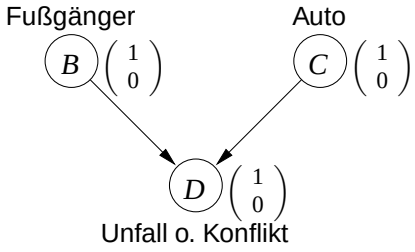
Knoten $A$ des Bayesschen Netzes Ampelsteuerung aufgespalten wird. Für verschiedene Instantiierungen der Knoten $A_1$ und $A_2$ sind die BEL-Werte der anderen Knoten in den Abbildungen 3.6(a), (b) und (c) dargestellt.




(a) Anfangswahrscheinlichkeiten




(b) BEL-Werte bei der Instantiierung des Knotens $B$ mit $b_1$ und des Knotens $C$ mit $c_2$




(c) BEL-Werte bei der Instantiierung des Knotens $B$ mit $b_1$ und des Knotens $C$ mit $c_1$

Abbildung 3.7: Beispielnetz Straßenverkehr $[\Pr(B)\ \Pr(C)\ \Pr(D \mid B,C)]$

Im nächsten Bayesschen Netz Straßenverkehr (siehe Abbildung 3.7) ist der Sachverhalt einer "Straßenüberquerung" mit schlechten Verhältnissen modelliert, so dass es weder eine Ampel oder ein Zebrastreifen noch eine andere Möglichkeit zur Mitteilung an einen Autofahrer gibt (z. B. durch eine starke Kurve oder verminderte Sichtverhältnisse wie Nebel oder Dunkelheit), dass man als Fußgänger die Straße überqueren möchte. Die Knoten $A_1$ und $A_2$ wie im Beispielnetz Zebrastreifen oder der Knoten $A$ wie im Beispielnetz Ampelsteuerung existieren im Bayesschen Netz Straßenverkehr nicht mehr. Die Knoten $B$ und $C$ erhalten A-priori-Wahrscheinlichkeiten (siehe Tabelle 3.5), die Wahrscheinlichkeiten für den Knoten $D$ bleiben aus dem Vorgängernetz erhalten.

Wie man aus den BEL-Werten, die in den Abbildungen 3.5(a), (b) und (c) ablesen kann, ist die Überquerung der Fahrbahn für den Fußgänger ein pures Glücksspiel. Wir der Passant bei der Überquerung der Fahrbahn rechtzeitig von einem ankommenden Fahrzeug bemerkt und kann dieses anhalten, so kommt es zu keinem Unfall (siehe Abbildung 3.5(b)). Reagiert ein Autofahrer jedoch zu spät, so kommt es zu einem Zusammenstoß (siehe Abbildung 3.5(c)).

Das Bayessche Netz Straßenverkehr kann man mit dem Bayesschen Netz Ampelsteuerung damit vergleichen, dass der Knoten $A$ des Bayesschen Netzes Am-

| Fußgänger ($B : b_1 =$ gehen, $b_2 =$ stehen) | | |
|---|---|---|
| $\Pr(b)$: | $\Pr(b_1) = 0.50$ | $\Pr(b_2) = 0.50$ |
| **Auto** ($C : c_1 =$ fahren, $c_2 =$ anhalten) | | |
| $\Pr(c)$: | $\Pr(c_1) = 0.50$ | $\Pr(c_2) = 0.50$ |

Tabelle 3.5: A-priori-Wahrscheinlichkeiten für die beiden Knoten $B$ und $C$ des Bayesschen Netzes Straßenverkehr (siehe Abbildung 3.7)

pelsteuerung gelöscht wird und die beiden Nachfolgerknoten $B$ und $C$, den BEL-Wert des Knotens $A$ nach der Initialisierung des Bayesschen Netzes als A-priori-Wahrscheinlichkeiten eingetragen bekommen.

**Bsp**

Wie man im Beispiel gesehen hat, entsteht beim Aufspalten oder Löschen eines Knotens in einem Bayesschen Netz ein Fehler. Dieser Fehler kann dabei so groß sein, dass die durch Inferenz gewonnenen BEL-Werte der Knoten nur noch bedingt oder gar nicht mehr brauchbar sind.

### 3.3.2 Anhängen von Zeitscheiben

Für das Lösen von dynamischen Bayesschen Netzen werden zwei verschiedene Strategien vorgestellt. Die eine Strategie kann als global angesehen werden, bei der das gesamte dynamische Bayessche Netz beim Lösen als ein Bayessches Netz betrachtet wird, das sich nicht aus Zeitscheiben aufbaut. Bei der zweiten Strategie hingegen soll sich der Aufbau des Bayesschen Netzes aus Zeitscheiben auch in der Struktur des Junction Tree widerspiegeln. Jede der beiden Strategien hat seine Vor- und Nachteile, die jeweils in den einzelnen Abschnitten angesprochen werden.

Nachfolgend werden zwei verschiedene Möglichkeiten für die Bearbeitung von dynamischen Bayesschen Netzen an einem Beispielnetz miteinander verglichen. Bei der ersten Möglichkeit wird das dynamische Bayessche Netz nur als Bayessches Netz betrachtet. Der Aufbau des dynamischen Bayesschen Netzes aus Zeitscheiben spiegelt sich dabei nicht im Junction Tree wider. Bei der zweiten Möglichkeit werden die Zeitscheiben des dynamischen Bayesschen Netzes bei der Evaluierung berücksichtigt. Hier deutet sich schon an, wie ein Rollup ohne Informationsverlust durchgeführt werden kann.

#### 3.3.2.1 Behandlung des dynamischen Bayesschen Netzes als normales Bayessches Netz

Im folgenden wollen wir dynamische Bayessche Netze mit den Verfahren für Bayessche Netze lösen. Zu jedem Zeitpunkt $t$ ist das dynamische Bayessche Netz genau bekannt und kann zu einem Zeitpunkt auch als statisches Bayessches Netz aufgefasst werden. Somit können zur Lösung des dynamischen Bayesschen Netzes beispielsweise die Verfahren ohne Anpassung angewendet werden, die in Abschnitt 2.3 vorgestellt wurden. Dabei wird für das dynamische Bayessche Netz der Junction Tree berechnet, ohne zu berücksichtigen, dass sich das dynamische Bayessche Netz aus Zeitscheiben aufbaut.

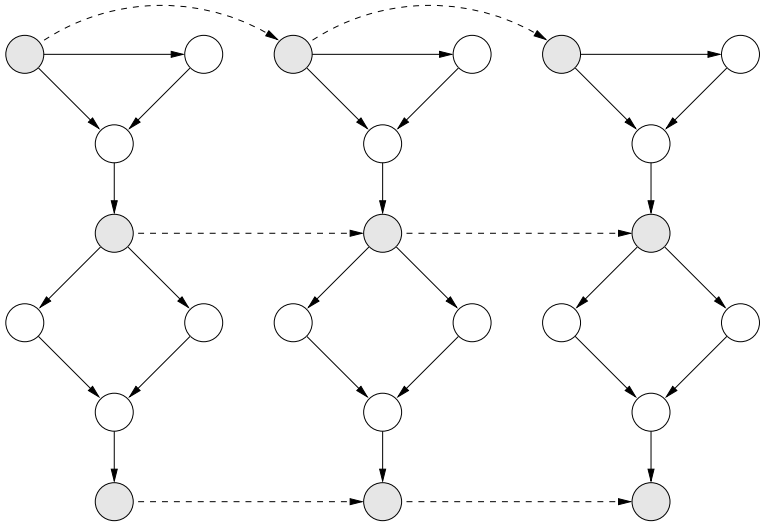


Abbildung 3.8: Temporale Kanten und dynamische Knoten eines dynamischen Bayesschen Netzes.

Schauen wir uns das dynamische Bayessche Netz in Abbildung 3.8 an, das aus drei Zeitscheiben besteht. Die dynamischen Knoten sind grau-schraffiert und die temporalen Kanten sind gestrichelt eingezeichnet. Zuallererst wird das dynamische Bayessche Netz moralisiert. In Abbildung 3.9 sind die *Moralisierungskanten* gestrichelt eingezeichnet. Die grau-schraffierten Knoten bezeichnet man als *Interfaceknoten*. Der Grund für diese Bezeichnung wird klar, wenn bei der Triangulierung des dynamischen Bayesschen Netzes die Zeitscheiben berücksichtigt werden. Dann nämlich bilden die Interfaceknoten das *Zustandsentwicklungsmodell* der Zeitscheibe.

Zunächst wird nun aber die globale Betrachtung vorgestellt, in der die Zeitscheiben des dynamischen Bayesschen Netzes unberücksichtigt bleiben sollen. Die optimale Eliminationsreihenfolge wird über alle Knoten des gesamten dyna-

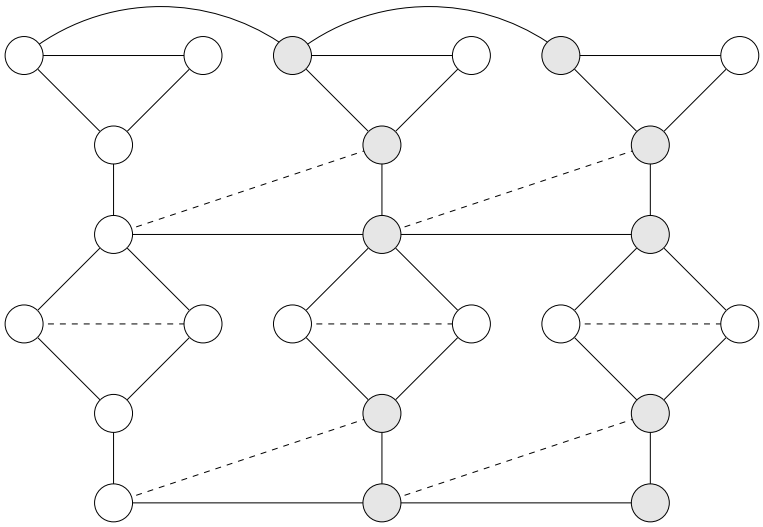


Abbildung 3.9: Moralisierungskanten und Interfaceknoten eines dynamischen Bayesschen Netzes.

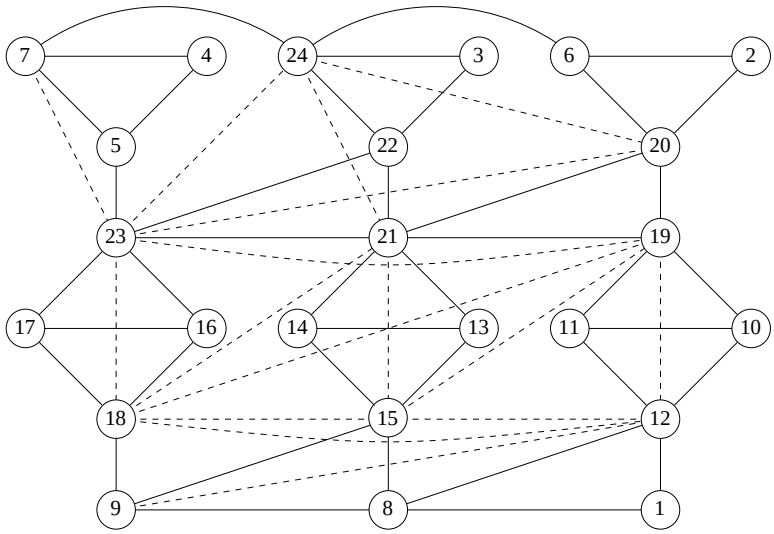



Abbildung 3.10: DBN: Triangulationskanten eines dynamischen Bayesschen Netzes ohne Berücksichtigung von Zeitscheiben. Die Eliminationsreihenfolge ist der Knotennumerierung zu entnehmen.

mischen Bayesschen Netzes bestimmt. In Abbildung 3.10 sind dazu die *Triangulationskanten* einer Eliminationsreihenfolge von Knoten ohne Berücksichtigung der Zeitscheiben gestrichelt eingezeichnet. Man erkennt, dass die Knoten durch die Triangulationskanten über mehrere Zeitscheiben hinweg miteinander verbun-

den werden. Die hieraus resultierenden Cliquen vereinen Knoten aus mehreren Zeitscheiben. Wird nun zusätzlich eine neue Zeitscheibe angefügt, und eine neue optimale Eliminationsreihenfolge über alle Knoten gesucht, so muss der *Junction Tree* im allgemeinen vollständig neu berechnet werden. Dabei können alte Berechnungen nicht wiederverwendet werden. In Abbildung 3.11 ist der aus der Abbildung 3.10 resultierende Junction Tree dargestellt.

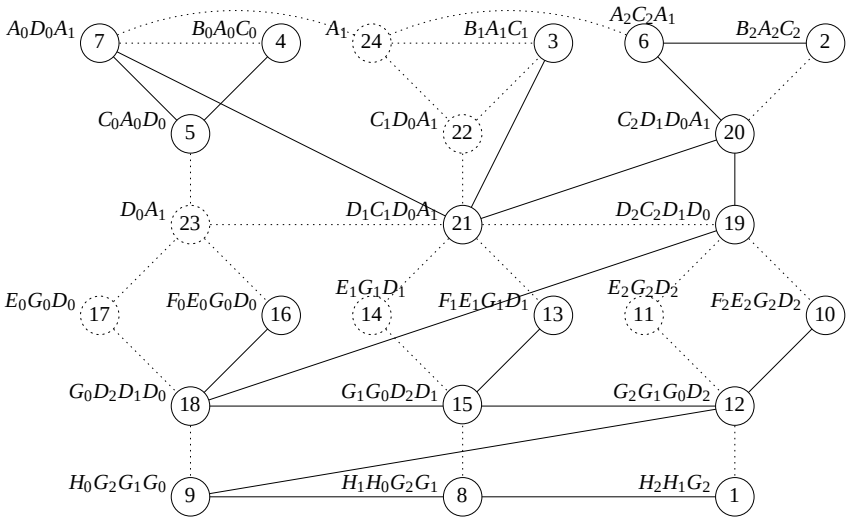



Abbildung 3.11: DBN: Junction Tree eines dynamischen Bayesschen Netzes ohne Berücksichtigung von Zeitscheiben.

Insgesamt müssen

- die Eliminationsreihenfolge,
- der Junction Tree des Bayesschen Netzes, der auf der Eliminationsreihenfolge beruht, und
- die einzelnen Cliquen-Nachrichten, die bei vorhergehenden Inferenzen verschickt wurden,

neu berechnet werden.

In Tabelle 3.6 sind die Eliminationsreihenfolge der Knoten und die dabei induzierten Cluster und Cliquen dargestellt. Die Cluster und Cliquen umfassen Knoten aus mehreren Zeitscheiben. Eine maximale Clique enthält vier Knoten.

#### 3.3.2.2 Strukturerhaltung im Junction Tree des dynamischen Bayesschen Netzes

Jetzt wird das dynamische Bayessche Netz beim Lösen so behandelt, dass sich der Aufbau des dynamischen Bayesschen Netzes aus Zeitscheiben auch in der Struk-

| *eliminierter Knoten* | *induzierter Cluster* | *induzierte Clique* |
|---|---|---|
| $H_2$ | $H_2H_1G_2$ | $Clq_1 = \{H_2, H_1, G_2\}$ |
| $B_2$ | $B_2A_2C_2$ | $Clq_2 = \{B_2, A_2, C_2\}$ |
| $B_1$ | $B_1A_1C_1$ | $Clq_3 = \{B_1, A_1, C_1\}$ |
| $B_0$ | $B_0A_0C_0$ | $Clq_4 = \{B_0, A_0, C_0\}$ |
| $C_0$ | $C_0A_0D_0$ | $Clq_5 = \{C_0, A_0, D_0\}$ |
| $A_2$ | $A_2C_2A_1$ | $Clq_6 = \{A_2, C_2, A_1\}$ |
| $A_0$ | $A_0D_0A_1$ | $Clq_7 = \{A_0, D_0, A_1\}$ |
| $H_1$ | $H_1H_0G_2G_1$ | $Clq_8 = \{H_1, H_0, G_2, G_1\}$ |
| $H_0$ | $H_0G_2G_1G_0$ | $Clq_9 = \{H_0, G_2, G_1, G_0\}$ |
| $F_2$ | $F_2E_2G_2D_2$ | $Clq_{10} = \{F_2, E_2, G_2, D_2\}$ |
| $E_2$ | $E_2G_2D_2$ | - |
| $G_2$ | $G_2G_1G_0D_2$ | $Clq_{11} = \{G_2, G_1, G_0, D_2\}$ |
| $F_1$ | $F_1E_1G_1D_1$ | $Clq_{12} = \{F_1, E_1, G_1, D_1\}$ |
| $E_1$ | $E_1G_1D_1$ | - |
| $G_1$ | $G_1G_0D_2D_1$ | $Clq_{13} = \{G_1, G_0, D_2, D_1\}$ |
| $F_0$ | $F_0E_0G_0D_0$ | $Clq_{14} = \{F_0, E_0, G_0, D_0\}$ |
| $E_0$ | $E_0G_0D_0$ | - |
| $G_0$ | $G_0D_2D_1D_0$ | $Clq_{15} = \{G_0, D_2, D_1, D_0\}$ |
| $D_2$ | $D_2C_2D_1D_0$ | $Clq_{16} = \{D_2, C_2, D_1, D_0\}$ |
| $C_2$ | $C_2D_1D_0A_1$ | $Clq_{17} = \{C_2, D_1, D_0, A_1\}$ |
| $D_1$ | $D_1C_1D_0A_1$ | $Clq_{16} = \{D_1, C_1, D_0, A_1\}$ |
| $C_1$ | $C_1D_0A_1$ | - |
| $D_0$ | $D_0A_1$ | - |
| $A_1$ | $A_1$ | - |

Tabelle 3.6: Eliminierung der Knoten: Zeitscheiben werden nicht berücksichtigt

tur des Junction Trees widerspiegelt. In diesem Zusammenhang wird der Begriff der *eingeschränkten Eliminationsreihenfolge* eingeführt. Werden neue Zeitscheiben angehängt, so wird man sehen, dass beim Lösen des neu entstandenen dynamischen Bayesschen Netzes alte Berechnungen wiederverwendet werden können. Leider ist im allgemeinen die eingeschränkte Eliminationsreihenfolge im Vergleich zu einer Eliminationsreihenfolge, die das gesamte Netz berücksichtigt, schlechter im Triangulationsergebnis. Vor allem die Cliquen, die an der Nahtstelle zwischen den Zeitscheiben entstehen, werden besonders groß. Dies ist jedoch vom entsprechenden Zustandsentwicklungsmodell zwischen den Zeitscheiben abhängig.

Sollen also alte Berechnungen wiederverwendet werden, so muss das Junc-

tion-Tree-Verfahren angepasst werden. Ideal wäre es, wenn man beim Junction Tree neue Strukturen auch so anhängen könnte, wie dies bei den dynamischen Bayesschen Netzen mit den Zeitscheiben möglich ist. Wie man aber gesehen hat, ist dies bei einer beliebigen Eliminationsreihenfolge im allgemeinen nicht machbar. Da man weiß, dass die Cliquen eines Junction Trees des Bayesschen Netzes alleine schon durch die Eliminationsreihenfolge festgelegt sind, wird man die Eliminationsreihenfolge entsprechend anpassen müssen. Durch die Eliminationsreihenfolge werden auch die Triangulationskanten bestimmt. Will man Cliquen, die nur Knoten aus der jeweiligen Zeitscheibe und maximal aus einer benachbarten Zeitscheibe enthalten, so dürfen die Triangulationskanten nur innerhalb einer Zeitscheibe liegen oder maximal zu einer benachbarten Zeitscheibe reichen. Den Zusammenhang zwischen der Eliminationsreihenfolge und den Triangulationskanten (und somit den Cliquen, die entstehen) liefert uns der folgende Satz von Rose, Tarjan und Lueker (entnommen aus [Kjærulff 93]) der allgemeinen Graphentheorie:

***Satz 3.3.2 (Rose, Tarjan und Lueker)***
Sei $\mathcal{G}_{\#} = (\mathbb{V}, \mathbb{E})$ ein Graph, dessen Knoten $V \in \mathbb{V}$ durch die bijektive Funktion $\#\colon \mathbb{V} \to \{1, \ldots, |\mathbb{V}|\}, V \mapsto \#(V)$ mit einer Nummer versehen werden. Ein solcher Graph wird auch als *geordneter Graph* bezeichnet.

Sei also $\mathcal{G}_{\#} = (\mathbb{V}, \mathbb{E})$ ein geordneter Graph, dann wird die Kante $\{U, V\}$ im triangulierten Graphen $\mathcal{G}^t_{\#}$ genau dann und nur dann enthalten sein, wenn es einen Pfad $\langle V, V_1, \ldots, V_k, U \rangle$ im geordneten Graphen $\mathcal{G}_{\#} = (\mathbb{V}, \mathbb{E})$ gibt, so dass $\#(V_i) < \min\{\#(V), \#(U)\}\ \forall i = 1, \ldots, k$. S

Wie schon im Satz erwähnt, ist # eine bijektive Funktion aus der Menge der Knoten $\mathbb{V}$ in die Menge $\{1, \ldots, |\mathbb{V}|\}$, d. h.

$$\#\colon \mathbb{V} \to \mathbb{N} \text{ mit } V \mapsto \#(V) \in \{1, \ldots, |\mathbb{V}|\}.$$

Für einen Knoten $V \in \mathbb{V}$ gibt $\#(V)$ die Position von $V$ in der Eliminationsreihenfolge an. Mit Satz 3.3.2 kann nun zusammen mit einer Eliminationsreihenfolge, die sich auf die Zeitscheiben einschränkt (Die Knoten einer Zeitscheibe $t$ werden vor den Knoten einer Zeitscheibe $t+1$ abgearbeitet und so fort.), gezeigt werden, dass bei der Eliminierung eines Knotens $V$ der Zeitscheibe $t$ maximal Triangulationskanten eingefügt werden, die Knoten der Zeitscheibe $t$ oder die Interfaceknoten der Zeitscheibe $t+1$ miteinander verbinden. Die entsprechenden Aussagen und dazugehörigen Beweise können in [Kjærulff 93] nachgelesen werden.

Dabei ist eine wichtige Schlussfolgerung aus dem Satz, dass bei der Eliminierung eines Knotens nur zwischen nicht eliminierten Knoten Triangulationskanten eingefügt werden.

Dieses wird wie folgt ausgenutzt: Beim Anhängen einer neuen Zeitscheibe an das dynamische Bayessche Netz behält man die Eliminationsreihenfolge bei, in

der die Knoten der alten Zeitscheiben abgearbeitet wurden, und fügt daran die Eliminationsreihenfolge für die Abarbeitung der Knoten der neuen Zeitscheibe an. So werden zwischen den alten Knoten keine neuen Triangulationskanten eingefügt, und die alten Cliquen bleiben zum großen Teil erhalten. Der *Junction Tree* muss nicht komplett neu berechnet werden. An den alten Junction Tree werden die neuen Cliquen angehängt und eventuell alte Cliquen an der Schnittstelle zwischen alten und neuen Cliquen um ein paar Knoten erweitert. Es können alte Berechnungen wiederverwendet werden. In Abbildung 3.12 sind die *Triangulationskanten* einer Eliminationsreihenfolge gestrichelt eingezeichnet, die berücksichtigt, dass das Netz aus Zeitscheiben aufgebaut ist. Man erkennt, wie die Triangulationskan-

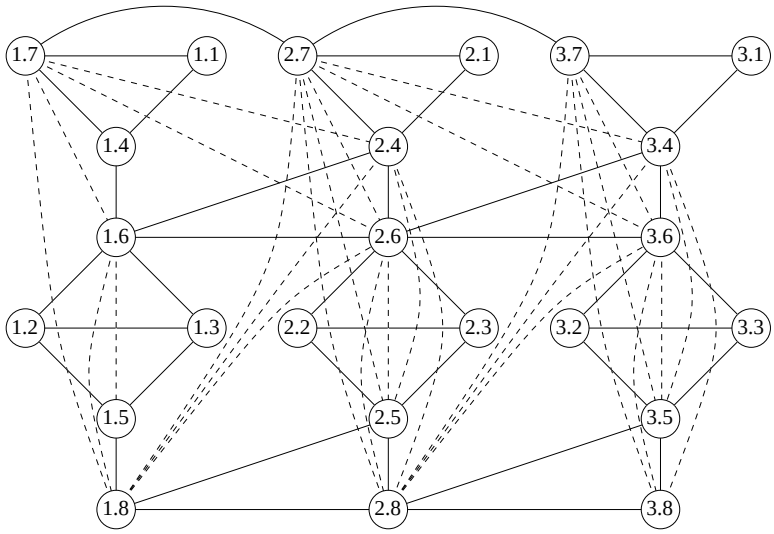



Abbildung 3.12: DBN: Triangulationskanten eines dynamischen Bayesschen Netzes mit Berücksichtigung von Zeitscheiben. Die Eliminationsreihenfolge ist der Knotennumerierung zu entnehmen.

ten Knoten einer Zeitscheibe oder zweier benachbarten Zeitscheiben miteinander verbinden. Die hieraus resultierenden Cliquen vereinen somit nur Knoten einer Zeitscheibe oder zweier benachbarter Zeitscheiben. Wird nun eine neue Zeitscheibe angefügt, und die alte Eliminationsreihenfolge für die alten Knoten beibehalten, so muss der *Junction Tree* nicht komplett neu berechnet werden. Hier können alten Berechnungen wiederverwendet werden.

In Abbildung 3.13 ist der aus Abbildung 3.12 resultierende Junction Tree dargestellt. Damit die Übersicht nicht noch mehr leidet, wurden die Separatorknoten nicht eingezeichnet, da sich diese für jede Kante des Junction Trees leicht bestimmen lassen. Die durchgezogenen Linien und Kreise geben die Kanten bzw. Cliquen des Junction Trees an. Die gestrichelten Linien und Kreise stehen für die Kanten bzw. Cliquen des Junction Trees, die beim Anhängen von Zeitscheiben

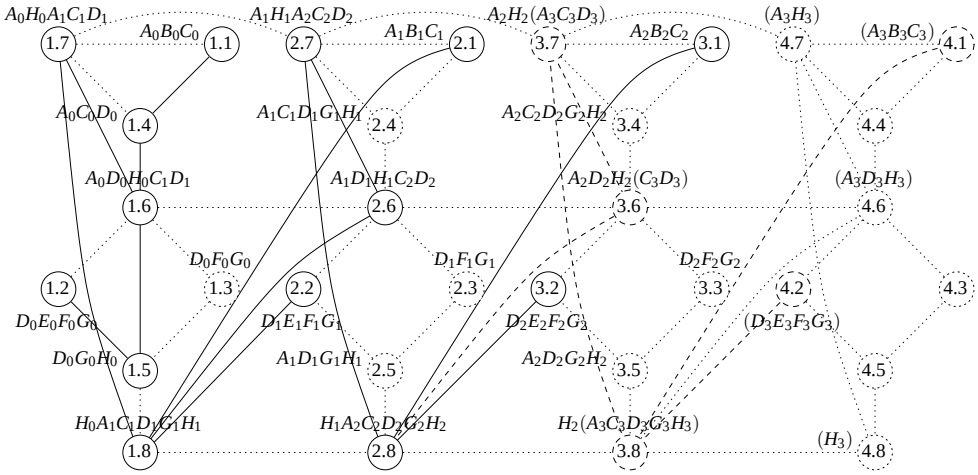



Abbildung 3.13: DBN: der Junction Tree eines dynamischen Bayesschen Netzes mit Berücksichtigung von Zeitscheiben. Die Separatorknoten wurden aus Gründen der Übersichtlichkeit nicht eingezeichnet.

entstehen. In Klammern ist angegeben, um welche Knoten sich die Cluster erweitern. Wie schon in Abschnitt 2.3.1 erwähnt wurde, wird nicht jeder Cluster zu einer Clique des Junction Trees. In der Abbildung sind die Cluster gepunktet dargestellt, die keine Cliquen des Junction Trees sind.

In Tabelle 3.7 ist die auf die Zeitscheiben eingeschränkte Eliminierung noch einmal übersichtlich dargestellt. Die Knoten der verschiedenen Zeitscheiben sind durch einen waagrechten Strich in der Tabelle voneinander getrennt. Man erkennt, dass die Knoten in den verschiedenen Zeitscheiben in der gleichen Reihenfolge abgearbeitet werden. Dabei werden die dynamischen Knoten in logischer Folgerung aus Satz 3.3.2 als letzte Knoten der Zeitscheibe abgearbeitet. Im Vergleich von Tabelle 3.6 zu Tabelle 3.7 erkennt man leider auch, dass die Einschränkung der Eliminationsreihenfolge auf die Zeitscheiben in diesem Beispiel sehr viel größere Cliquen erzeugt. In Kapitel 3.3 werden wir ein Rollup-Verfahren kennen lernen, dass sich nur bei Junction Trees einsetzen lässt, die mittels einer eingeschränkten Eliminationsreihenfolge bestimmt wurden. Dabei müssen vom dynamischen Bayesschen Netz maximal immer nur zwei Zeitscheiben repräsentiert sein. Andere Junction Trees, die mittels einer Eliminationsreihenfolge bestimmt wurden, die die Zeitscheiben nicht berücksichtigen, lassen sich mit diesem Rollup-Verfahren nicht behandeln. Die Anzahl der Zeitscheiben des dynamischen Bayesschen Netzes wächst mit der Zeit, und damit auch die Komplexität (Verbrauch von Speicherplatz und Rechenzeit) des dynamischen Bayesschen Netzes. Irgendwann übersteigt die Komplexität des dynamischen Bayesschen Netzes, das mit einer Eliminationsreihenfolge behandelt wird, die die Zeitscheiben

| *eliminierter Knoten* | *induzierter Cluster* | *induzierte Clique* |
|---|---|---|
| $B_0$ | $A_0B_0C_0$ | $Clq_1 = \{A_0, B_0, C_0\}$ |
| $E_0$ | $D_0E_0F_0G_0$ | $Clq_2 = \{D_0, E_0, F_0, G_0\}$ |
| $F_0$ | $D_0F_0G_0$ | - |
| $C_0$ | $A_0C_0D_0$ | $Clq_3 = \{A_0, C_0, D_0\}$ |
| $G_0$ | $D_0G_0H_0$ | $Clq_4 = \{D_0, G_0, H_0\}$ |
| $D_0$ | $A_0D_0H_0C_1D_1$ | $Clq_5 = \{A_0, D_0, H_0, C_1, D_1\}$ |
| $A_0$ | $A_0H_0A_1C_1D_1$ | $Clq_6 = \{A_0, H_0, A_1, C_1, D_1\}$ |
| $H_0$ | $H_0A_1C_1D_1G_1H_1$ | $Clq_7 = \{H_0, A_1, C_1, D_1, G_1, H_1\}$ |
| $B_1$ | $A_1B_1C_1$ | $Clq_8 = \{A_1, B_1, C_1\}$ |
| $E_1$ | $D_1E_1F_1G_1$ | $Clq_9 = \{D_1, E_1, F_1, G_1\}$ |
| $F_1$ | $D_1F_1G_1$ | - |
| $C_1$ | $A_1C_1D_1G_1H_1$ | - |
| $G_1$ | $A_1D_1G_1H_1$ | - |
| $D_1$ | $A_1D_1H_1C_2D_2$ | $Clq_{10} = \{A_1, D_1, H_1, C_2, D_2\}$ |
| $A_1$ | $A_1H_1A_2C_2D_2$ | $Clq_{11} = \{A_1, H_1, A_2, C_2, D_2\}$ |
| $H_1$ | $H_1A_2C_2D_2G_2H_2$ | $Clq_{12} = \{H_1, A_2, C_2, D_2, G_2, H_2\}$ |
| $B_2$ | $A_2B_2C_2$ | $Clq_{13} = \{A_2, B_2, C_2\}$ |
| $E_2$ | $D_2E_2F_2G_2$ | $Clq_{14} = \{D_2, E_2, F_2, G_2\}$ |
| $F_2$ | $D_2F_2G_2$ | - |
| $C_2$ | $A_2C_2D_2G_2H_2$ | - |
| $G_2$ | $A_2D_2G_2H_2$ | - |
| $D_2$ | $A_2D_2H_2(C_3D_3)$ | $(Clq_{15} = \{A_2, D_2, H_2, C_3, D_3\})$ |
| $A_2$ | $A_2H_2(A_3C_3D_3)$ | $(Clq_{16} = \{A_2, H_2, A_3, C_3, D_3\})$ |
| $H_2$ | $H_2(A_3C_3D_3G_3H_3)$ | $(Clq_{17} = \{H_2, A_3, C_3, D_3, G_3, H_3\})$ |

Tabelle 3.7: Eliminierung der Knoten: die Zeitscheiben werden berücksichtigt

nicht berücksichtigt, die Komplexität des dynamischen Bayesschen Netzes, von dem immer maximal nur zwei Zeitscheiben repräsentiert werden. Dann relativiert sich das Verfahren, das die Zeitscheiben berücksichtigt, gegenüber dem Verfahren, welches eine globale Strategie verfolgt.

Man erkennt, wie sich der Junction Tree durch das Anhängen von neuen Zeitscheiben entwickelt, wenn die Zeitscheiben bei der Berechnung der Eliminationsreihenfolge berücksichtigt werden.

Um eine neue Zeitscheibe anzuhängen, wird also wie folgt vorgegangen:

1. Moralisiere den zusammengesetzten Graphen aus *Junction Tree* und DAG.

2. Trianguliere (eingeschränkt auf die Zeitscheiben) den moralisierten Graphen und bestimme die neuen Cliquen.

3. Konstruiere den neuen (expandierten) *Junction Tree*.

4. Berechne die neuen Potentiale der Cliquen und berücksichtige die alten Potentiale dabei.

Vom alten *Junction Tree* $\mathscr{T} = (\mathbb{C}, \mathbb{S})$ soll so viel wie möglich wiederverwendet werden, um den Konstruktionsaufwand für den neuen *Junction Tree* $\mathscr{T}' = (\mathbb{C}', \mathbb{S}')$ zu minimieren. Als direkte Konsequenz der eingeschränkten Eliminationsreihenfolge existiert für jede alte Clique $Clq \in \mathbb{C}$ eine neue Clique $Clq' \in \mathbb{C}'$, so dass gilt: $Clq \subseteq Clq'$. Für einige Cliquen ist diese Relation strikt. Bei einer beliebigen Eliminationsreihenfolge ist eine Zuordnung alte zu neue Clique im allgemeinen nicht möglich. Der neue *Junction Tree* $\mathscr{T}$ wird nun wie folgt erzeugt:

1. Bestimme die Menge der Cliquen $\mathbb{C}'$ von $\mathscr{T}'$.

2. Erzeuge $\mathscr{T}'$:

   (a) Erzeuge die $\psi$-Tabellen der Cliquen $Clq \in \mathbb{C}' \setminus \mathbb{C}$ und der Separatoren $Sep \in \mathbb{S}' \setminus \mathbb{S}$.

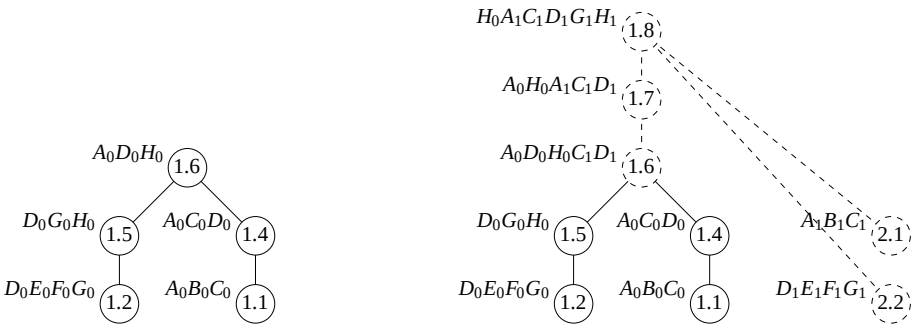
   (b) Initialisiere alle neuen Tabellen mit 1. (Alle anderen Tabellen bleiben unverändert.)

3. Für jede Clique $Clq \in \mathbb{C} \setminus \mathbb{C}'$ und jeden Separator $Sep \in \mathbb{S} \setminus \mathbb{S}'$ multipliziere die entsprechenden Belief-Potentiale zu den zugehörigen Belief-Potentialen der Cliquen und Separatoren.

4. Ordne die Knoten der neuen Zeitscheibe einer bestimmten Clique zu. Es muss dabei gelten, dass die Eltern des Knotens auch in der Clique vorhanden sind.

In den beiden Abbildungen 3.14 und 3.15 ist das Anhängen von Zeitscheiben in der Repräsentation des Junction Trees dargestellt. Dabei ist in Abbildung 3.14 zu sehen, wie vorhandene Cliquen um Knoten erweitert werden. In Abbildung 3.15 wird der Junction Tree nur um neue Cliquen ergänzt. Alte Cliquen werden nicht um Knoten erweitert.

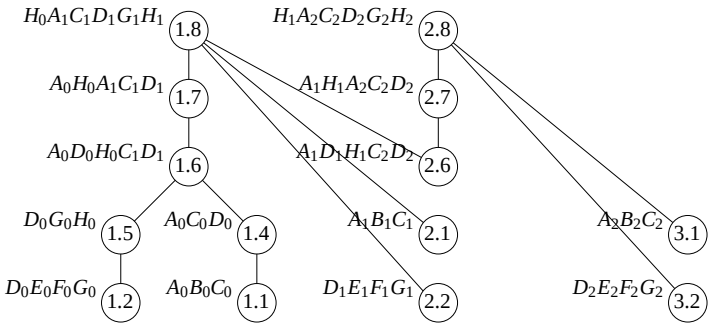
Die eingeschränkte Eliminationsreihenfolge ermöglicht somit auch die Verwaltung von *vorberechneten Zeitscheiben*. Diese bereits berechneten Zeitscheiben sind Junction Trees, deren Cliquen und Kanten berechnet und für die Collect Evidence in Richtung der Nahtstelle an die Vorgängerzeitscheibe schon durchgeführt wurde. Kann ein Schema $\mathscr{A}$ an Zeitscheiben angehängt werden, die von



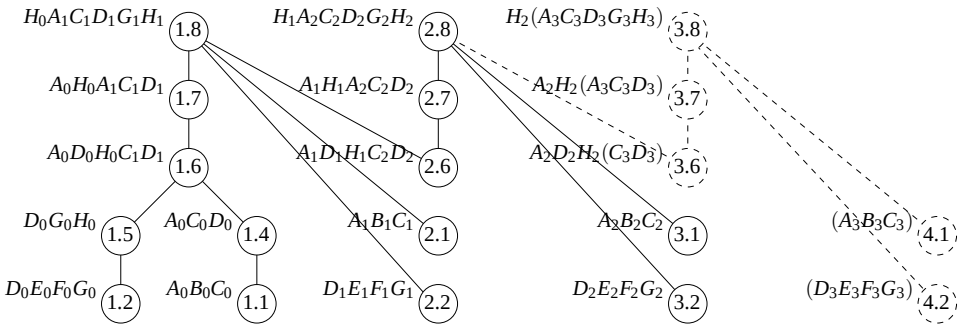

(a) der Junction Tree für Zeitscheibe 0

(b) der Junction Tree nach dem Anhängen von Zeitscheibe 1

Abbildung 3.14: In der Repräsentation des Junction Trees wird an Zeitscheibe 0 die Zeitscheibe 1 angehängt. Veränderte und neue Cliquen sind gestrichelt eingezeichnet.




(b) der Junction Tree für die Zeitscheiben 0 bis 2




(c) der Junction Tree nach dem Anhängen von Zeitscheibe 3

Abbildung 3.15: In der Repräsentation des Junction Trees wird die Zeitscheibe 3 angehängt. Die neuen Cliquen sind gestrichelt eingezeichnet. Es wurden keine alten Cliquen verändert.

verschiedenen Schemata erzeugt werden, so wird je nach Vorgängerzeitscheibe eine andere vorberechnete Zeitscheibe für die Instantiierung des Schemas $\mathscr{A}$ benötigt. Damit der gesamte Junction Tree kalibriert ist, muss, wenn keine Evidenz im neuen Junction Tree eingetragen wurde und der alten Junction Tree vor dem Anhängen des neuen Junction Tree kalibriert war, nur noch ein Distribute Evidence an der Nahtstelle zwischen altem und neu angehängtem Junction Tree in Richtung des neu angehängten Junction Trees aufgerufen werden. Wurde jedoch im neu angehängten Junction Tree Evidenz eingetragen, so muss in den Ästen des neu angehängten Junction Trees, in denen wenigstens eine Evidenz eingetragen wurde, auch ein Collect Evidence bis zur Nahtstelle durchgeführt werden. Von der Nahtstelle aus muss dann auch ein Distribute Evidence in den alten Junction Tree erfolgen (siehe dazu auch [Darwiche 01], in dem beschrieben wird, wie sich eine Inferenz bei konstantem Speicherverbrauch realisieren lässt).

Es wurden systematisch die Eigenschaften der beiden Verfahren “Behandlung des dynamischen Bayesschen Netzes als normales Bayessches Netz” und “Strukturerhaltung im Junction Tree des dynamischen Bayesschen Netzes” miteinander verglichen. Mit dem zweiten Verfahren, das auf einem von Kjærulff in [Kjærulff 93] vorgestellten Verfahren beruht, werden die Grundlagen für einige Rollup-Verfahren geschaffen. Dadurch dass sich die Struktur des dynamischen Bayesschen Netzes im Junction Tree widerspiegelt, wird ein Rollup von dynamischen Bayesschen Netzen ermöglicht, der ohne Informationsverlust durchgeführt werden kann. Im folgenden Abschnitt werden nun einige Rollup-Verfahren vorgestellt (darunter auch das Rollup-Verfahren von Kjærulff), die je nach Gegebenheit und Anwendung ihre Schwächen und Stärken haben. Eine Gegenüberstellung und Zusammenfassung von Kriterien erleichtern die Wahl, welches Rollup-Verfahren für die Anwendung am Besten zum Einsatz kommt. Gegebenenfalls lassen sich einige Rollup-Verfahren auch miteinander kombinieren.

Lösungsverfahren, die bei der Bestimmung der Eliminationsreihenfolge der Knoten den Algorithmus maximum cardinality search oder ähnliche zwingend vorschreiben, sind für dieses Rollup-Verfahren nicht geeignet.

## 3.4 Algorithmen

Wie man gesehen hat, ändert sich beim Aufspalten oder Löschen von Knoten im Bayesschen Netz die Gesamtwahrscheinlichkeit der verbleibenden Knoten. Es entsteht ein Fehler. Im folgenden werden verschiedene Algorithmen vorgestellt, die beim Entfernen von Zeitscheiben die Gesamtwahrscheinlichkeit der verbleibenden Zeitscheiben unverändert erhalten.

Die folgenden Algorithmen erhalten die Gesamtwahrscheinlichkeit der verbleibenden Zeitscheiben, indem entweder die Struktur des dynamischen Bayes-

schen Netzes beim Rollup verändert wird, oder der Rollup auf dem zugehörigen Junction Tree des dynamischen Bayesschen Netzes ausgeführt wird. Als nächstes wird ein Algorithmus vorgestellt, der die Struktur des Bayesschen Netzes verändert.

### 3.4.1 Knotenabsorption

Das Verfahren der *Knotenabsorption* dient dazu, uninteressante Knoten aus dem Bayesschen Netz zu entfernen, und dadurch gegebenenfalls die Inferenz im neuen Bayesschen Netz zu vereinfachen. Wie man aber sehen wird, kann bei der Absorption eines Knotens oder einer Knotenmenge die Komplexität des Bayesschen Netzes im Gegenteil auch steigen.

Es wird nun gezeigt, ob und wie sich das Verfahren der Knotenabsorption als Rollup-Verfahren einsetzen lässt.

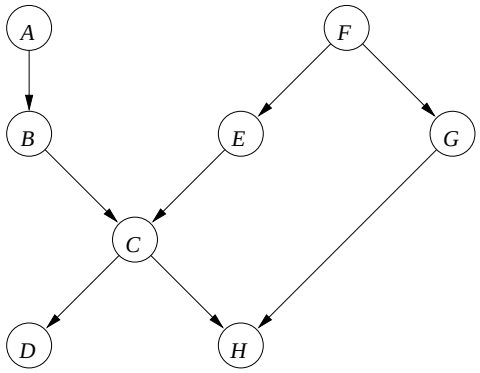
Das Verfahren der *Knotenabsorption* formt ein Bayessches Netz um, indem Knoten des Netzes so entfernt werden, dass jede Inferenz des resultierenden neuen Bayesschen Netzes dasselbe Ergebnis liefert wie das alte unveränderte Bayessche Netz. Für die absorbierten Knoten können aber keine neue Evidenz eingetragen oder die BEL-Werte abgelesen werden. Die gesamte gemeinsame Wahrscheinlichkeitsverteilung der verbleibenden Knoten bleibt unverändert. Das umgewandelte Bayessche Netz muss wieder in einen Junction Tree compiliert werden.

Bei der Absorption eines Knotens $A \in \mathbb{X}$ werden die Tabellen der bedingten Wahrscheinlichkeiten der verbleibenden Knoten $V \in \mathbb{X}$ so verändert, dass die Gesamtwahrscheinlichkeit des Bayesschen Netzes ohne Berücksichtigung des absorbierten Knotens gleich bleibt. Formal bedeutet dies:

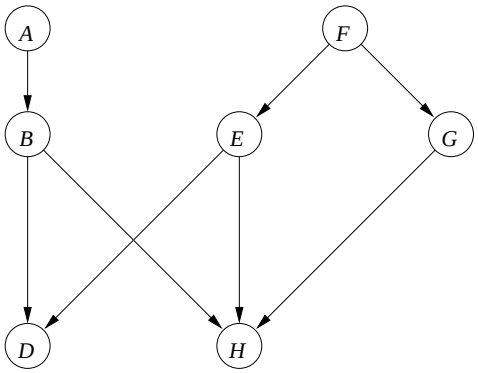
$$\forall \mathbb{K} \subset \mathbb{X} \setminus \{A\} : \sum_{\mathbb{X} \setminus \mathbb{K}} \Big( \prod_{V \in \mathbb{X}} \theta_{(V)} \, \lambda_V \Big) = \sum_{\mathbb{X} \setminus \mathbb{K}} \Big( \prod_{V \in \mathbb{X} \setminus \{A\}} \widetilde{\theta}_{(V)} \, \lambda_V \Big)$$

Die Absorption eines Knotens $A$ lässt sich zu einer Absorption einer Knotenmenge $\mathbb{A} \subset \mathbb{X}$ fortsetzen. Je nachdem in welcher Reihenfolge die Knoten aus dem Bayesschen Netz absorbiert werden, ändert sich dabei das Ergebnis der Knotenabsorption. Bei der Veränderung der Tabellen der bedingten Wahrscheinlichkeiten eines Knotens des Bayesschen Netzes, sind nicht nur die Zahlenwerte der bedingten Wahrscheinlichkeiten betroffen, sondern auch die Abhängigkeiten der Knoten untereinander ändern sich. D. h. die Struktur des Bayesschen Netzes ändert sich. Es werden Kanten gelöscht und eingefügt.

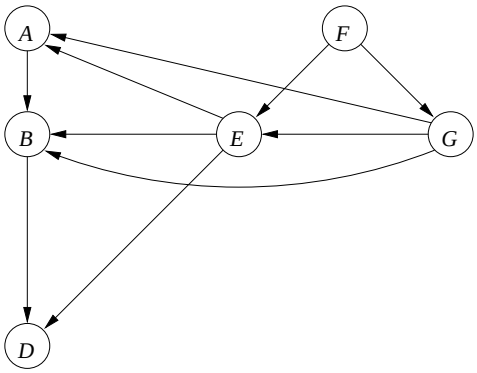
In Abbildung 3.16(a) ist das Beispielnetz Asienbesuch aus dem Beispiel 2.1.2 zu sehen, an dem nun strukturell die Absorption zweier Knoten dargestellt wird. (Für die Berechnungen wurde das Programm Netica™ herangezogen.) Die Anzahl der Einträge in der Wahrscheinlichkeitentabelle eines Knotens $V$ wird auch als




(a) Das Beispielnetz Asienbesuch (siehe Beispiel 2.1.2) dient als Ausgangsnetz. Die Anzahl der Tabelleneinträge summiert sich zu 36.




(b) Der uninstantiierte Knoten $C$ wurde absorbiert. Die Anzahl der Tabelleneinträge summiert sich zu 40.




(c) Der instantiierte Knoten $H$ wurde absorbiert. Die Anzahl der Tabelleneinträge summiert sich zu 46.

Abbildung 3.16: Knotenabsorption

Größe der Wahrscheinlichkeitentabelle oder mit $\#(\theta_{(V)})$ bezeichnet. Die Größen der Wahrscheinlichkeitentabellen aller Knoten des Bayesschen Netzes addieren sich zu $\#(\theta_{(A)})+\#(\theta_{(B)})+\#(\theta_{(C)})+\#(\theta_{(D)})+\#(\theta_{(E)})+\#(\theta_{(F)})+\#(\theta_{(G)})+\#(\theta_{(H)})=2+4+8+4+4+2+4+8=36$.

Jetzt wird der uninstantiierte Knoten $C$ des Bayesschen Netzes absorbiert. Wie in Abbildung 3.16(b) zu sehen ist, werden Kanten vom Knoten $B$ zu den Knoten $D$ und $H$ sowie Kanten vom Knoten $E$ zu den Knoten $D$ und $H$ eingefügt. Dabei werden die Einträge der Wahrscheinlichkeitentabellen und die Größe der Wahrscheinlichkeitentabellen der beiden Knoten $D$ und $H$ vergrößert. Es gilt: $\#(\widetilde{\theta}_{(D)})=8$ und $\#(\widetilde{\theta}_{(H)})=16$. Die Größen der Wahrscheinlichkeitentabellen aller Knoten des Bayesschen Netzes addieren sich zu $\#(\theta_{(A)})+\#(\theta_{(B)})+0+\#(\widetilde{\theta}_{(D)})+\#(\theta_{(E)})+\#(\theta_{(F)})+\#(\theta_{(G)})+\#(\widetilde{\theta}_{(H)})=2+4+0+8+4+2+4+16=40$.

Nun wird der Knoten $H$ instantiiert. Dadurch werden unter anderem die Kno-

ten $G$ und $A$ voneinander abhängig. Die Abhängigkeiten von Knoten(mengen) untereinander lassen sich mit dem *D-Separations-Kriterium* bestimmen: Seien $\mathbb{X}$, $\mathbb{Y}$ und $\mathbb{Z}$ drei disjunkte Teilmengen von Knoten des gerichteten azyklischen Graphen $\mathscr{G}$ des Bayesschen Netzes $\mathscr{BN}$. Dabei muss die Vereinigung von $\mathbb{X}$, $\mathbb{Y}$ und $\mathbb{Z}$ nicht die gesamte Knotenmenge des Graphen $\mathscr{G}$ ergeben. Dann *d-separiert* die Menge $\mathbb{Z}$ die beiden Mengen $\mathbb{X}$ und $\mathbb{Y}$ voneinander (Schreibweise: $<\mathbb{X}\,|\,\mathbb{Z}\,|\,\mathbb{Y}>_{\mathscr{G}}$), wenn es entlang jeden Pfades zwischen einem Knoten aus $\mathbb{X}$ und einem Knoten aus $\mathbb{Y}$ ein Knoten $w$ existiert, der eine der beiden nachfolgenden Eigenschaften erfüllt:

1. $w$ hat Vorgängerknoten und weder $w$ noch seine Nachfolgerknoten sind in $\mathbb{Z}$, oder

2. $w$ hat keine Vorgängerknoten, und $w$ ist in $\mathbb{Z}$.

Weitere Begriffe und ein Beispiel im Zusammenhang mit dem *D-Separations-Kriterium* sind für den interessierten Leser im Anhang B.3 zusammengefasst.

Nun aber wieder zurück vom D-Separations-Kriterium zur Absorption des Knotens $H$. Es werden Kanten vom Knoten $G$ zu den Knoten $A$, $B$ und $E$ sowie vom Knoten $E$ zu den Knoten $A$ und $B$ eingefügt. Dabei werden die Einträge der Wahrscheinlichkeitentabellen und die Größe der Wahrscheinlichkeitentabellen der Knoten $A$, $B$ und $E$ vergrößert. Es gilt: $\#(\widetilde{\theta}_{(A)}) = 16$, $\#(\widetilde{\theta}_{(B)}) = 16$ und $\#(\widetilde{\theta}_{(E)}) = 8$. Die Größen der Wahrscheinlichkeitentabellen aller Knoten des Bayesschen Netzes addieren sich zu $\#(\widetilde{\theta}_{(A)}) + \#(\widetilde{\theta}_{(B)}) + 0 + \#(\widetilde{\theta}_{(D)}) + \#(\widetilde{\theta}_{(E)}) + \#(\theta_{(F)}) + \#(\theta_{(G)}) + 0 = 8 + 16 + 0 + 8 + 8 + 2 + 4 + 0 = 46$.

Um mit der Absorption von Knotenmengen einen Rollup durchführen zu können, geht man nun wie folgt vor:

- Das Bayessche Netz besteht aus zwei Zeitscheiben $t-1$ und $t$. Die Knoten der Zeitscheiben vor der Zeitscheibe $t-1$ wurden alle absorbiert. Liegt Evidenz für Knoten der Zeitscheibe $t-1$ vor, so wird sie eingetragen.

- Die Knoten der Zeitscheibe $t-1$ werden absorbiert.

- Es wird die Zeitscheibe $t+1$ angehängt, und das Bayessche Netz ist für den nächsten Berechnungsvorgang bereit.

Obwohl ein Bayessches Netz, von dem einige Knoten absorbiert wurden, weniger Knoten als das vorhergehende Bayessche Netz enthält, kann es, wenn Kanten durch die Absorption eingefügt wurden, sehr viel komplexer sein als das vorhergehende Bayessche Netz. Es werden also Kanten ins Bayessche Netz eingefügt bei der Absorption

- eines Knotens mit Evidenz, der viele Vorgängerknoten besitzt, oder
- eines Knotens ohne Evidenz, der viele Nachfolgerknoten besitzt.

Es werden keine neuen Kanten im Netz eingefügt bei der Absorption von

- Knoten mit Evidenz, die keine Vorgängerknoten besitzen, oder
- Knoten ohne Evidenz, die keine Nachfolgerknoten besitzen.

Mit Hilfe des D-Separations-Kriteriums weiß man zum Beispiel, dass durch die Instantiierung eines Knotens mit Evidenz seine Vorgängerknoten voneinander abhängig werden.

Ist die Anzahl der Knoten gering, für die Evidenz eingetragen werden sollen oder an deren BEL-Werten man interessiert ist, und haben diese Knoten auch nur eine geringe Anzahl an Hypothesen, so ist auch das resultierende Bayessche Netz nach der Absorption der uninteressanten Knoten von geringer Größe. Der Berechnungsvorgang kann aber vorübergehend so enorme Speicherplatzprobleme aufwerfen, dass er wegen Speicherplatzmangel gegebenenfalls sogar abbricht.

### 3.4.2 Prediction-Estimation-Verfahren

Beim *Prediction-Estimation-Verfahren* sind immer maximal nur zwei Zeitscheiben des dynamischen Bayesschen Netzes repräsentiert. Dabei wird in dem dynamischen Bayesschen Netz eine neue Zeitscheibe etabliert und dann eine alte Zeitscheibe aufgerollt (siehe auch [Russell & Norvig 03]).

Das Anhängen einer Zeitscheibe $t$ an ein dynamisches Bayessches Netz kann mathematisch so betrachtet werden, dass an das Produkt von bedingten Wahrscheinlichkeitentabellen und findings der Knoten der alten Zeitscheiben, die bedingten Wahrscheinlichkeitentabellen und findings der Knoten der neuen Zeitscheibe $t$ hinzumultipliziert werden[3]:

$$\begin{aligned}\Pr(\mathbb{X}_t,\ldots,\mathbb{X}_0 \mid \mathbb{E}_t,\ldots,\mathbb{E}_0) = \Pr(\mathbb{X}_{t-1},\ldots,\mathbb{X}_0 \mid \mathbb{E}_{t-1},\ldots,\mathbb{E}_0)\ \Pr(\mathbb{X}_t \mid \mathbb{E}_t) = \\ = \Big(\prod_{K\in\mathbb{X}_0}\Big(\theta_{(K)}\ \lambda_K\Big)\ \ldots\ \prod_{K\in\mathbb{X}_{t-1}}\Big(\theta_{(K)}\ \lambda_K\Big)\Big) \\ \prod_{K\in\mathbb{X}_t}\Big(\theta_{(K)}\ \lambda_K\Big).\end{aligned}$$

[3] Die Schreibweise $K \in \mathbb{X}$ ist ein wenig nachlässig. Ein Knoten $K$ kann nie das Element einer Menge von Zufallsvariablen sein. Vielmehr ist mit diesem Ausdruck gemeint, dass $K$ ein Knoten ist, der mit einer Zufallsvariablen $X_K$ assoziiert ist, die in $\mathbb{X}$ enthalten ist. Streng mathematisch müsste man also für $K \in \mathbb{X}$ schreiben: $K \in \{V : V \text{ ist assoziiert mit } X_V \in \mathbb{X}\}$.

Dabei stehen die $\mathbb{X}_k$ und $\mathbb{E}_k$, mit $0 \leq k \leq t$, für die Menge der Zufallsvariablen einer Zeitscheibe $k$ bzw. die Menge der Variablen einer Zeitscheibe $k$, für die Evidenz vorliegt. Die alten Zeitscheiben werden aus dem Produkt entfernt, indem sie herausmarginalisiert werden:

$$\sum_{K \in \mathbb{X}_0 \cup \ldots \cup \mathbb{X}_{t-1}} \Pr(\mathbb{X}_t, \ldots, \mathbb{X}_0 \mid \mathbb{E}_t, \ldots, \mathbb{E}_0) =$$
$$= \sum_{K \in \mathbb{X}_0 \cup \ldots \cup \mathbb{X}_{t-1}} \Big( \prod_{K \in \mathbb{X}_0} \left( \theta_{(K)}\, \lambda_K \right) \ldots \prod_{K \in \mathbb{X}_{t-1}} \left( \theta_{(K)}\, \lambda_K \right) \prod_{K \in \mathbb{X}_t} \left( \theta_{(K)}\, \lambda_K \right) \Big).$$

In Abbildung 3.17 sind die drei Schritte eines Zyklus des Prediction-Estimation-Verfahrens grafisch dargestellt. Jeder Zyklus des Verfahrens arbeitet dabei wie folgt:

- *Prediction*: Das dynamische Bayessche Netz bestehe aus den beiden Zeitscheiben $t-1$ und $t$. Weiterhin werde angenommen, dass schon der BEL-Wert $\mathrm{BEL}(\mathbb{X}_{t-1})$[4] berechnet worden sei mit der Berücksichtigung aller Evidenz bis einschließlich $\mathbb{E}_{t-1}$. Die Zeitscheibe $t-1$ besitzt keine Verbindungen mehr zu vorhergehenden Zeitscheiben. Den Zustandsvariablen der Zeitscheibe $t-1$ wurden A-priori-Wahrscheinlichkeiten zugeordnet (siehe auch den nächsten Schritt). Dann wird der BEL-Wert $\widehat{\mathrm{BEL}}(\mathbb{X}_t)$ wie folgt neu berechnet:

  $$\widehat{\mathrm{BEL}}(\mathbb{X}_t) = \sum_{\mathbb{X}_{t-1}} \Pr(\mathbb{X}_t \mid \mathbb{X}_{t-1} = \curvearrowleft_{t-1}) \mathrm{BEL}(\mathbb{X}_{t-1} = \curvearrowleft_{t-1}).$$

  Die $\curvearrowleft_{t-1}$ gehen dabei alle möglichen Werte von $\mathbb{X}_{t-1}$ durch. Die Berechnung lässt sich durch einen Update des aktuellen Bayesschen Netzes (Zeitscheiben $t-1$ und $t$ mit den A-priori-Wahrscheinlichkeiten für die Zustandsvariablen der Zeitscheibe $t-1$) mit der Evidenz $\mathbb{E}_{t-1}$ erreichen.

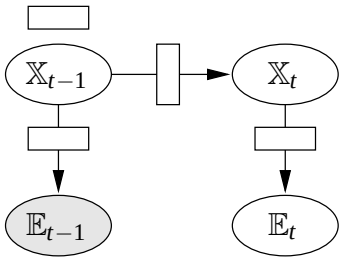
- *Rollup*: Die Zeitscheibe $t-1$ wird entfernt. Dazu muss eine A-priori-Wahrscheinlichkeitentabelle für die Zustandsvariablen der Zeitscheibe $t$ hinzugefügt werden. Diese A-priori-Wahrscheinlichkeitentabelle ist gerade

  $$\widehat{\mathrm{BEL}}(\mathbb{X}_t).$$

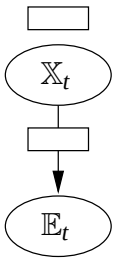
[4] $\mathrm{BEL}(\mathbb{X}_{t-1})$ steht für

$$\sum_{K \in \mathbb{X}_0 \cup \ldots \cup \mathbb{X}_{t-2}} \Big( \prod_{K \in \mathbb{X}_0} \left( \theta_{(K)}\, \lambda_K \right) \ldots \prod_{K \in \mathbb{X}_{t-1}} \left( \theta_{(K)}\, \lambda_K \right) \Big).$$

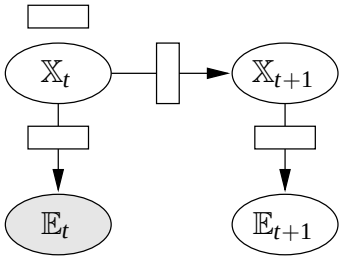
- *Estimation*: Jetzt wird die neue Evidenz $\mathbb{E}_t$ eingetragen, und dann der BEL-Wert $\mathrm{BEL}(\mathbb{X}_t)$, d. h. die Wahrscheinlichkeitsverteilung des aktuellen Zustandes, durch ein Update des jetzigen dynamischen Bayesschen Netzes berechnet. Anschließend wird die Zeitscheibe $t+1$ angehängt, und das dynamische Bayessche Netz ist für den nächsten Berechnungsvorgang bereit.




(a) Prediction: Berechnung von $\widehat{\mathrm{BEL}}(\mathbb{X}_t)$ unter Berücksichtigung von $\mathrm{BEL}(\mathbb{X}_{t-1})$ und $\mathbb{E}_{t-1}$.




(b) Rollup: Die neue A-priori-Wahrscheinlichkeitentabelle ist $\widehat{\mathrm{BEL}}(\mathbb{X}_t)$.




(c) Estimation: Berechnung von $\mathrm{BEL}(\mathbb{X}_t)$ bei gegebener neuer Evidenz $\mathbb{E}_t$.

Abbildung 3.17: Die drei Schritte beim Prediction-Estimation-Verfahren

Die Speicherung des *Zustandsraumes* $\mathrm{BEL}(\mathbb{X}_t)$ kann dabei zum Problem werden. Die Tabelle kann so groß werden, dass sie sich nicht mehr repräsentieren bzw. verarbeiten lässt. Dieses ist von den involvierten Knoten $X \in \mathbb{X}_t$ und ihrer Hypothesenanzahl abhängig. Ebenso findet bei diesem Verfahren keine Neuberechnung alter Zeitscheiben statt.

Soll eine alte Zeitscheibe, die abgeschnitten wurde, neu berechnet werden, kann das Verfahren von Kjærulff (siehe auch [Kjærulff 95]) angewendet werden, das im nachfolgenden Abschnitt vorgestellt wird.

### 3.4.3 dHugin

In [Kjærulff 92] wird ein Schema zur Verarbeitung dynamischer Bayesscher Netze vorgestellt. Es heißt dHUGIN und stellt eine Erweiterung von HUGIN dar. Mit dHUGIN werden die Zeitscheiben des dynamischen Bayesschen Netzes nicht aufgerollt sondern nur aus dem Inferenzprozess herausgenommen. Die aus dem Inferenzprozess herausgenommenen Zeitscheiben des dynamischen Bayesschen Netzes können bei Bedarf wieder in den Inferenzprozess aufgenommen werden. D. h. wird in einer neuen Zeitscheibe Evidenz eingetragen, so lassen sich herausgenommene Zeitscheiben wieder in den Inferenzprozess aufnehmen und somit auffrischen. Dazu wird das dynamische Bayessche Netz in verschiedene Modelle aufgeteilt.

Da die Zeitscheiben des dynamischen Bayesschen Netzes die Markov-Eigenschaft erfüllen (d. h. die Zukunft ist unabhängig von der Vergangenheit, wenn die Gegenwart bekannt ist), kann das dynamische Bayessche Netz dargestellt werden als eine Reihe von Modellen $\mathscr{P}_1, \ldots, \mathscr{P}_{N+1}$.

Das Modell $\mathscr{P}_N$ wird auch als *Zeitfenster* (*time window*) bezeichnet, die Modelle $\mathscr{P}_1, \ldots, \mathscr{P}_{N-1}$ als *Backward smoothing models* und das Modell $\mathscr{P}_{N+1}$ als *Forecasting model* (siehe hierzu Abbildung 3.18).

Im Zeitfenster des dynamischen Bayesschen Netzes sind die Zeitscheiben vertreten, die im Inferenzprozess enthalten sind. Für die Knoten kann man die BEL-Werte abfragen und Evidenz eintragen. Das Zeitfenster kann vergrößert, verkleinert und verschoben werden. Das Zeitfenster ist als Junction Tree repräsentiert, der nach dem Verfahren "Strukturerhaltung im Junction Tree des dynamischen Bayesschen Netzes" konstruiert wurde, das im Kapitel 3.3.2.2 beschrieben wird.

Die Backward smoothing models entsprechen jeweils einer Zeitscheibe, die aus dem Inferenzprozess herausgenommen wurden. Die Modelle $\mathscr{P}_1, \ldots, \mathscr{P}_{N-1}$ sind einzeln als Junction Tree repräsentiert.

Dem Forecasting model $\mathscr{P}_{N+1}$ entsprechen mehrere Zeitscheiben, die als gerichteter azyklischer Graph repräsentiert sind. Das Modell dient dazu, eine schnelle Voraussage über die Entwicklung zu erhalten. Die Inferenz auf diesem gerichteten azyklischen Graphen erfolgt durch *Stochastische Simulation*, da man an einem schnellen Ergebnis interessiert ist, die Genauigkeit aber nicht so wichtig ist. Evidenz kann in diesem Modell nicht eingetragen werden.

Das Zeitfenster wird wie folgt vorwärts geschoben:

- *Vergrößerung des Zeitfensters*: An den Junction Tree des Zeitfensters wird

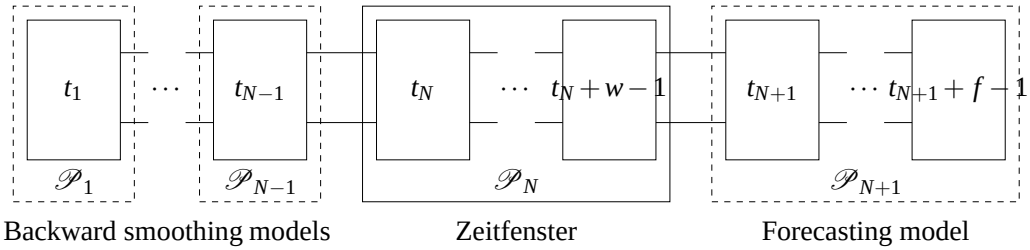



Abbildung 3.18: dHUGIN: Einteilung des dynamischen Bayesschen Netzes in eine Reihe von Modellen

ein neuer Teilbaum angehängt.

- *Verkleinerung des Zeitfensters*: Vom Junction Tree des Zeitfensters werden Teilbäume aus dem Inferenzprozess herausgenommen (sogenanntes emph-pruning). Dabei wird in dieser Teilbäumen kein Collect Evidence aufgerufen und auch Distribute Evidence nicht mehr durchgeführt.

Wie in [Kjærulff 93] beschrieben wird, wird bei der *Vergrößerung des Zeitfensters* um $k$ Zeitscheiben wie folgt vorgegangen:

1. Es werden $k$ neue konsekutive Zeitscheiben dem *Forecasting model* hinzugefügt.
2. Die $k$ ältesten Zeitscheiben des *Forecasting models* werden zum Zeitfenster geschoben.
3. Der Graph, der aus dem triangulierten Graphen des Zeitfensters und dem gerichteten Graphen der $k$ neuen Zeitscheiben besteht, wird moralisiert.
4. Der moralisierte Graph wird trianguliert nach dem Verfahren “Strukturerhaltung im Junction Tree des dynamischen Bayesschen Netzes”, und die neuen Cliquen werden bestimmt.
5. Der neue (expandierte) Junction Tree wird erzeugt.
6. Unter Berücksichtigung der alten Cliquen werden die Werte für die neuen Cliquen berechnet.

Bei der *Verkleinerung des Zeitfensters* um $k$ Zeitscheiben wird wie folgt vorgegangen:

1. Es werden $k$ Teilbäume vom Junction Tree des Zeitfensters abgeschnitten. Die $k$ Teilbäume entsprechen dabei den $k$ Zeitscheiben, um die das Zeitfenster verkleinert werden soll. (Ein Teilbaum besteht nur aus Cliquen, in

denen ein Knoten der Zeitscheibe enthalten ist, für den der Teilbaum stehen soll. Ein Teilbaum einer Zeitscheibe enthält in seinen Cliquen alle Knoten der Zeitscheibe.)

2. Diese Teilbäume werden ihren entsprechenden Backward smoothing models zugeordnet. (Über die Schnittstellen an denen die Teilbäume voneinander getrennt wurden, kann der Informationsaustausch stattfinden.)

Dies ist deswegen so einfach möglich, da der Junction Tree nach dem Verfahren "Strukturerhaltung im Junction Tree des dynamischen Bayesschen Netzes" konstruiert ist, das in Abschnitt 3.3.2.2 beschrieben wird.

Jedem *Backward smoothing model* $\mathscr{P}_n$ ($1 \leq n \leq N-1 = b$) ist eine Zeitscheibe $t_n$ zugeordnet, die auch als *Backward smoothing slice* bezeichnet wird. Das *Forecasting model* $\mathscr{P}_{N+1}$ enthält $f = w_{N+1}$ Zeitscheiben, nämlich die Zeitscheiben $t_{N+1}$ bis $t_{N+1} + f - 1$. Diese Zeitscheiben werden auch als *Forecast slices* bezeichnet. Das Zeitfenster $\mathscr{P}_N$ enthält die $w$ Zeitscheiben $t_N, \ldots, t_N + w - 1$, die auch als *Time window slices* bezeichnet werden.

Das aktuelle dynamische Bayessche Netz besteht somit aus den folgenden Zeitscheiben:

- $b$ Backward smoothing slices,
- $w$ Time window slices und
- $f$ Forecasting slices.

Beschränkt man die maximale Größe des Zeitfensters auf zwei Zeitscheiben (im Zeitfenster muss wenigstens immer eine Zeitscheibe vertreten sein), und erlaubt somit auch nur die Vergrößerung des Zeitfensters um eine Zeitscheibe, so lässt sich das Prediction-Estimation-Verfahren simulieren, wenn das Zeitfenster nicht rückwärts geschoben wird. Man erkennt, dass bei dHUGIN ähnliche Probleme wie beim Prediction-Estimation-Verfahren auftreten. Jeder Teilbaum des Junction Tree enthält eine Clique mit allen Interfaceknoten der zugehörigen Zeitscheibe. Dynamische Bayessche Netze mit einer großen Anzahl von Interfaceknoten oder Interfaceknoten mit einer großen Hypothesenanzahl (so dass die Clique mit den Interfaceknoten aus Speicherplatzproblemen und/oder Rechenzeitbedarf unhabbar groß wird) werden mit dHUGIN nicht ohne weiteres behandelt werden können.

### 3.4.4 Approximativer Rollup

Wie man bei den vorigen Rollup-Verfahren gesehen hat, kann die Repräsentation eines Zustandsraumes (der BEL-Wert über allen relevanten Knoten der Zeitscheibe) aus Speicherplatzproblemen und/oder Rechenzeitbeschränkungen unmöglich

werden, da er ohne Informationsverlust nicht faktorisiert dargestellt werden kann. Beim approximativen Rollup wird dieser Informationsverlust hingenommen und der genaue BEL-Wert durch einen approximativen BEL-Wert angenähert, indem der genaue BEL-Wert faktorisiert wird und als approximativer BEL-Wert kompakter gespeichert werden kann. Genauer gesagt, der approximative BEL-Wert wird direkt ausgerechnet, ohne dass der genaue BEL-Wert benötigt wird. Bei einer geeigneten Wahl der Approximation des BEL-Wertes akkumuliert sich der Fehler über die Zeit nicht.

Im folgenden werden wir den approximativen Rollup vorstellen, wie er in [Boyen & Koller 98] und [Boyen & Koller 99] beschrieben wird. Die Schreibweise, die in den beiden Artikeln benutzt wird, wurde dabei an die Schreibweise dieser Arbeit angepasst.

Die Menge der dynamischen Knoten der Zeitscheibe $t$ ist die Menge $\mathbb{D}_t := \{V : V \in \mathbf{pa}(\mathbb{V}(t+1))\}$. (In [Boyen & Koller 98] wird die Menge der Zufallsvariablen, die mit den dynamischen Knoten assoziiert sind, als die *kanonische Menge der Zufallsvariablen* bezeichnet.) Die Menge der dynamischen Knoten ist also die Menge der Knoten, die Elternknoten eines Knotens der darauffolgenden Zeitscheibe sind. Ein dynamisches Bayessches Netz, das aus zwei Zeitscheiben besteht, ist in *kanonischer Form*, wenn zum Zeitpunkt $t+1$ in der Zeitscheibe $t$ nur noch die Menge der dynamischen Knoten vorhanden sind. Die Menge der dynamischen Knoten wird in disjunkte Teilmengen $\mathbb{K}_{1,t}, \ldots, \mathbb{K}_{L,t}$ partitioniert, d. h. $\dot{\cup}_\ell \mathbb{K}_{\ell,t} = \mathbb{D}_t$. Zusätzlich muss die Partition die folgende Forderung erfüllen: Keine disjunkte Teilmenge $\mathbb{K}_{\ell,t}$ $(1 \leq \ell \leq L)$ darf durch eine andere Teilmenge $\mathbb{K}_{\ell',t}$ $(1 \leq \ell' \leq L, \ell' \neq \ell)$ innerhalb derselben Zeitscheibe $t$ beeinflusst werden, d. h., wenn $U \in \mathbb{K}_{\ell,t}$ in der Zeitscheibe $t$, dann kann $U$ keinen Vorgängerknoten $V$ in der Zeitscheibe $t$ besitzen mit $V \in \mathbb{K}_{\ell',t}$, $\mathbb{K}_{\ell,t} \neq \mathbb{K}_{\ell',t}$.

Als erstes wird nun ein Junction Tree konstruiert, in dem für jedes $\ell$, $1 \leq \ell \leq L$, einige Cliquen des Junction Trees die Partitionen von $\mathbb{D}_t$ und einige Cliquen des Junction Trees die Partitionen von $\mathbb{D}_{t+1}$ enthalten. Es wird die Evidenz $\mathbb{E}_{t+1}$ eingetragen und dann eine Propagierung auf dem Junction Tree durchgeführt. Aus den entsprechenden Cliquen können nun die Werte für die Zufallsvariablen der Zeitscheibe $t+1$, die mit den Partitionen assoziiert sind, herausgezogen werden.

Weitere Einsparungen können erzielt werden, wenn immer dasselbe Schema zur Instantiierung einer Zeitscheibe herangezogen und immer dieselbe Approximation vorgenommen wird. Dann ist es möglich, die Struktur des Junction Trees vorauszuberechnen, und die Knoten des dynamischen Bayesschen Netzes den entsprechenden Cliquen des Junction Trees zuzuordnen. Dieses resultiert in einer Vorlage eines Junction Trees, die als Ausgangspunkt für jede Propagierung herangezogen werden kann.

Das Vorgehen beim approximativen Rollup kann man nun wie folgt zusammenfassen:

- Eingabe:
  1. zwei Zeitscheiben eines dynamischen Bayesschen Netzes in kanonischer Form
  2. eine Partition der dynamischen Knoten $\mathbb{D} := \cup_{\ell=1}^{L} \mathbb{K}_\ell$
  3. ein approximativer BEL-Wert $\widetilde{\mathrm{BEL}}(\mathbb{D}_0)$ als Initialisierung
  4. Evidenz $\mathbb{E}_1, \mathbb{E}_2, \mathbb{E}_3, \ldots$
- Ausgabe: Eine Reihe approximativer BEL-Werte

$$\widetilde{\mathrm{BEL}}(\mathbb{D}_1),\ \widetilde{\mathrm{BEL}}(\mathbb{D}_2),\ \widetilde{\mathrm{BEL}}(\mathbb{D}_3),\ \ldots$$

- Vorgehen:
  1. Konstruktion eines Junction Trees für das gegebene dynamische Bayessche Netz aus zwei Zeitscheiben. Jedes $\mathbb{K}_{\ell,t}$ und $\mathbb{K}_{\ell,t+1}$ muss wenigstens in einer Clique des Junction Trees vollständig enthalten sein.
  2. Initialisiere jeden Cliquen-Faktor mit der konstanten Funktion = 1.
  3. Ordne jeden Knoten des dynamischen Bayesschen Netzes einer Clique des Junction Trees zu. Sei $\mathscr{T}_0$ die daraus resultierende (unkalibrierte) "Vorlage eines Junction Trees".
  4. Für $t = 0, 1, 2, \ldots$:
     (a) Lege eine Kopie $\mathscr{T}$ des Junction Trees $\mathscr{T}_0$ an, auf der gearbeitet wird. Erzeuge $\widetilde{\mathrm{BEL}}(\mathbb{D}_{t+1})$ als neuen approximativen BEL-Wert, der noch nicht belegt ist.
     (b) Trage jeden Faktor $\mathrm{BEL}(\mathbb{K}_{\ell,t})$ von $\widetilde{\mathrm{BEL}}(\mathbb{D}_t)$ in die entsprechende Clique des Junction Trees $\mathscr{T}$ ein.
     (c) Trage die Evidenz $\mathbb{E}_{t+1}$ im Junction Tree $\mathscr{T}$ ein.
     (d) Kalibriere die Cliquen des Junction Trees $\mathscr{T}$ zueinander, d. h. führe auf dem Junction Tree $\mathscr{T}$ eine Propagierung durch.
     (e) Für jedes $\ell$ berechne $\mathrm{BEL}(\mathbb{K}_{\ell,t+1})$ auf dem Junction Tree $\mathscr{T}$ und speichere es im approximativen BEL-Wert $\widetilde{\mathrm{BEL}}(\mathbb{D}_{t+1})$.
     (f) Lösche den Junction Tree $\mathscr{T}$ und gebe $\widetilde{\mathrm{BEL}}(\mathbb{D}_{t+1})$ aus.

Um dieses Verfahren auf einem bestimmten dynamischen Bayesschen Netz anwenden zu können, muss eine Partition für die dynamischen Knoten bestimmt werden. Dabei ist zu beachten, dass kleine Partitionen mit einer geringen Anzahl von Knoten eine schnellere Propagierung ermöglichen. Jedoch müssen die Partitionen groß genug sein, damit keine Kanten zwischen Partitionen innerhalb einer

Zeitscheibe vorhanden sind. Weiterhin muss beachtet werden, wenn die Partitionen klein gewählt werden, dass der Fehler größer ist durch die Annahme, dass die Partitionen unabhängig voneinander sind. Hat man insbesondere zwei (Mengen von) Knoten, die sich untereinander stark beeinflussen, so ist es sehr wahrscheinlich eine schlechte Idee diese beiden in zwei verschiedene Partitionen aufzuteilen.

In [Boyen & Koller 98] wird informell argumentiert, dass der Fehler durch die Approximation des BEL-Wertes der dynamischen Knoten gering ist, wenn die Interaktion zwischen den Partitionen der Variablen gering ist. Dieses wird in [Boyen & Koller 99] genauer ausgeführt. Hier werden informationstheoretische Definitionen für die Begriffe *schwache Interaktion* und *spärliche Interaktion* eingeführt, um die Bedingungen zu analysieren, unter denen der Fehler durch die Approximation des BEL-Wertes der dynamischen Knoten klein ist. Wie in [Boyen & Koller 98] bewiesen wird, schaukelt sich der Fehler über die Zeit nicht auf, d. h. er ist beschränkt.

Bei dynamischen Bayesschen Netzen, bei denen zu jedem Zeitpunkt schon im Voraus bekannt ist, welches Schema zu welchem Zeitpunkt als Zeitscheibe instantiiert wird, wird ein kleiner Fehler, der zudem noch beschränkt ist, keine großen Auswirkungen auf das dynamische Bayessche Netz haben. Hingegen sieht dies bei dynamischen Bayesschen Netzen anders aus, bei denen erst bei der Instantiierung einer Zeitscheibe entschieden wird, welches Schema zu verwenden ist. Zum Beispiel werden in [Schäfer 99] dynamische Bayessche Netze in der Dialogsteuerung verwendet, die aufgrund der BEL-Werte bestimmter Knoten(-mengen) des dynamischen Bayesschen Netzes entscheiden, welches Zeitscheibenschema als nächste Zeitscheibe instantiiert wird. Je nachdem wie sensibel die Dialogsteuerung ist, kann schon ein kleiner Fehler die Dialogentwicklung in eine andere Richtung lenken. Die Dialogentwicklung kann aber auch in die andere Richtung gehen, wenn schon im exakten Fall eine knappe Entscheidung getroffen wird, die im approximativen Fall umschwenkt.

## 3.5 Vergleich der Algorithmen

Im folgenden werden die wichtigsten Eigenschaften der Rollup-Verfahren kurz zusammengefasst. In einer Tabelle sind dazu die Vor- und Nachteile der verschiedenen Verfahren und ihre Anwendbarkeit aufgelistet.

Die drei Verfahren Knotenabsorption, Prediction-Estimation und dHUGIN benötigen beim Rollup den aktuellen BEL-Wert der Zeitscheibe, die aufgerollt werden soll. In diesem BEL-Wert ist das gesamte Wissen enthalten, das für die folgende Zeitscheibe notwendig ist, um eine genaue Inferenz durchführen zu können. Dieser BEL-Wert kann leider nicht faktorisiert dargestellt werden, ohne dass ein Fehler entsteht. Dieses kann bei sehr komplexen dynamischen Bayesschen Netzen

zu Problemen bei der Berechnung und Speicherung des BEL-Wertes führen.

Beim approximativen Rollup wird der BEL-Wert einer Zeitscheibe faktorisiert. Dadurch entsteht ein Fehler, der aber bei einer geeigneten Faktorisierung beschränkt ist.

In Tabelle 3.8 sind die Vor- und Nachteile der verschiedenen Verfahren und ihre Anwendbarkeit aufgelistet.

| *Verfahren* | *Vorteile* | *Nachteile* | *Anwendbarkeit* |
|---|---|---|---|
| Knotenabsorption | Exaktes Verfahren<br>Erzeugt im allgemeinen nichtkomplexe Netze bzgl. Inferenzverfahren, wenn nur wenige Knoten des ursprünglichen Netzes erhalten bleiben | Während der Berechnung sind Speicherplatzprobleme möglich, die zum Abbruch führen können (auch wenn kleine Netze erhalten blieben) | Für dynamische Bayessche Netze, die nicht komplex bzgl. Inferenzverfahren sind |
| Prediction-Estimation | Exaktes Verfahren<br>Gut für das Verständnis der Rollup-Verfahren im allgemeinen | BEL-Wert einer Zeitscheibe ist nicht faktorisiert darstellbar | Für dynamische Bayessche Netze, die nicht komplex bzgl. Inferenzverfahren sind |
| dHugin | Exaktes Verfahren<br>Die Zeitscheiben können rückwirkend eingeschätzt werden | Die Interface-Cliquen der Zeitscheiben des Zeitfensters und der Backward smoothing models können bzgl. Speicherplatzbedarf und Rechenzeitverbrauch zu groß werden | Für dynamische Bayessche Netze, die nicht komplex bzgl. Inferenzverfahren sind |
| approximativer Rollup | Schnellere Inferenz<br>Geringerer Speicherverbrauch<br>Fehlertolerant | BEL-Werte der Knoten sind mit einem Fehler behaftet<br>Faktorisierung nicht beliebig erlaubt | Dynamische Bayessche Netze, die sich faktorisieren lassen und keine exakten BEL-Werte benötigen |
| JavaDBN | Exaktes Verfahren (mit approximativem Rollup kombinierbar)<br>Parametrierung möglich<br>Sensitivitätsanalyse möglich<br>Laufzeitgarantie<br>Speicherverbrauch konstant | Komplexität ist abhängig vom Junction Tree | Für eingebettete Systeme |

Tabelle 3.8: Vergleich der Rollup-Verfahren

# Kapitel 4

# Differentieller Ansatz für dynamische Bayessche Netze

Im folgenden wollen wir die Ideen von Darwiche (siehe Abschnitt 2.4 und [Darwiche 00]) erweitern und auf dynamische Bayessche Netze anwenden. Dazu definieren wir Polynome, die es uns erlauben, in einem dynamischen Bayesschen Netz den BEL-Werte eines Knotens in einer beliebigen Zeitscheibe zu bestimmen, alte Zeitscheiben abzuschneiden und neue Zeitscheiben anzuhängen.

Wir wollen unser Verfahren anhand des folgenden dynamischen Bayesschen Netzes zuerst veranschaulichen, um dann die allgemeingültigen Prozeduren zu entwickeln. Sei dazu das dynamische Bayessche Netz gegeben, das in Abbildung 4.1 dargestellt ist. Es besteht zu diesem Zeitpunkt aus drei Zeitscheiben, die durch zwei *Zeitscheibenschemata* erzeugt wurden, die in Abbildung 4.2 dargestellt sind. Das erste Schema, das für die erste Zeitscheibe benutzt wurde, ist einfach ein Bayessches Netz. Das zweite Schema, das jeweils für eine der nachfolgenden Zeitscheiben verwendet wurde, ist ein Bayessches Netz mit der zusätzlichen Angabe der Elternknoten, die zur vorhergehenden Zeitscheibe gehören.

Nehmen wir an, dass wir Evidenz $\mathbf{e} = \mathbf{e}_1, \ldots, \mathbf{e}_3$ für die Zeitscheiben 1 bis 3 vorliegen haben, und wir am BEL-Wert für einen Knoten in Zeitscheibe 2 interessiert sind. Wir müssen dann unterscheiden zwischen (a) Vorwärtspropagierung, die den Einfluss der Evidenz von Zeitscheibe 1 zur Zeitscheibe 2 bringt; und (b) Rückwärtspropagierung, die den Einfluss der Evidenz von Zeitscheibe 3 zur Zeitscheibe 2 bringt.

Im folgenden definieren wir Verfahren, die es uns erlauben, in einem dynamischen Bayesschen Netz mit $L$ Zeitscheiben den BEL-Werte eines Knotens in Zeitscheibe $t$ (mit $1 \leq t \leq L$) zu bestimmen, alte Zeitscheiben abzuschneiden und neue Zeitscheiben anzuhängen. Diese Berechnungen können in konstanter Zeit

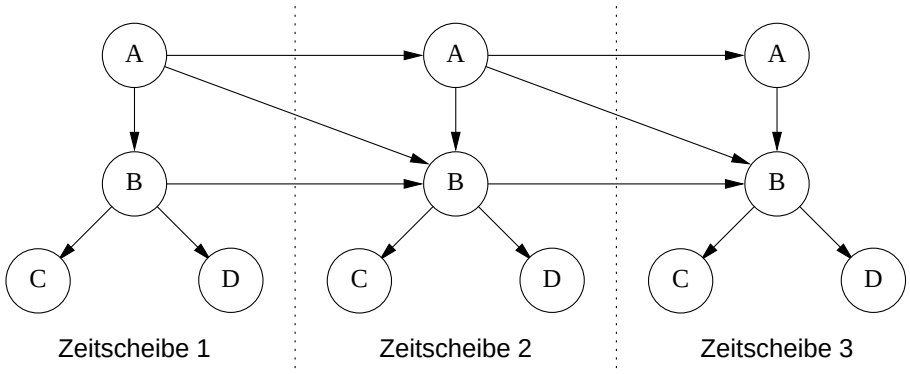



Abbildung 4.1: Beispiel eines dynamischen Bayesschen Netzes, das im Text diskutiert wird [$\Pr(A_1, B_1, \ldots) = \Pr(A_1)\Pr(B_1 \mid A_1)\ldots$]

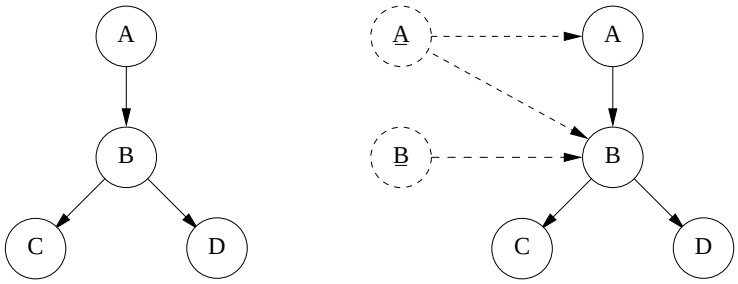



Abbildung 4.2: Zeitscheibenschemata 1 und 2 für das dynamische Bayessche Netz in Abbildung 4.1

durchgeführt werden (in Abhängigkeit von der Struktur der Zeitscheibenschemata). Der Rollup wird durchgeführt, indem die Variablen des Polynoms mit Werten belegt werden und das Polynom vereinfacht wird, indem wir es auswerten. Neue Zeitscheiben werden effizient angehängt, indem Polynome und Berechnungen vorhergehender Zeitscheiben wiederverwendet werden.

Üblicherweise verwenden wir nicht eine beliebige Eliminationsreihenfolge für das gesamte dynamische Bayessche Netz, sondern eine Eliminationsreihenfolge, die auf eine Zeitscheibe eingeschränkt ist. Auf diese Weise erhalten wir für jedes Zeitscheibenschema eine allgemeine Prozedur für das Teilpolynom, das zu diesem Zeitscheibenschema gehört.

## 4.1 Vorwärtspropagierung

Als erstes wollen wir uns den Fall der Vorwärtspropagierung anschauen. Um die Darstellung zu vereinfachen, nehmen wir an, dass wir wie in unserem Beispiel

für ein dynamisches Bayessches Netz nur ein Zeitscheibenschema für die Instantiierung der ersten Zeitscheibe und auch nur ein Zeitscheibenschema für die Instantiierung der nachfolgenden Zeitscheiben haben. Später werden wir skizzieren, wie sich das Verfahren verallgemeinern lässt, um mit einer beliebigen Anzahl an Zeitscheibenschemata zur Instantiierung von initialen und nachfolgenden Zeitscheiben umgehen zu können.

Für die erste Zeitscheibe bestimmen wir ein Verfahren, das es uns erlaubt einfach und effizient das alte Polynom zu erweitern, wenn eine neue Zeitscheibe dem dynamischen Bayesschen Netz angehängt wird. Dazu wollen wir uns die Zeitscheiben 1 und 2 in Abbildung 4.1 anschauen. Man erkennt, dass in Zeitscheibe 1 die Knoten nicht herausmarginalisiert werden können, die Elternknoten von Knoten in Zeitscheibe 2 sind, solange die Zeitscheibe 2 nicht angehängt wurde. Das Ergebnis der Prozedur für die erste Zeitscheibe ist eine Tabelle über den Knoten der aktuellen Zeitscheibe, die nicht herausmarginalisiert werden konnten. Da nur ein Zeitscheibenschema für die Instantiierung der nachfolgenden Zeitscheiben in Frage kommt, wissen wir, welche Knoten in der aktuellen Zeitscheibe Elternknoten von Knoten in der nachfolgenden Zeitscheibe werden, und wir können somit die Knoten bestimmen, die zum Zustandsentwicklungsmodell gehören. (Die Menge dieser Knoten bezeichnen wir mit $\mathbf{bs}(.)$.) In unserem Beispiel sind dies die Knoten $A_1$ und $B_1$, so dass gilt: $\mathbf{bs}(1) = \{A_1, B_1\}$. Die Indizes der Knoten bezeichnen die Zeitscheibe, zu der sie gehören.

In unserem Beispiel für ein dynamisches Bayessches Netz erhalten wir als Prozedur zur Vorwärtspropagierung für die erste Zeitscheibe:

$$\text{fwd}_{\text{init}} = \theta_{(A_1)}\ \lambda_{A_1}\ \theta_{(B_1|A_1)}\ \lambda_{B_1}\ \left(\sum_{C_1} \theta_{(C_1|B_1)}\ \lambda_{C_1}\right)\ \left(\sum_{D_1} \theta_{(D_1|B_1)}\ \lambda_{D_1}\right).$$

Wie wir bereits erwähnt haben, ist das Ergebnis eine Tabelle über dem Cartesischen Produkt der Hypothesen der Knoten $A_1$ und $B_1$, d. h. $\text{fwd}_{\text{init}} = \mathbf{T}_{(A_1B_1)}$. Die Tabelleneinträge von $\mathbf{T}_{(A_1B_1)}$ selbst sind Polynome. Wir erhalten das Polynom für die erste Zeitscheibe, indem wir über alle Tabelleneinträge summieren — d. h. über alle Knoten die bisher ausgelassen wurden. Mit diesem Polynom ist es beispielsweise möglich, den BEL-Wert eines Knotens zu berechnen. In Abbildung 4.3 wird ein arithmetischer Schaltkreis gezeigt, der die Prozedur für die Vorwärtspropagierung (durchgezogene Linien) und das Polynom (gestrichelte und durchgezogene Linien) für die erste Zeitscheibe repräsentiert. Die Tabelleneinträge von $\mathbf{T}_{(A_1B_1)}$ sind in Abbildung 4.3 als $\mathrm{T}_{(a_1b_1)}, \ldots, \mathrm{T}_{(a_2b_2)}$ oben dargestellt. Jeder Knoten des arithmetischen Schaltkreises ist mit einer Beschriftung versehen (z. B. $c_1b_1$). Konsistente Beschriftungen auf einer Ebene spannen eine Tabelle auf. Zum Beispiel spannen $a_1$ and $a_2$ die Tabelle $\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$ auf, und $b_1a_1, \ldots, b_2a_2$ spannen die

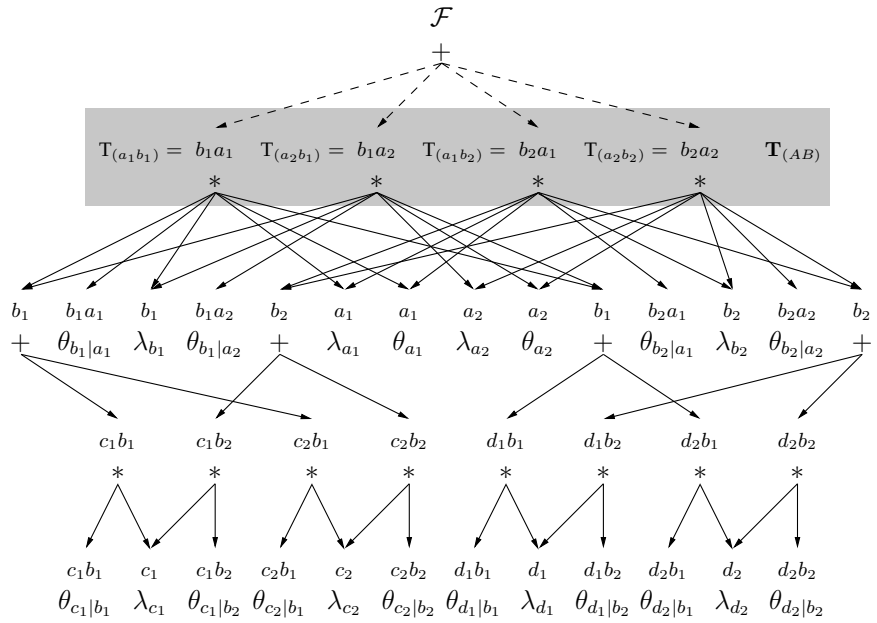



Abbildung 4.3: Ein arithmetischer Schaltkreis, der die Prozedur zur Vorwärtspropagierung (durchgezogene Linien) und das Polynom (gestrichelte und durchgezogene Linien) für das Zeitscheibenschema 1 in Abbildung 4.2 repräsentiert. (Die Tabelle $\mathbf{T}_{(AB)}$ wird durch einen grauen Hintergrund hervorgehoben.)

Tabelle $\begin{bmatrix} b_1a_1 & b_1a_2 \\ b_2a_1 & b_2a_2 \end{bmatrix}$ auf.

In $\mathbf{T}_{(A_1B_1)}$ ist andererseits auch alle Information gespeichert, die an die nachfolgende Zeitscheibe weitergereicht werden muss, um eine exakte Inferenz zu ermöglichen. Bisher wurde noch keine Evidenz berücksichtigt, indem das Polynom oder $\mathbf{T}_{(A_1B_1)}$ instantiiert wurden. Um das Polynom oder $\mathbf{T}_{(A_1B_1)}$ zu vereinfachen, können alle $\lambda_{x_i}$ der nicht interessierenden Knoten auf 1 oder die gegebene Evidenz gesetzt werden.

Nun wollen wir uns Zeitscheibe 2 mit ihrer Vorgängerzeitscheibe 1 und ihrer Nachfolgerzeitscheibe 3 in Abbildung 4.1 anschauen. Wir wollen eine Prozedur bestimmen, die die Auswirkungen der vorhergehenden Zeitscheibe auf die nachfolgenden Zeitscheiben berücksichtigt und eine einfache und effiziente Erweiterung des alten Polynoms erlaubt, wenn eine neue Zeitscheibe angehängt wird.

Wie es schon mit der Prozedur für die erste Zeitscheibe der Fall gewesen ist, lassen sich die Knoten des Zustandsentwicklungsmodells nicht herausmarginalisieren, solange die nachfolgende Zeitscheibe nicht angehängt wurde. In unserem Beispiel sind dies die Knoten $A_2$ und $B_2$.

Wir erhalten als Prozedur für die Vorwärtspropagierung für die zweite Zeit-

scheibe in unserem Beispiel in Abbildung 4.1 das folgende:

$$\begin{aligned}&\mathrm{fwd}_2(\mathbf{T}_{(A_1B_1)}) = \\ &\Big(\sum_{C_2}\theta_{(C_2|B_2)}\ \lambda_{C_2}\Big)\ \Big(\sum_{D_2}\theta_{(D_2|B_2)}\ \lambda_{D_2}\Big) \\ &\Big(\sum_{A_1}\theta_{(A_2|A_1)}\ \lambda_{A_2}\ \Big(\sum_{B_1}\theta_{(B_2|A_2,A_1,B_1)}\ \lambda_{B_2}\ \mathbf{T}_{(A_1B_1)}\Big)\Big) = \mathbf{T}_{(A_2B_2)}\ .\end{aligned}$$

Ein arithmetischer Schaltkreis, der diese Prozedur repräsentiert, ist in Abbildung 4.4 abgebildet. In dieser Abbildung sind die Hypothesen und Knoten der vorhergehenden Zeitscheibe unterstrichen und die Indizes, die die Zeitscheibennummer angeben, wurden ausgelassen. Um Speicher zu sparen, können beide arithmetische Schaltkreise, die in den Abbildungen 4.3 und 4.4 gezeigt werden, verschmolzen werden, um gemeinsame Strukturen miteinander zu teilen, wie z. B. $(\sum_C\ \theta_{(C|B)}\lambda_C)$.

Man sieht leicht, dass die Prozeduren für die nachfolgenden Zeitscheiben alle identisch sind und die Prozeduren für die Zeitscheiben $i$ (mit $i \geq 1$) wie folgt verallgemeinert werden können:

$$\begin{aligned}&\mathrm{fwd}_i(\mathbf{T}_{(A_{i-1}B_{i-1})}) = \\ &\Big(\sum_{C_i}\theta_{(C_i|B_i)}\ \lambda_{C_i}\Big)\ \Big(\sum_{D_i}\theta_{(D_i|B_i)}\ \lambda_{D_i}\Big) \\ &\Big(\sum_{A_{i-1}}\theta_{(A_i|A_{i-1})}\ \lambda_{A_i}\ \Big(\sum_{B_{i-1}}\theta_{(B_i|A_i,A_{i-1},B_{i-1})}\ \lambda_{B_i}\ \mathbf{T}_{(A_{i-1}B_{i-1})}\Big)\Big) = \mathbf{T}_{(A_iB_i)}\ .\end{aligned}$$

Im Fall $i = 1$ verschwinden die Variablen mit dem Index 0, und wir erhalten:

$$\mathrm{fwd}_1(\mathbf{T}_{(A_0B_0)}) = \mathrm{fwd}_1(1).$$

Das Polynom für Zeitscheibe 1 bis $L$ sieht wie folgt aus:

$$\begin{aligned}\mathscr{F}_{\mathrm{elim}(\{X\in\mathscr{N}\})}(\lambda_{X_i},\theta_{(X_i|\mathbf{pa}(X_i))}) &= \sum_{\mathrm{elim}(\{A_L,B_L\})}(\mathrm{fwd}_L(\ldots\mathrm{fwd}_2(\mathrm{fwd}_1(1))\ldots)) \\ &= \sum_{\mathrm{elim}(\{A_L,B_L\})}(\mathrm{fwd}_L\circ\ldots\circ\mathrm{fwd}_2\circ\mathrm{fwd}_1(1))\ .\end{aligned}$$

Bevor wir zeigen, wie das Polynom bei konstantem Speicherverbrauch ausgewertet werden kann, möchten wir die bisher erzielten Ergebnisse in einer allgemeinen Schreibweise festhalten. Als allgemeine Prozedur für die Vorwärtspropagierung startend mit der ersten Zeitscheibe, erhalten wir:

$$\begin{aligned}\mathrm{fwd}_{\mathrm{init}} = &\sum_{\mathrm{elim}(\{X|X\in\mathrm{TS}(1)\setminus\mathbf{bs}(1)\})}\ \prod_{X\in\mathrm{TS}(1)}\theta_{(X|\mathbf{pa}(X))}\ \lambda_X = \\ &\sum_{\mathrm{elim}(\{X|X\in\mathrm{TS}(1)\setminus\mathbf{bs}(1)\})}\ \prod_{X\in\mathrm{TS}(1)}\theta_{(X|\mathbf{pa}(X))}\ \lambda_X\ 1 = \mathrm{fwd}_1(1)\ .\end{aligned}$$

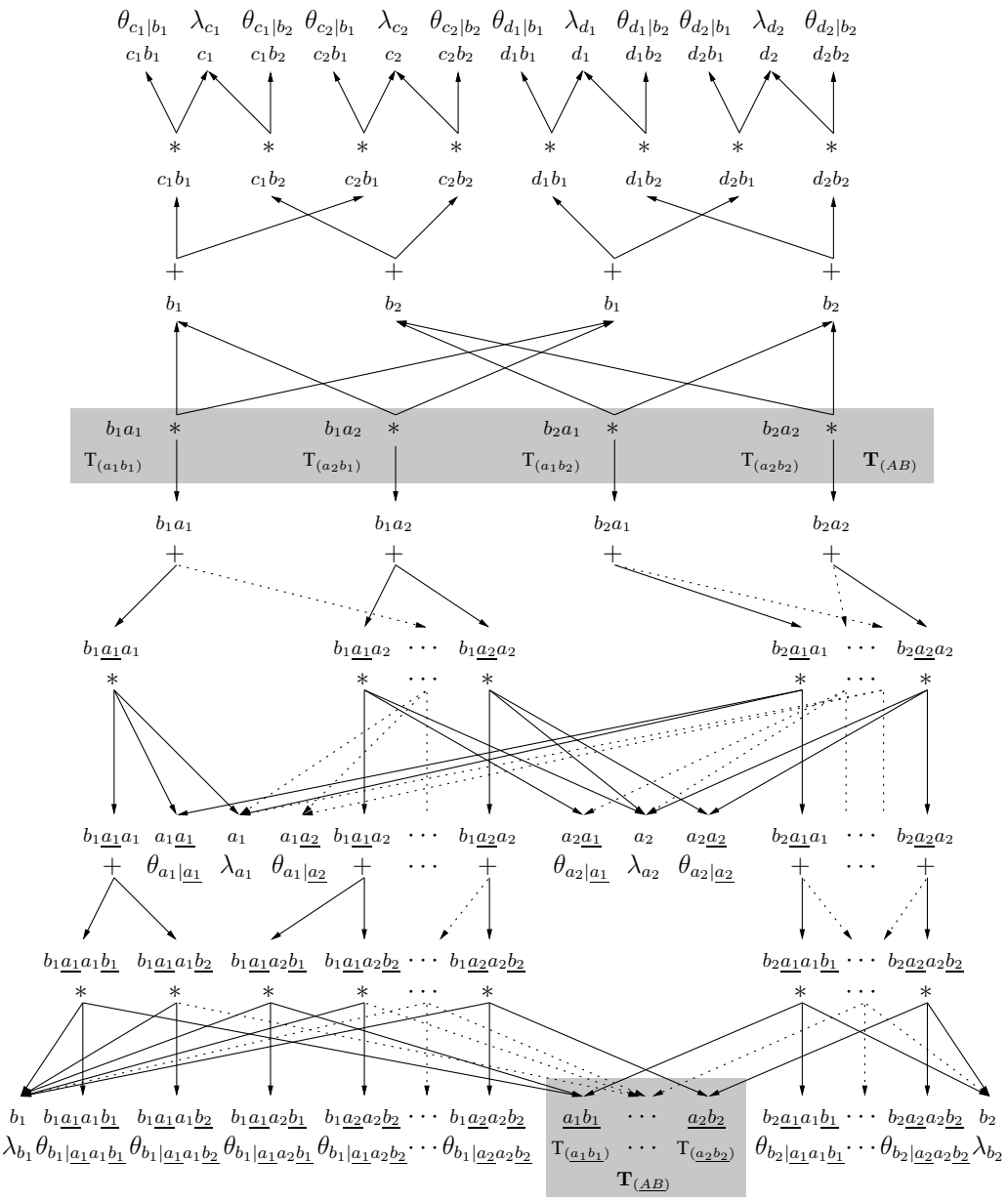


Abbildung 4.4: Ein arithmetischer Schaltkreis, der die Prozedur für die Vorwärtspropagierung für Zeitscheibenschema 2 in Abbildung 4.2 repräsentiert. (Hypothesen und Knoten der vorhergehenden Zeitscheibe sind unterstrichen. Die Tabellen $\mathbf{T}_{(\underline{AB})}$ and $\mathbf{T}_{(AB)}$ werden durch einen grauen Hintergrund hervorgehoben.)

TS$(1)$ bezeichnet die Knoten der ersten Zeitscheibe und **bs**$(1)$ bezeichnet die Menge der Knoten, die zum Zustandsentwicklungsmodell der ersten Zeitscheibe

gehören. Um die Darstellung zu vereinfachen (insbesondere für den nachfolgenden Algorithmus) haben wir hier die Schreibweise $\mathrm{fwd}_1(1)$ für $\mathrm{fwd}_{\mathrm{init}}$ eingeführt.

Das Polynom für die erste Zeitscheibe sieht in dieser allgemeinen Schreibweise dann wie folgt aus:

$$\begin{aligned}&\mathscr{F}_{\mathrm{elim}(\{X\in\mathscr{N}\})}(\lambda_{X_i},\theta_{(X_i|\mathbf{pa}(X_i))}) = \\ &\sum_{\mathrm{elim}(\mathbf{bs}(1))}(\mathrm{fwd}_{\mathrm{init}}) = \sum_{\mathrm{elim}(\mathbf{bs}(1))}(\mathrm{fwd}_1(1)) = \\ &\sum_{\mathrm{elim}(\mathbf{bs}(1))}\Big(\sum_{\mathrm{elim}(\{X|X\in\mathrm{TS}(1)\setminus\mathbf{bs}(1)\})}\prod_{X\in\mathrm{TS}(1)}\theta_{(X|\mathbf{pa}(X))}\ \lambda_X\ 1\Big)\ .\end{aligned}$$

Nun werden wir den Fall betrachten, wenn eine Zeitscheibe dem existierenden dynamischen Bayesschen Netz angehängt wird. Die Situation unterscheidet sich von der ersten Zeitscheibe wie folgt: Die Tabelle über alle Knoten, die noch nicht durch die gerade angewendete Prozedur herausmarginalisiert wurden, müssen nun an die allgemeine Prozedur für die $i$-te Zeitscheibe weitergereicht werden:

$$\mathrm{fwd}_i(\mathbf{T}) = \sum_{\mathrm{elim}(\{X|X\in\mathrm{TS}(i)\setminus\mathbf{bs}(i)\})}\ \sum_{\mathrm{elim}(\{X|X\in\mathbf{bs}(i-1)\})}\ \prod_{X\in\mathrm{TS}(i)}\theta_{(X|\mathbf{pa}(X))}\ \lambda_X\ \mathbf{T}\ .$$

$\mathbf{T}$ ist eine Tabelle über der Menge der Knoten $\{X \mid X \in \mathbf{bs}(i-1)\}$, die im allgemeinen durch die Prozedur der vorhergehenden Zeitscheibe berechnet wurde. Das Ergebnis der Funktion $\mathrm{fwd}_i$ ist eine Tabelle über der Menge der Knoten $\{X \mid X \in \mathbf{bs}(i)\}$.

Das Polynom für die Zeitscheiben 1 bis $L$ sieht wie folgt aus:

$$\begin{aligned}\mathscr{F}_{\mathrm{elim}(\{X\in\mathscr{N}\})}(\lambda_{X_i},\theta_{(X_i|\mathbf{pa}(X_i))}) = &\sum_{\mathrm{elim}(\mathbf{bs}(L))}(\mathrm{fwd}_L(\ldots\mathrm{fwd}_2(\mathrm{fwd}_1(1))\ldots)) = \\ &\sum_{\mathrm{elim}(\mathbf{bs}(L))}(\mathrm{fwd}_L\circ\ldots\circ\mathrm{fwd}_2\circ\mathrm{fwd}_1(1))\ .\end{aligned}$$

Die Auswertung des Polynoms bis zur Zeitscheibe $L$ lässt sich wie folgt mit konstantem Speicherverbrauch realisieren:

1. $\mathbf{T} = 1$;
2. For $i = 1$ to $L$:
   $\mathbf{T}_{\mathrm{new}} = \mathrm{fwd}_i(\mathbf{T})$; $\mathbf{T} = \mathbf{T}_{\mathrm{new}}$;
3. $\mathscr{F}(\mathbf{e}_1,\ldots,\mathbf{e}_L) = \sum_{\mathrm{elim}(\mathbf{bs}(L))}(\mathbf{T})$;

In Schritt 1 wird für die erste Zeitscheibe der Wert 1 an die Tabelle $\mathbf{T}$ übergeben. In Schritt 2 wird eine Schleife von 1 bis $L$ durchlaufen. In der ersten Iteration wird eine neue Tabelle $\mathbf{T}_{\mathrm{new}}$ durch die erste Zeitscheibe berechnet, d. h.

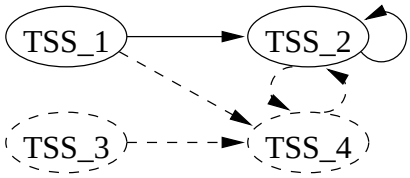



Abbildung 4.5: Gerichteter Graph, der bestimmt, in welcher Reihenfolge Zeitscheibenschemata instantiiert werden können.

$\mathbf{T}_{\text{new}} = \text{fwd}_1(1)$. In der $i$-ten Iteration wird $\mathbf{T}_{\text{new}} = \text{fwd}_i(\mathbf{T})$ berechnet, wobei die Auswirkungen der vorhergehenden Zeitscheiben berücksichtigt werden. Während der Berechnungen in der Schleife werden schon vorhandene Variablen effizient wiederverwendet. Insbesondere werden für die Bestimmung von $\text{fwd}_i(\mathbf{T})$ die dazugehörigen allgemeinen Prozeduren angewendet. Infolgedessen wird in einer weiteren Iteration kein weiterer Speicher vergeben, wenn sämtliche Evidenz-Vektoren der aktuellen Zeitscheibe mit Zahlenwerten belegt werden. In Schritt 3 wird schließlich das Polynom berechnet.

Das Polynom $\mathscr{F}(\mathbf{e}_1, \ldots, \mathbf{e}_L)$ kann sowohl für die Berechnung der BEL-Werte der Knoten in Zeitscheibe $t$ als auch für andere Berechnungen verwendet werden, wie z. B. eine Sensitivitätsanalyse, wie es kurz schon in Abschnitt 2.4.1 erwähnt und mehr im Detail in [Darwiche 00] diskutiert wird.

Die Prozedur kann ebenso für dynamische Bayessche Netze verallgemeinert werden, bei denen mehr als ein Zeitscheibenschema für die Instantiierung einer Zeitscheibe zur Verfügung steht. Ein gerichteter Graph kann dazu verwendet werden, um die Abhängigkeiten zu modellieren, in welcher Reihenfolge die Zeitscheibenschemata instantiiert werden können, so dass die Anzahl der Prozeduren minimiert werden können. In Abbildung 4.5 geben die durchgezogenen Linien den gerichteten Graphen an, der die Abhängigkeiten unter den Zeitscheibenschemata in Abbildung 4.2 für das Beispiel eines dynamischen Bayesschen Netzes in Abbildung 4.1 angibt. Der gesamte gerichtete Graph (mit gestrichelten und durchgezogenen Linien) in Abbildung 4.5 umfasst zwei Zeitscheibenschemata (1 und 3), die als erste Zeitscheibe für das dynamische Bayessche Netz instantiiert werden können, und zwei andere Zeitscheibenschemata (2 und 4), die als nachfolgende Zeitscheiben im dynamischen Bayesschen Netz instantiiert werden können. Nach Zeitscheibenschema 1 kann entweder Zeitscheibenschema 2 oder Zeitscheibenschema 4 als nachfolgende Zeitscheibe instantiiert werden. Nach dem Zeitscheibenschema 3 kann nur das Zeitscheibenschema 4 als nachfolgende Zeitscheibe instantiiert werden.

Die Anzahl der allgemeinen Prozeduren für ein Zeitscheibenschema hängt von der Anzahl der möglichen vorhergehenden und nachfolgenden Zeitscheibenschemata ab.

Im folgenden bezeichnen wir mit $\text{strct}(i)$ die Knoten von Zeitscheibenschema $i$, und mit $\text{tss}(t)$ bezeichnen wir die Knoten des Zeitscheibenschemas, das zum Zeitpunkt $t$ instantiiert wurde. In unserem Beispiel für ein dynamisches Bayessches Netz (siehe Abb. 4.2) haben wir $\text{tss}(1) = \text{strct}(1)$ und $\text{tss}(t) = \text{strct}(2)\ \forall t \geq 2$. Das Zustandsentwicklungsmodell hängt nicht nur von der aktuellen Zeitscheibe sondern auch von der nachfolgenden Zeitscheibe ab. So bezeichnen wir mit $\mathbf{bs}(\text{tss}(i), \text{tss}(i+1))$ die Menge der Knoten, die zum Zustandsentwicklungsmodell der Zeitscheibe $i$ gehören, wenn Zeitscheibe $i+1$ nachfolgt. Die Menge der Knoten, die zum Zustandsentwicklungsmodell des Zeitscheibenschemas $i$ gehören, wenn das Zeitscheibenschema $j$ als nachfolgende Zeitscheibe instantiiert wird, wird mit $\mathbf{bs}(\text{strct}(i), \text{strct}(j))$ bezeichnet.

Die allgemeine Prozedur für die Vorwärtspropagierung für ein Zeitscheibenschema, das die erste Zeitscheibe instantiiert, lautet somit wie folgt:

$$\text{fwd}_{\text{tss}(1),\text{tss}(2)} = \sum_{\text{elim}(\{X \mid X \in \text{tss}(1) \setminus \mathbf{bs}(\text{tss}(1),\text{tss}(2))\})} \prod_{X \in \text{tss}(1)} \theta_{(X \mid \mathbf{pa}(X))}\ \lambda_X\ .$$

Die allgemeinen Prozeduren für Zeitscheibenschema 1 in Abbildung 4.5 lauten $\text{fwd}_{\text{strct}(1),\text{strct}(2)}$ und $\text{fwd}_{\text{strct}(1),\text{strct}(4)}$. Die Prozedur für das Zeitscheibenschema 3 lautet $\text{fwd}_{\text{strct}(3),\text{strct}(4)}$.

Nun wollen wir den Fall betrachten, in dem eine Zeitscheibe an das existierende dynamische Bayessche Netz angehängt werden soll. Die Tabelle über alle Knoten, die durch die gerade angewendete Prozedur nicht herausmarginalisiert werden konnten, wird nun an die allgemeine Prozedur für die $i$-te Zeitscheibe weitergereicht:

$$\text{fwd}_{\text{tss}(i-1)\to\text{tss}(i),\text{tss}(i+1)}(\mathbf{T}) = \sum_{\text{elim}(\{X \mid X \in \text{tss}(i) \setminus \mathbf{bs}(\text{tss}(i),\text{tss}(i+1))\})} \sum_{\text{elim}(\{X \mid X \in \mathbf{bs}(\text{tss}(i-1),\text{tss}(i))\})} \prod_{X \in \text{tss}(i)} \theta_{(X \mid \mathbf{pa}(X))}\ \lambda_X\ \mathbf{T}\ .$$

$\mathbf{T}$ ist eine Tabelle über der Menge der Knoten $\{X \mid X \in \mathbf{bs}(\text{tss}(i-1), \text{tss}(i))\}$, die im allgemeinen durch die Prozedur der vorhergehenden Zeitscheibe berechnet wurde. Das Ergebnis der Funktion fwd ist eine Tabelle über den Knoten $\{X \mid X \in \mathbf{bs}(\text{tss}(i), \text{tss}(i+1))\}$. Die allgemeinen Prozeduren für das Zeitscheibenschema 2 in Abbildung 4.5 sind

$$\begin{aligned}
&\text{fwd}_{\text{strct}(1)\to\text{strct}(2),\text{strct}(2)}, \text{fwd}_{\text{strct}(1)\to\text{strct}(2),\text{strct}(4)},\\
&\text{fwd}_{\text{strct}(2)\to\text{strct}(2),\text{strct}(2)}, \text{fwd}_{\text{strct}(2)\to\text{strct}(2),\text{strct}(4)},\\
&\text{fwd}_{\text{strct}(4)\to\text{strct}(2),\text{strct}(2)}, \text{fwd}_{\text{strct}(4)\to\text{strct}(2),\text{strct}(4)},
\end{aligned}$$

und die allgemeinen Prozeduren für das Zeitscheibenschema 4 in Abbildung 4.5 sind

$$\text{fwd}_{\text{strct}(1)\to\text{strct}(4),\text{strct}(2)}, \text{fwd}_{\text{strct}(2)\to\text{strct}(4),\text{strct}(2)} \text{ und } \text{fwd}_{\text{strct}(3)\to\text{strct}(4),\text{strct}(2)} .$$

Wiederum führen wir hier die Schreibweise

$$\text{fwd}_{\text{strct}(0)\to\text{strct}(1),\text{strct}(2)}(1) \text{ für } \text{fwd}_{\text{tss}(1),\text{tss}(2)}$$

ein, um die Darstellung zu vereinfachen (insbesondere für den folgenden Algorithmus in Abschnitt 4.3).

Das Polynom für die Zeitscheiben 1 bis $L$ (mit $\text{tss}(0) = \emptyset$) lautet demnach wie folgt:

$$\begin{aligned}\mathscr{F}_{\text{elim}(\{X\in\mathcal{N}\})}(\lambda_{x_i}, \theta_{(x_i|\mathbf{pa}(x_i))}) = \\ \sum_{\text{elim}(\mathbf{bs}(\text{tss}(L),\text{tss}(L+1)))} (\text{fwd}_{\text{tss}(L-1)\to\text{tss}(L),\text{tss}(L+1)}(\ldots \\ \text{fwd}_{\text{tss}(1)\to\text{tss}(2),\text{tss}(3)}(\text{fwd}_{\text{tss}(0)\to\text{tss}(1),\text{tss}(2)}(1))\ldots)) .\end{aligned}$$

## 4.2 Rückwärtspropagierung

Mit den Prozeduren für die Vorwärtspropagierung kann der Einfluss der Evidenz in neuen Zeitscheiben auch zu älteren Zeitscheiben gebracht werden. Zuerst muss ein Aufwärtsdurchgang von den älteren Zeitscheiben bis zu den neuen Zeitscheiben durchzuführen, deren Evidenz in Betracht gezogen werden soll. Dann ist ein Abwärtsdurchgang von den neueren Zeitscheiben zu den älteren Zeitscheiben notwendig. Zeitscheiben können allerdings erst nach dem Abwärtsdurchgang aufgerollt werden. Somit ist eine Auswertung bei konstantem Speicherverbrauch nur mit den Prozeduren für die Vorwärtspropagierung nicht möglich.

Schauen wir uns die Zeitscheiben 2 und 3 in Abbildung 4.1 an. Wir erhalten für die letzte Zeitscheibe als Prozedur für die Rückwärtspropagierung:

$$\begin{aligned}\text{bwd}_{\text{tss}(3)\to\text{tss}(2)} = \sum_{A_3,B_3} \theta_{(A_3|A_2)}\lambda_{A_3}\ \theta_{(B_3|A_3,A_2,B_2)}\lambda_{B_3} \\ \left(\sum_{C_3} \theta_{(C_3|B_3)}\lambda_{C_3}\right)\left(\sum_{D_3} \theta_{(D_3|B_3)}\lambda_{D_3}\right) .\end{aligned}$$

Im Gegensatz zu den allgemeinen Prozeduren für die Vorwärtspropagierung, marginalisieren wir alle Knoten heraus, die zur betrachteten Zeitscheiben gehören. Das Ergebnis ist eine Tabelle **T** über dem kartesischen Produkt der Hypothesen der Elternknoten, die zur vorhergehenden Zeitscheibe gehören—in unserem Beispiel sind dies die Knoten $A_2$ und $B_2$.

Nun wollen wir eine Prozedur bestimmen, die die Auswirkungen der folgenden Zeitscheibe auf die vorhergehende Zeitscheibe berücksichtigt. Wie es auch für die Prozedur für die letzte Zeitscheibe der Fall war, können wir alle Knoten herausmarginalisieren, die zur betrachteten Zeitscheibe gehören. Um jedoch Rechenzeit- und Speicherbedarf so gering wie möglich zu halten, marginalisieren wir zuerst die Knoten heraus, die nicht zum Zustandsentwicklungsmodell dieser Zeitscheibe gehören, d. h. $\mathbf{bs}(\mathrm{tss}(2),\mathrm{tss}(3))$, und dann multiplizieren wir die Tabelle $\mathbf{T}$ mit den Knoten des Zustandsentwicklungsmodells dieser Zeitscheibe, um im Anschluss daran die Knoten des Zustandsentwicklungsmodells dieser Zeitscheibe herauszumarginalisieren.

Wir erhalten als Prozedur für die Rückwärtspropagierung für die zweite Zeitscheibe in unserem Beispiel in Abbildung 4.1 das folgende:

$$\begin{aligned}&\mathrm{bwd}_{\mathrm{tss}(3),\mathrm{tss}(2)\rightarrow\mathrm{tss}(1)}(\mathbf{T}_{(A_2B_2)}) = \\ &\Big(\sum_{A_2}\theta_{(A_2|A_1)}\ \lambda_{A_2}\ \Big(\sum_{B_2}\theta_{(B_2|A_2,A_1,B_1)}\ \lambda_{B_2}\ \mathbf{T}_{(A_2B_2)} \\ &\quad\Big(\sum_{C_2}\theta_{(C_2|B_2)}\ \lambda_{C_2}\Big)\ \Big(\sum_{D_2}\theta_{(D_2|B_2)}\ \lambda_{D_2}\Big)\Big)\Big) = \mathbf{T}_{(A_1B_1)}\ .\end{aligned}$$

Wir wollen nun in einer allgemeinen Schreibweise das bisherige Ergebnis festhalten. Wir lassen dabei die allgemeine Prozedur zur Rückwärtspropagierung von der letzten Zeitscheibe aus, das sie nur ein Spezialfall der allgemeinen Prozedur für die Rückwärtspropagierung von der $i$-ten Zeitscheibe ist ($1 \leq i \leq L$):

$$\begin{aligned}&\mathrm{bwd}_{\mathrm{tss}(i+1),\mathrm{tss}(i)\rightarrow\mathrm{tss}(i-1)}(\mathbf{T}) = \\ &\sum_{\mathrm{elim}(\{X|X\in\mathbf{bs}(\mathrm{tss}(i),\mathrm{tss}(i+1))\})}\ \prod_{X\in\mathbf{bs}(\mathrm{tss}(i),\mathrm{tss}(i+1))}\theta_{(X|\mathbf{pa}(X))}\lambda_X\ \mathbf{T} \\ &\Big(\sum_{\mathrm{elim}(\{X|X\in\mathrm{tss}(i)\setminus\mathbf{bs}(\mathrm{tss}(i),\mathrm{tss}(i+1))\})}\ \prod_{X\in\mathrm{tss}(i)\setminus\mathbf{bs}(\mathrm{tss}(i),\mathrm{tss}(i+1))}\theta_{(X|\mathbf{pa}(X))}\ \lambda_X\Big)\ .\end{aligned}$$

Es gilt das folgende:

- im Fall $i = L$: $\mathrm{tss}(L+1) = \emptyset$ und $\mathbf{T} = \mathbf{1}$;
- im Fall $i = 1$: $\mathrm{tss}(0) = \emptyset$.

Im nächsten Abschnitt zeigen wir, wie sich die Vorwärts- mit der Rückwärtspropagierung kombinieren lassen.

## 4.3 Kombinierte Vorwärts- und Rückwärtspropagierung

Wenn wir am BEL-Wert von Knoten in Zeitscheibe $t$ eines dynamischen Bayesschen Netzes mit Zeitscheiben 1 bis $L$ (mit $1 \leq t \leq L$) interessiert sind (siehe Abbildung 4.6), dann können wir die Prozeduren für die Vorwärts- und Rückwärtspropagierung miteinander kombinieren: Wir multiplizieren die Prozeduren für das Polynom zur Vorwärtspropagierung von Zeitscheibe 1 bis $t$ mit den Prozeduren für das Polynom zur Rückwärtspropagierung von Zeitscheibe $L$ bis $t+1$ und marginalisieren dann die Knoten des Zustandsentwicklungsmodells $\mathbf{bs}(\mathrm{tss}(t), \mathrm{tss}(t+1))$ heraus.

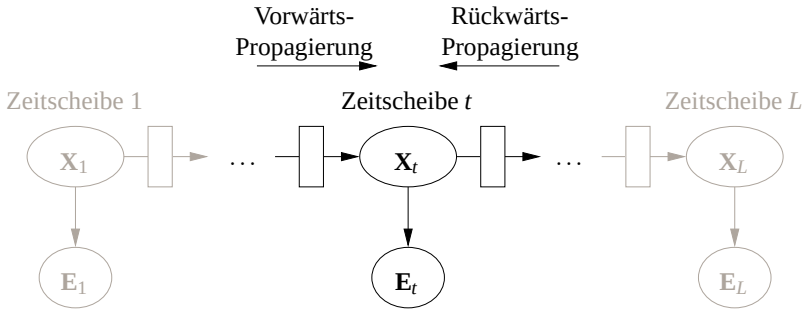



Abbildung 4.6: Dynamisches Bayessches Netz mit $L$ Zeitscheiben.

Somit sieht das Polynom, das für den Zeitpunkt $t$ mit Evidenz $\mathbf{e}_1, \ldots, \mathbf{e}_t, \ldots, \mathbf{e}_L$ ausgewertet werden soll, wie folgt aus:

$$
\begin{aligned}
&\mathscr{F}_t(\mathbf{e}_1, \ldots, \mathbf{e}_t, \ldots, \mathbf{e}_L) = \\
&\quad \sum_{\mathrm{elim}(\mathbf{bs}(\mathrm{tss}(t),\mathrm{tss}(t+1)))} ( \\
&\qquad \mathrm{fwd}_{\mathrm{tss}(t-1)\to\mathrm{tss}(t),\mathrm{tss}(t+1)}( \\
&\qquad\quad \mathrm{fwd}_{\mathrm{tss}(t-2)\to\mathrm{tss}(t-1),\mathrm{tss}(t)}(\ldots \mathrm{fwd}_{\mathrm{tss}(0)\to\mathrm{tss}(1),\mathrm{tss}(2)}(\mathbf{1})\ldots)) \\
&\qquad \mathrm{bwd}_{\mathrm{tss}(t+2),\mathrm{tss}(t+1)\to\mathrm{tss}(t)}( \\
&\qquad\quad \mathrm{bwd}_{\mathrm{tss}(t+3),\mathrm{tss}(t+2)\to\mathrm{tss}(t+1)}(\ldots \mathrm{bwd}_{\mathrm{tss}(L+1),\mathrm{tss}(L)\to\mathrm{tss}(L-1)}(\mathbf{1})\ldots))) \,.
\end{aligned}
$$

Die Auswertung des Polynoms für den Zeitpunkt $t$ bei konstantem Speicherverbrauch erreicht man wie folgt:

1. $\mathbf{F} = 1$;
2. For $i = 1$ to $t$:
   $\mathbf{F}_{\mathrm{neu}} = \mathrm{fwd}_{\mathrm{tss}(i-1)\to\mathrm{tss}(i),\mathrm{tss}(i+1)}(\mathbf{F})$; $\mathbf{F} = \mathbf{F}_{\mathrm{neu}}$;

3. $\mathbf{B} = \mathbf{1}$;
4. For $i = L$ downto $t+1$:
   $\mathbf{B}_{\text{neu}} = \text{bwd}_{\text{tss}(i+1),\text{tss}(i)\rightarrow\text{tss}(i-1)}(\mathbf{B})$; $\mathbf{B} = \mathbf{B}_{\text{neu}}$;
5. $\mathscr{F}_t(\mathbf{e}_1,\ldots,\mathbf{e}_t,\ldots,\mathbf{e}_L) = \sum_{\text{elim}(\mathbf{bs}(\text{tss}(t),\text{tss}(t+1)))}(\mathbf{F} * \mathbf{B})$;

Es gelten dieselben Bemerkungen wie für den vorhergehenden Algorithmus. Wir bezeichnen hier die Tabelle im Fall für die Vorwärtspropagierung mit **F** und im Fall für die Rückwärtspropagierung mit **B**.

## 4.4 Approximation

Die Prozeduren können angepasst werden, so dass wir die Tabelle **T** nicht berechnen müssen, die die Information ohne Approximation weiterleitet. Stattdessen können mehrere kleinere Tabellen berechnet werden, die weniger komplex sind als die Gesamttabelle **T** und deren Produkt die Gesamttabelle approximiert wiedergibt. [Boyen & Koller 99] beschreibt, wie die Gesamttabelle in kleinere Tabelle mit minimalem Informationsverlust aufgespalten werden kann. Sie spezifizieren mehrere Kriterien (*weak interaction*, *conditional weak interaction* und *sparse interaction*) anhand derer es möglich ist, die Menge der Knoten in mehrere kleinere Mengen zu partitionieren, um auf diese Weise den entstehenden Informationsverlust so gering wie möglich zu halten.

In [Boyen & Koller 98] wird gezeigt, dass der Fehler im Zustandsentwicklungsmodell mit Fortschreiten des Prozesses exponentiell kontrahiert, so dass bei mehreren Approximationen der Fehler auf unbestimmte Zeit beschränkt bleibt.

Wir zeigen nun, wie diese Approximation in den vorgeschlagenen Algorithmus eingebracht werden kann. In Abbildung 4.7(i) wird schematisch die allgemeine Prozedur zur Vorwärtspropagierung in der $i$-ten Zeitscheibe ohne Approximation dargestellt ($\text{fwd}_{\text{tss}(i-1)\rightarrow\text{tss}(i),\text{tss}(i+1)}$). Um der Klarheit willen, haben wir die Schreibweise $\mathbf{bs}(\text{tss}(i-1),\text{tss}(i))$ durch die kürzere Schreibweise $\mathbf{bs}(i-1,i)$ ersetzt, die wir auch im folgenden beibehalten werden. Die Tabelle $\mathbf{T}_{(\mathbf{bs}(i-1,i))}$ wurde durch die allgemeine Prozedur zur Vorwärtspropagierung der vorhergehenden Zeitscheibe berechnet. Diese Tabelle dient nun als Eingabe in die allgemeine Prozedur zur Vorwärtspropagierung für die $i$-te Zeitscheibe, die die Tabelle $\mathbf{T}_{(\mathbf{bs}(i,i+1))}$ als Ergebnis hervorbringt.

In Abbildung 4.7(ii) wird schematisch die allgemeine Prozedur zur Vorwärtspropagierung in der $i$-ten Zeitscheibe mit Approximation dargestellt. Die Tabelle $\mathbf{T}_{(\mathbf{bs}(i-1,i))}$ wurde in die Tabellen $\mathbf{T}_{(\mathbf{bs}_1(i-1,i))},\ldots,\mathbf{T}_{(\mathbf{bs}_m(i-1,i))}$ aufgespalten, die als Eingabe in die allgemeine Prozedur mit Approximation dient. Diese Prozedur bringt als Ergebnis die Tabellen $\mathbf{T}_{(\mathbf{bs}_1(i,i+1))},\ldots,\mathbf{T}_{(\mathbf{bs}_n(i,i+1))}$ hervor.

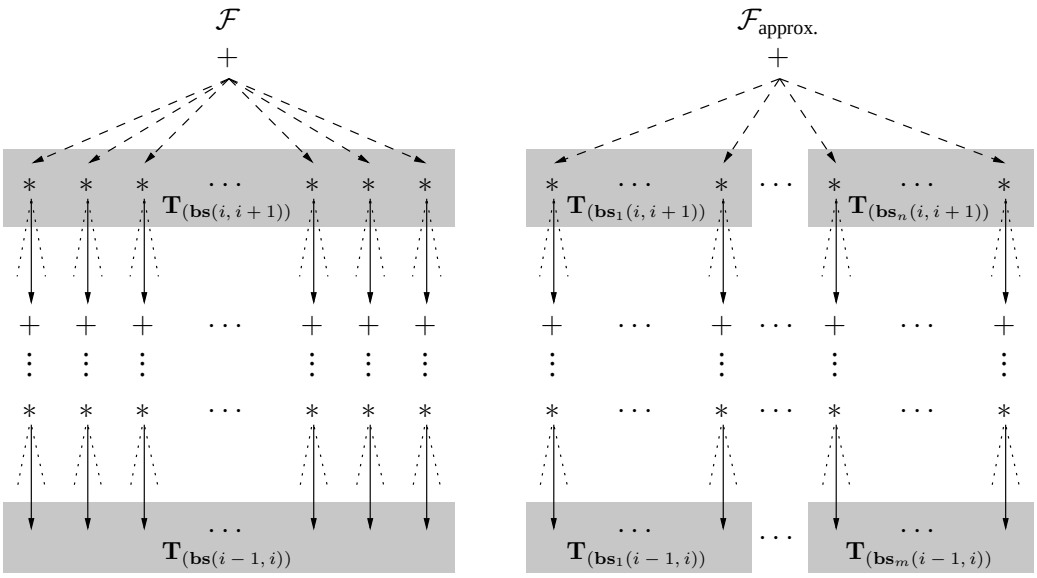



Abbildung 4.7: Zwei arithmetische Schaltkreise, die die Prozeduren zur Vorwärtspropagierung (durchgezogene und gepunktete Linien) und das Polynom (gepunktete, gestrichelte und durchgezogene Linien) repräsentieren – entsprechend (i) ohne und (ii) mit Approximation. Für eine bessere Darstellung wurde die Bezeichnung $\mathbf{bs}(\text{tss}(i-1),\text{tss}(i))$ durch $\mathbf{bs}(i-1,i)$ ersetzt.

Das hier vorgestellte Approximations-Verfahren für die Vorwärtspropagierung kann analog für die Rückwärtspropagierung angewendet werden. Die approximierten Prozeduren benötigen weniger Rechenspeicher und weniger Rechenleistung als die nicht-approximierten Prozeduren. Diese beiden Vorteile beschleunigen den gesamten Inferenzprozess.

## 4.5 Ausnutzung des erweiterten Ansatzes

In Abbildung 4.8 sind schematisch die verschiedenen Arten an Berechnungen dargestellt, die von einem System für eine an die Ressourcen angepasste Verarbeitung dynamischer Bayesscher Netze durchgeführt werden können. Es können dabei zwei Berechnungsarten unterschieden werden: solche die *offline* auf einem PC durchgeführt werden und solche, die *online* auf einem PDA durchgeführt werden, der vom Benutzer verwendet wird.

Zuerst wollen wir uns die Offline-Berechnungen anschauen. Wenn wir eine Menge an Zeitscheibenschemata, ihre Abhängigkeiten untereinander und die Menge der möglichen Evidenz- und Query-Knoten vorliegen haben, dann kön-

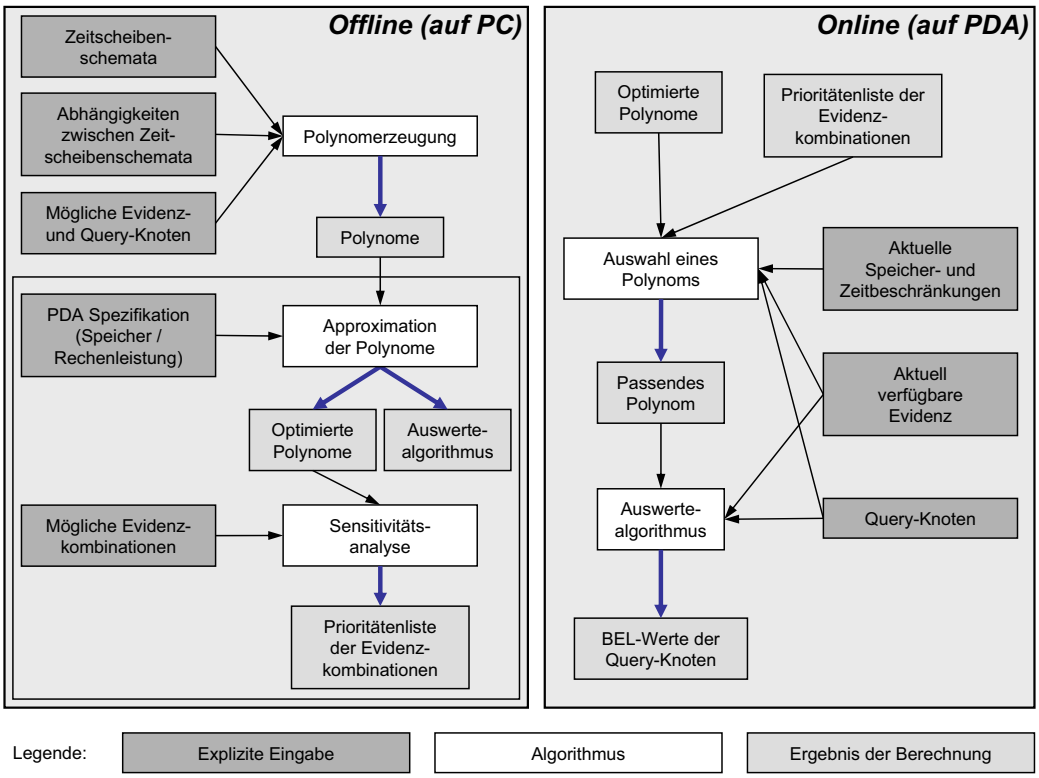



Abbildung 4.8: Online- und Offline-Berechnungen für eine an die Ressourcen angepasste Verarbeitung dynamischer Bayesscher Netze.

nen wir mit unserem oben vorgestellten Verfahren eine Menge an Polynomen berechnen. Die Abhängigkeiten der Zeitscheibenschemata untereinander sind in einem gerichteten azyklischen Graph wie dem in Abbildung 4.5 modelliert, in dem die Reihenfolge festgelegt ist, in der die Zeitscheibenschemata instantiiert werden können, so dass nicht für alle Zeitscheibenkombinationen ein Polynom berechnet wird.

Ist man ausschließlich an den BEL-Werten der dynamischen oder statischen Knoten (siehe Abschnitt 5.1.3) interessiert, so müssen wir lediglich einen Aufwärtsdurchgang im dazugehörigen arithmetischen Schaltkreis durchführen und die Tabelle **T** berechnen. Aus den Tabelleneinträgen erhalten wir schon die gewünschten BEL-Werte, so dass man in diesem Fall den Abwärtsdurchgang nicht durchführen muss.

In einem zweiten Schritt der Offline-Berechnungen kann die Menge der Polynome an die Ressourcenbeschränkungen eines PDAs durch Approximation angepasst werden. Die Eigenschaften des PDAs wie Prozessortyp, Taktfrequenz, Art des Hauptspeichers und ob Gleitkomma-Arithmetik oder nur Integer-Berechnun-

gen unterstützt werden, müssten von Hand eingegeben werden, z. B. aufgrund der Herstellerspezifikationen. Wir erhalten eine Menge an optimierten Polynomen und einen Auswertealgorithmus. Für die Menge der optimierten Polynome kann das System Sensitivitätsanalysen durchführen[1], um eine Rangfolge für die verschiedenen Kombinationen von Teilevidenz geordnet nach ihrem Einfluss auf bestimmte Query-Knoten zu bestimmen: Evidenzkombinationen mit keinem oder nur einem geringen Einfluss sollen nicht instantiiert werden.

Die Menge der optimierten Polynome, die Rangfolge und der Auswertealgorithmus werden dann auf den PDA übertragen. In Abhängigkeit der aktuellen Speicher- und Zeitbeschränkungen, der aktuell verfügbaren Evidenz und der ausgewählten Query-Knoten wird das Polynom aus der Menge der optimierten Polynome durch einen Algorithmus ausgewählt, der wie folgt arbeiten könnte. Nehmen wir an, dass bei der Interaktion des Benutzers mit dem PDA innerhalb eines gegebenen Zeitraums mehr Evidenz erfasst wird, als das dynamische Bayessche Netz verarbeiten kann. In diesem Fall muss eine Teilmenge der Evidenz zur Instantiierung verwendet werden; diese Teilmenge sollte diejenige sein, die den größten Einfluss auf die Query-Knoten gemäß der Rangfolge ausübt.

Kommt dies häufig vor, kann es eventuell daran liegen, dass die Rechenressourcen des PDAs zu optimistisch angegeben wurden; in diesem Fall wäre es besser die Angaben der Rechenressourcen zu überarbeiten, so dass die Polynome weiter approximiert werden müssen.

Während der Interaktion des Benutzers mit dem PDA könnte es für das System möglich sein, typische Eigenschaften festzustellen, beispielsweise wie der aktuelle Anwender das Gerät bedient. Beispielsweise könnte ein bestimmter Benutzer durchgängig die Spracherkennung vermeiden, indem er sich auf die Stifteingabe einschränkt. Wenn der PDA das nächste Mal mit dem PC verbunden wird, kann er das gesammelte Wissen über den Benutzer an den PC übermitteln. Das System auf dem PC kann dann sowohl die möglichen Evidenz-Kombinationen als auch die Annahmen über die Ressourceneinschränkungen des PDAs auffrischen und sowohl eine neue Approximation der Polynome und eine neue Rangfolge der Evidenz-Kombinationen berechnen. Wenn der Benutzer den PDA erneut einsetzt, hat sich das System den Bedürfnissen und Eigenschaften des Benutzers weiter angepasst.

Unser Verfahren weist kein Anytime- oder Anyspace-Verhalten auf. Das dynamische Bayessche Netz muss schon im Vorfeld an die Ressourcen-Einschränkungen des Gerätes angepasst werden, um so sicherstellen zu können, dass die Inferenz innerhalb der durch das Gerät vorgegebenen Grenzen berechnet werden kann, d. h. dass unser Verfahren *ressourcenadaptiert* ist (siehe [Wahlster & Tack 97]).

[1] Die Möglichkeit Sensitivitätsanalysen dieser Art durchführen zu können ist eine der allgemeinen Stärken von Darwiches differentiellem Ansatz.

## 4.6 Zusammenfassung

Wir haben gezeigt, wie Darwiches differentieller Ansatz zur Auswertung Bayesscher Netz (siehe Abschnitt 2.4) für dynamische Bayessche Netze erweitert werden kann. Es wurden die Prozeduren näher beschrieben, die man benötigt, um die maßgeblichen Polynome für beliebig große dynamische Bayessche Netze bestimmen zu können.

Wir können im dynamischen Bayesschen Netz eine Vorwärts- und Rückwärtspropagierung durchführen (und auch eine Kombination der beiden Verfahren). Wir können alte Zeitscheiben und andere überflüssige Netzstrukturen aufrollen und ermöglichen dabei die Auswertung des Polynoms bei konstantem Speicherverbrauch. Die anderen Vorzüge des differentiellen Ansatzes sind nun auch für dynamische Bayessche Netz nutzbar wie beispielsweise effiziente Verfahren zur Sensitivitätsanalyse (siehe [Darwiche 00, Darwiche 99]).

Weiterhin haben wir beschrieben, wie die vorgestellten Verfahren an die Ressourcen eines eingebetteten Systems angepasst werden können.

# Kapitel 5

# Dynamische Bayessche Netze *n*-ter Ordnung

Bei der Modellierung der dynamischen Bayesschen Netze ergab sich insbesondere im Zusammenhang mit der Benutzermodellierung und der Sensorverarbeitung die Fragestellung, wie der Modellierungprozess der dynamischen Bayesschen Netze vereinfacht und ihre Darstellung übersichtlicher gestaltet werden könnte. Dazu erweitern wir die dynamischen Bayesschen Netze um besondere Modellierungsstrukturen, die nicht nur in der Benutzermodellierung oder der Sensorverarbeitung als sinnvoll erscheinen. Wir beschreiben sie allgemeingültig, so dass sie auch in anderen Domänen eingesetzt werden können.

Man wird sehen, dass diese neuen Modellierungsstrukturen die direkte Anwendung der Rollup-Verfahren aus Abschnitt 3.3 verhindern, aber dass im allgemeinen durch graphentheoretische Umformungen der Rollup wieder ermöglicht wird.

Im folgenden Abschnitt stellen wir die neuen Modellierungsstrukturen vor, die im Zusammenhang der Benutzermodellierung mit dynamischen Bayesschen Netzen sinnvoll erscheinen.

## 5.1 Modellierungsstrukturen

Die neuen Modellierungsstrukturen wollen wir anhand des dynamischen Bayesschen Netzes motivieren, das in Abschnitt 7.1 vorgestellt wird.

Es handelt sich dabei um ein dynamisches Bayessches Netz, dessen Zeitscheibenschema aus zwei Bayesschen Netzen besteht. Eines der beiden Bayesschen Netze wurde auf der Basis von zwei Experimenten (siehe [Müller et al. 01] und [Kiefer 02]) für die Interpretation von Sprachsymptomen gelernt. Das andere Bay-

essches Netz wurde basierend auf einer umfassenden Literaturstudie für die Interpretation der Eigenschaften manuellem Eingabeverhaltens (siehe [Lindmark 00]) erstellt. Die Kombination dieser beiden Bayesschen Netze erlaubte es nun auf der Basis multimodaler Eingaben (hier sprachliche Äußerung und manuelle Eingabe) zu schließen (siehe Abschnitt 7.1 und Abbildung 7.5).

Wie man in Abbildung 7.5 sieht, wurden spezielle Knoten eingeführt, die mehrere Zeitscheiben beeinflussen, ohne dass sie einer gewissen Zeitscheibe zugeordnet sind. Mit diesen Knoten werden *typische* Benutzereigenschaften modelliert, die über die Laufzeit des Systems als unveränderlich angesehen werden können. Sie haben auf alle Zeitscheiben denselben Einfluss, und man bezeichnet sie daher als *statische Knoten*. Wie in dynamischen Bayesschen Netzen modellieren *dynamische Knoten* in den dynamischen Bayesschen Netzen *n*-ter Ordnung Sachverhalte, die sich über die Zeit entwickeln. Wir wollen jetzt aber auch Kanten zwischen nicht direkt benachbarten Zeitscheiben zulassen. Kanten zwischen benachbarten Zeitscheiben bezeichnen wir als temporale Kanten erster Ordnung. Und analog dazu bezeichnen wir mit temporalen Kanten *k*-ter Ordnung Kanten, die *k* Zeitscheiben überspringen. Dabei erlauben wir, dass im voraus nicht bekannt sein muss, wie viele Zeitscheiben tatsächlich übersprungen werden. Knoten, die Sachverhalten entsprechen, die nur innerhalb einer Zeitscheibe existieren, werden *temporäre Knoten* genannt. Kanten innerhalb einer Zeitscheibe sind Kanten 0-ter Ordnung.

Bei der Zusammenführung der beiden Bayesschen Netze wäre es wünschenswert gewesen, wenn beide Bayesschen Netze als einzelne Zeitscheiben repräsentiert blieben und sie eine übergeordnete Struktur vereinen würde. In Abbildung 7.5 sind die beiden Netzstrukturen, um die motorischen und sprachlichen Symptome zu verarbeiten, durch eine entsprechende Gruppierung angedeutet. Mithilfe eines sogenannten *Doppelschemas*, das beide Bayesschen Netze umfasst, ließen sich beide Netze strukturiert zusammenführen (siehe Abbildung 5.1), so dass man in den einzelnen Netzen Modellverbesserungen durchführen könnte, die dann auch direkt im Doppelschema Sprache-Handlung zur Verfügung stünden, ohne dass man erst das Netz wieder aus den beiden Teilnetzen neu von Hand zusammenfügen müsste. Das oben erwähnte Doppelschema dient zur Einschätzung der Ressourcenlage eines Anwenders aufgrund seiner multimodalen Eingaben (Sprache und Intra-Gestik) zur selben Zeit, wobei aber nicht zwingend immer beide Eingabemodalitäten vorkommen müssen. Die Knoten Time Pressure und Cognitive Load wären im gewissen Sinne dann statische Knoten innerhalb dieses Doppelschemas. Ein anderes Beispiel für ein mögliches Doppelschema wäre das Doppelschema Frage-Antwort, das die beiden Schemata Frage und Antwort zusammenführt, so dass eine Sinneinheit gebildet und das dynamische Bayessche Netz weiter gegliedert wird. Eine weiteres Beispiel einer Sinneinheit wäre das Doppelschema Instruktion-Handlung. Im Gegensatz zum Doppelschema Sprache-Hand-

lung werden bei den beiden zuletzt aufgeführten Doppelschemata die Evidenzen nicht gleichzeitig in beiden Zeitscheiben des Doppelschemas eingetragen (siehe [Schäfer & Weyrath 97]). Erst nach der Frage wird eine Antwort erwartet, und erst nach der Instruktion kann eine Handlung erfolgen.

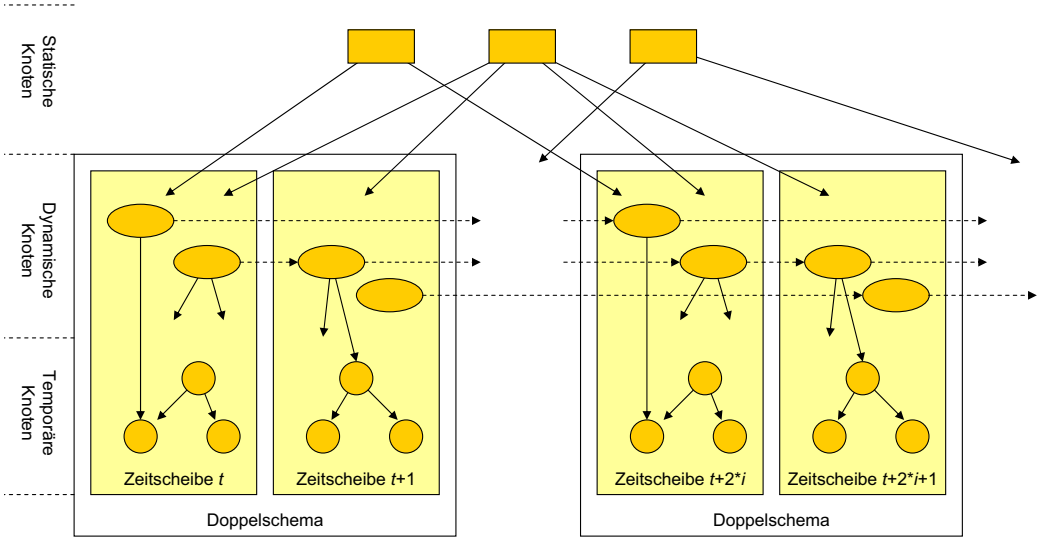



Abbildung 5.1: Prototypische Darstellung eines dynamischen Bayesschen Netzes *n*-ter Ordnung.

Wie man in den nachfolgenden Betrachtungen sehen wird, sind die folgenden Modellierungsstrukturen

- temporale Kanten *k*-ter Ordnung (zeitscheibenüberspringende Kanten), um Abhängigkeiten zu beschreiben, die über eine und mehr Zeitscheiben gehen,
- Statische Knoten, die typische Eigenschaften beschreiben, die sich während der Laufzeit des dynamischen Bayesschen Netzes nicht verändern und
- Doppelschemata, um Sinneinheiten von (eigentlich beliebig vielen) Zeitscheiben zu bilden

problematisch beim Aufrollen der Netze.

In den folgenden Abschnitten werden wir die verschiedenen Modellierungsstrukturen anhand eines generischen Schemas vorstellen und die aus den Modellierungsstrukturen resultierenden Schwierigkeiten für den Rollup diskutieren und abschließend zeigen, durch welche Maßnahmen man wieder einen Rollup mit den Mitteln aus Kapitel 3.3 ermöglichen kann.

Unser spezielles Ziel ist es, Umformungen vorzunehmen, die die neuen Modellierungsstrukturen in ein dynamisches Bayessches Netz übersetzen, das den Spezifikationen aus Abschnitt 3.2 entspricht.

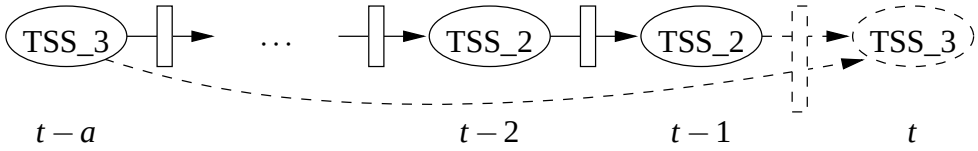
Im Abschnitt 3.3 haben wir gesehen, dass es ausreicht, immer nur maximal zwei Zeitscheiben des dynamischen Bayesschen Netzes in Bearbeitung zu haben. Die neuen Modellierungsstrukturen werden jetzt auf dieses Vorgehen hin überprüft.

Wie wir im folgenden sehen werden, lassen sich die Probleme, die durch die statischen Knoten und die Doppelschemata auftreten, auf die Probleme reduzieren, die durch die *zeitscheibenüberspringenden Kanten* verursacht werden. Deswegen besprechen wir nun als erstes die zeitscheibenüberspringenden Kanten.

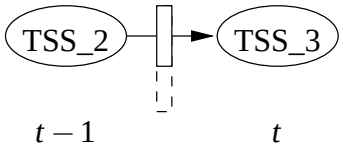
### 5.1.1 Zeitscheibenüberspringende Kanten

Im folgenden wird anhand eines generischen Schemas beschrieben, welche Probleme zeitscheibenüberspringende Kanten für dynamische Bayessche Netze und insbesondere für die Rollup-Verfahren aus Kapitel 3.3 aufwerfen. Man wird sehen, dass sich die Rollup-Verfahren aus Kapitel 3.3 nicht auf das dynamische Bayessche Netz anwenden lassen, wenn zeitscheibenüberspringende Kanten vorkommen, da durch sie die Markov-Eigenschaft (siehe Abschnitt 3.2) verletzt wird. Desweiteren wird gezeigt, wie sich die Markov-Eigenschaft durch das Einfügen von bestimmten Knoten an entsprechenden Stellen der zeitscheibenüberspringenden Kanten wieder herstellen lässt und sich die Rollup-Verfahren dadurch wieder anwenden lassen.

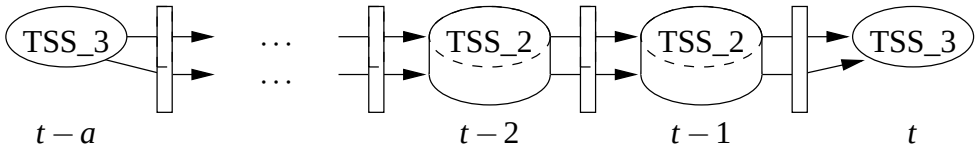
Es sei ein dynamisches Bayessches Netz zum Zeitpunkt $t-a-1$ gegeben, das nicht aufgerollt wird. Als Zeitscheibe $t-a$ werde jetzt das Zeitscheibenschema TSS_3 instantiiert. Bis zum Zeitpunkt $t-1$ werden dann sukzessive die einzelnen Zeitscheiben erzeugt. Für die Zeitscheiben kommen dabei verschiedene Schemata zum Tragen. Zu jedem Zeitpunkt $h$ ($1 \leq h \leq t-1$) seien die Schemata des dynamischen Bayesschen Netzes so instantiiert, dass das jeweilige Zustandsentwicklungsmodell der Zeitscheibe $h$ nur von der vorhergehenden Zeitscheibe $h-1$ bestimmt ist. Dieses wird im generischen Schema durch gerichtete Kanten angedeutet, die durch einen Kasten gehen. Die Zeitscheiben erfüllen somit die Markov-Eigenschaft für dynamische Bayessche Netze. In Abbildung 5.2(a) ist ein Schnappschuss des resultierenden dynamischen Bayesschen Netzes zum Zeitpunkt $t-1$ (nur die durchgezogenen Linien) dargestellt. Es werden die Zeitscheiben $t-a$ bis $t-1$ mit den dazwischenliegenden Zustandsentwicklungsmodellen schematisch dargestellt. Die Zeitscheiben bis vor dem Zeitpunkt $t-a$ sind ausgeblendet, da sie nichts zum Sachverhalt beitragen. Zum Zeitpunkt $t-a$ sei das Zeitscheibenschema TSS_3 zum letzten Mal instantiiert worden (Dies ist wichtig für die Instantiierung der Zeitscheibe $t$.). Danach seien bis zum Zeitpunkt $t-1$ nur




(a) Dynamisches Bayessches Netz ohne Rollup zum Zeitpunkt $t-1$ (durchgezogene Linien) und zum Zeitpunkt $t$ (durchgezogene und gestrichelte Linien). Zum ersten Mal nach Zeitpunkt $t-a$ wird wieder TSS_3 instantiiert und dadurch zeitscheibenüberspringende Kanten induziert.




(b) Dynamisches Bayessches Netz mit Rollup zum Zeitpunkt $t$. Es können keine zeitscheibenüberspringende Kanten instantiiert werden, so dass die Inferenz nicht stimmt.




(c) Korrigiertes dynamisches Bayessches Netz zum Zeitpunkt $t$. Durch Zwischenknoten werden zeitscheibenüberspringende Kanten in die Zeitscheiben eingebettet, so dass ein Rollup zu jeder Zeit durchführbar ist.

Abbildung 5.2: Das Problem der zeitscheibenüberspringenden Kanten und wie man ihm begegnet.

noch Zeitscheibenschemata TSS_2 $\neq$ TSS_3 als Zeitscheiben eingefügt worden. Zum Zeitpunkt $t-1$ besteht das dynamische Bayessche Netz somit aus $t$ Zeitscheiben.

Zum Zeitpunkt $t$ wird zum ersten Mal nach dem Zeitpunkt $t-a$ wieder das Zeitscheibenschema TSS_3 als Zeitscheibe instantiiert. In diesem Zeitscheibenschema ist ein Knoten enthalten, der eine Zufallsvariable repräsentiert, die sich selbst beeinflusst. Das Zustandsentwicklungsmodell für TSS_3 zum Zeitpunkt $t$ ist nicht mehr nur von Zeitscheibe $t-1$ sondern auch von Zeitscheibe $t-a$ abhängig, in der auch das Zeitscheibenschema TSS_3 als Zeitscheibe instantiiert wurde. Es wird eine *zeitscheibenüberspringende Kante* von Zeitscheibe $t-a$ zu Zeitscheibe $t$ induziert. Dies ist in Abbildung 5.2(a) durch einen gestrichelten gerichteten Pfeil von Zeitscheibe $t-a$ zu Zeitscheibe $t$ dargestellt. Die Zustandsentwicklungsmodelle und die Schemata dazwischen ändern sich dabei nicht.

Werden keine Zeitscheiben aufgerollt, so muss zwar der Junction Tree des dynamischen Bayesschen Netzes vollständig neu berechnet werden, es ergeben sich aber keine weiteren Probleme.

Probleme treten allerdings auf, wenn ein Rollup immer dann ausgeführt wird,

wenn eine neue Zeitscheibe etabliert wurde (siehe Kapitel 3.3). In Abb. 5.2(b) ist das dynamische Bayessche Netz zum Zeitpunkt $t$ zu sehen, bevor Zeitscheibe $t-1$ aufgerollt wird. Bis zum Zeitpunkt $t-1$ wurden die Zeitscheiben $h$ ($0 \leq h < t-1$) jeweils dann aufgerollt, wenn die neue Zeitscheibe $h+1$ etabliert war. Zu jedem Zeitpunkt $h$ ($0 \leq h \leq t-2$) waren somit immer maximal zwei Zeitscheiben des dynamischen Bayesschen Netzes repräsentiert.

Als nun zum Zeitpunkt $t$ das Zeitscheibenschema TSS_3 als Zeitscheibe $t$ instantiiert wurde, ließ sich zeitscheibenüberspringende Kante nicht mehr etablieren. Insbesondere wird die Abhängigkeit der Zeitscheibe $t$ von Zeitscheibe $t-a$ nicht mehr modelliert.

Damit ein Rollup mit den Mitteln aus Kapitel 3.3 möglich ist, muss also eine mögliche Beeinflussung einer Zeitscheibe auf eine nachfolgende Zeitscheibe durch alle Zeitscheiben durchgereicht werden, die zwischen den beiden Zeitscheiben liegen. In Abbildung 5.2(c) ist dargestellt, dass sich die Schemata für die Zeitscheiben und die Zustandsentwicklungsmodelle zwischen den Zeitscheiben erweitern. Man sieht, dass temporale Kanten hinzukommen. Es existiere zum Beispiel vom Knoten $Z_{t-a}$ aus Zeitscheibe $t-a$ eine zeitscheibenüberspringende Kante zum Knoten $Z_t$ aus Zeitscheibe $t$. Damit die Beeinflussung durch die dazwischenliegenden Zeitscheiben durchgereicht werden kann, wird für den Knoten $Z_{t-a}$ in jeder Zeitscheibe $h$ ($t-a < h \leq t-1$) ein Knoten $Z_h$ instantiiert. Die bedingten Wahrscheinlichkeiten der Knoten werden wie folgt angepasst, wenn $i$ die Anzahl der Hypothesen von $Z_{t-a}$ bezeichnet:

- Für die Knoten $Z_h$ mit $h \in \{(t-a)+1,\ldots,t-1\}$:

$$\theta_{(Z_h)} = T(Z_h, Z_{h-1}) = \left[\begin{array}{c|ccc} & z_{h,1} & \cdots & z_{h,i} \\ \hline z_{h-1,1} & 1 & & 0 \\ \vdots & & \ddots & \\ z_{h-1,i} & 0 & & 1 \end{array}\right].$$

  Den Knoten $Z_h$ wird somit eine 2-dimensionale Tabelle über den beiden Knoten $Z_{h-1}$ und $Z_h$ der Größe $i \times i$ zugeordnet, die in der Diagonale von $(z_{h-1,1}, z_{h,1})$ bis $(z_{h-1,i}, z_{h,i})$ Einsen und ansonsten nur Nullen als Eintrag enthält. Man kann $\theta_{(Z_h)}$ auch als Identitätstabelle bedingter Wahrscheinlichkeiten bezeichnen, da durch sie der Knoten $Z_h$ denselben BEL-Wert wie sein Vorgängerknoten $Z_{h-1}$ erhält.

- Für den Knoten $Z_t$:
  Anstatt der Tabelle $\theta_{(Z_t)} = T(Z_t, Z_{t-a}, E_1, \ldots, E_p)$ erhält Knoten $Z_t$ nun die Tabelle $\theta_{(Z_t)} = T(Z_t, Z_{t-1}, E_1, \ldots, E_p)$, wobei mit $E_1, \ldots, E_p$ bis auf $Z_{t-a}$ oder $Z_{t-1}$ die Elternknoten von $Z_t$ gemeint sind. Die bedingten Wahrscheinlichkeiten bleiben unverändert. In der Tabelle der bedingten Wahrschein-

lichkeiten des Knotens $Z_t$ wird Elternknoten $Z_{t-a}$ durch Knoten $Z_{t-1}$ ersetzt.

Dadurch dass die bedingten Wahrscheinlichkeiten wie vor definiert sind, wird die Beeinflussung des Knotens $Z_{t-a}$ an Zeitscheibe $t$ unverändert weitergegeben. Jetzt kann der Rollup wie gewohnt mit den Mitteln aus Kapitel 3.3 durchgeführt werden.

Eine zeitscheibenüberspringende Kante kann sich über beliebig viele Zeitscheiben erstrecken. Für jede zeitscheibenüberspringende Kante ist pro Zeitscheibe, die sie überspringt, ein Zwischenknoten einzufügen. Die Komplexität des daraus entstandenen Netzes lässt sich allerdings erst feststellen, wenn ein Rollup-Verfahren oder Inferenzalgorithmus angewendet wird. Die Komplexität des dynamischen Bayesschen Netzes wird im dazugehörigen Junction Tree an der Größe der einzelnen Cliquen festgestellt. Die Bestimmung eines Junction Trees ist entscheidend von der Eliminationsreihenfolge abhängig. Da die Berechnung eines optimalen Junction Trees *NP*-vollständig ist, kann es durchaus vorkommen, dass der neue Junction Tree nach dem Hinzufügen von neuen Knoten weniger komplex ist als der alte Junction Tree.

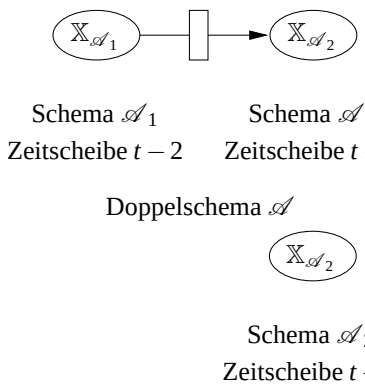
Ein Spezialfall von zeitscheibenüberspringenden Kanten sind die *Doppelschemata*.

### 5.1.2 Doppelschemata

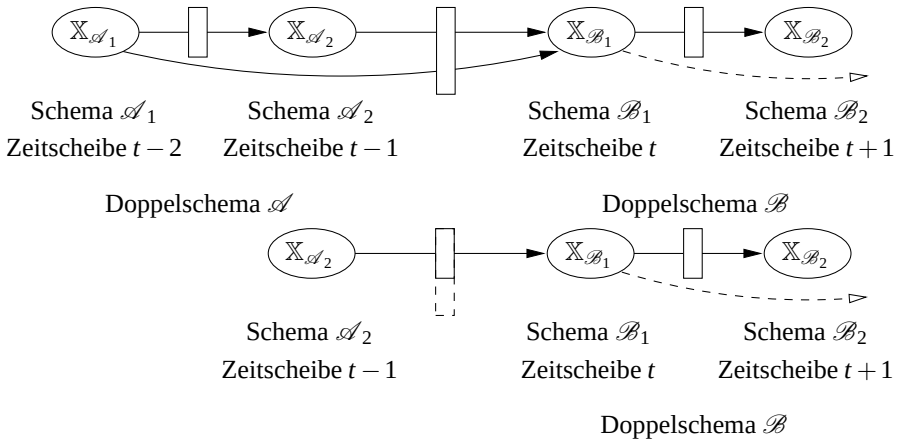
Im folgenden wird beschrieben, wie bei den Doppelschemata zeitscheibenüberspringende Kanten entstehen können. Die Probleme, die von den Doppelschemata herrühren, lassen sich dann mit den Mitteln beheben, die für den Fall der zeitscheibenüberspringenden Kanten im vorhergehenden Abschnitt entwickelt wurden: An entsprechenden Stellen im dynamischen Bayesschen Netz werden Zwischenknoten eingefügt.

Es sei ein dynamisches Bayessches Netz zum Zeitpunkt $t-3$ gegeben, das nicht aufgerollt wird. Es werde nun das Doppelschema $\mathscr{A}$ mit den Schemata $\mathscr{A}_1$ und $\mathscr{A}_2$ instantiiert. Dabei entstehen die beiden Zeitscheiben $t-2$ und $t-1$. In Abbildung 5.3(a) ist ein Schnappschuss des resultierenden dynamischen Bayesschen Netzes zum Zeitpunkt $t-1$ zu sehen.

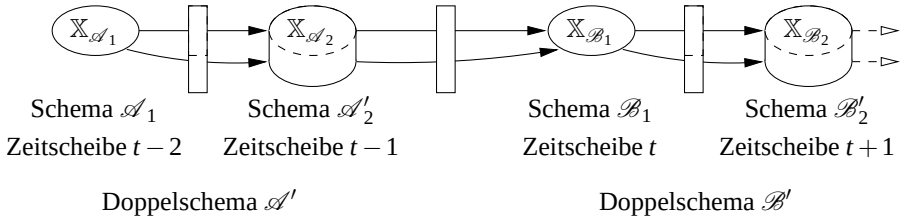
Es werden die Zeitscheiben $t-2$ und $t-1$ mit dem dazwischenliegenden Zustandsentwicklungsmodell schematisch dargestellt. Die Zeitscheiben bis vor dem Zeitpunkt $t-2$ sind ausgeblendet. Zum Zeitpunkt $t-1$ besteht das dynamische Bayessche Netz somit aus $t$ Zeitscheiben. Jetzt wird das Doppelschema $\mathscr{B}$ mit den beiden Schemata $\mathscr{B}_1$ und $\mathscr{B}_2$ als Zeitscheiben $t$ und $t+1$ instantiiert. Die beiden Doppelschemata $\mathscr{A}$ und $\mathscr{B}$ können auch identisch sein. Beispielsweise sei im Schema $\mathscr{B}_1$ ein Knoten enthalten, der eine Zufallsvariable repräsentiert,




(a) DBN ohne (oben) und mit Rollup (unten) zum Zeitpunkt $t-1$. Oben: Das Doppelschema $\mathscr{A}$ ist instantiiert. Unten: Vom Doppelschema $\mathscr{A}$ ist noch das Schema $\mathscr{A}_2$ vorhanden.




(b) DBN ohne (oben) und mit Rollup (unten) zum Zeitpunkt $t$: Es wird das Doppelschemata $\mathscr{B}$ instantiiert. Oben: Es werden zeitscheibenüberspringende Kanten von Zeitscheibe $t-2$ zu Zeitscheibe $t$ induziert. Unten: Es werden keine zeitscheibenüberspringende Kanten induziert.




(c) Zeitscheibenüberspringende Kanten mit Korrektur: Die verschiedenen Schema der Doppelschemata werden um Zwischenknoten ergänzt, so dass eine zeitscheibenüberspringende Kante im jeweiligen Teilschema zu liegen kommt.

Abbildung 5.3: Zeitscheibenüberspringende Kanten (Doppelschema)

die von einer Zufallsvariablen beeinflusst wird, deren entsprechender Knoten im Schema $\mathscr{A}_1$ enthalten sei. Das Zustandsentwicklungsmodell für $\mathbb{X}_{\mathscr{B}_1}$ zum Zeit-

punkt $t$ ist nicht mehr nur von der Zeitscheibe $t-1$ abhängig, sondern auch von der Zeitscheibe $t-2$, in der das Schema $\mathscr{A}_1$ als Zeitscheibe instantiiert wurde. Es wird eine *zeitscheibenüberspringende Kante* von Zeitscheibe $t-a$ zu Zeitscheibe $t$ induziert[1]. Dies ist in der Abbildung 5.3(b) durch einen gerichteten Pfeil von Zeitscheibe $t-2$ zu Zeitscheibe $t$ dargestellt. Das Schema $\mathscr{A}_2$ und das Zustandsentwicklungsmodell zwischen $\mathscr{A}_1$ und $\mathscr{A}_2$ ändern sich dabei nicht.

Werden keine Zeitscheiben aufgerollt, so muss zwar der Junction Tree für das dynamische Bayessche Netz insgesamt neu berechnet werden, ohne dass auch alten Berechnungen wiederzuverwenden sind, aber es ergeben sich keine weiteren Probleme. Wenn jedoch der Rollup immer dann durchgeführt wird, bevor ein neues Doppelschema instantiiert wird, so stellt sich die Situation allerdings anders dar: In Abbildung 5.3(a) ist das dynamische Bayessche Netz zum Zeitpunkt $t-1$ zu sehen, bevor das Doppelschema $\mathscr{B}$ etabliert wird. Bis zum Zeitpunkt $t-2$ wurden die Zeitscheiben $h$ ($0 \leq h \leq t-2$) jeweils dann aufgerollt, wenn mehr als zwei Zeitscheiben etabliert war. Zu jedem Zeitpunkt $h$ ($0 \leq h \leq t-2$) waren somit immer maximal nur drei Zeitscheiben des dynamischen Bayesschen Netzes repräsentiert.

Wird nun zum Zeitpunkt $t$ das Doppelschema $\mathscr{B}$ mit den Schemata $\mathscr{B}_1$ und $\mathscr{B}_2$ als Zeitscheiben $t$ und $t+1$ instantiiert, so lässt sich die Abhängigkeit der Zeitscheibe $t$ von Zeitscheibe $t-2$, in der das Schema $\mathscr{A}_1$ instantiiert wurde, nicht mehr modellieren. In den Abbildungen 5.3(a) und 5.3(b) sind die aufgerollten Zeitscheiben gestrichelt eingezeichnet.

In Abbildung 5.3(b) erkennt man zusätzlich, dass die zeitscheibenüberspringende Kante nicht mehr etabliert werden kann. Insbesondere wird die Abhängigkeit der Zeitscheibe $t$ von der Zeitscheibe $t-2$ nicht mehr modelliert.

Damit ein Rollup mit den Mitteln aus Kapitel 3.3 möglich ist, muss also eine mögliche Beeinflussung einer Zeitscheibe auf eine andere Zeitscheibe durch alle Zeitscheiben, die dazwischen liegen, durchgereicht werden. In Abbildung 5.3(c) ist dargestellt, dass sich die Schemata und die Zustandsentwicklungsmodelle der Zeitscheiben erweitern. Man sieht, dass temporale Kanten hinzukommen.

Im Fall der Doppelschemata muss nun in vergleichbarer Weise wie im allgemeinen Fall der zeitscheibenüberspringenden Kanten vorgegangen werden. Entsprechende Knoten sind in den verschiedenen Schemata neu zu modellieren. Wie die bedingten Wahrscheinlichkeiten dieser entsprechenden Knoten zu wählen sind, ist ebenfalls im allgemeinen Fall der zeitscheibenüberspringenden Kanten behandelt und kann unverändert übernommen werden. Jetzt kann der Rollup wie gewohnt mit den Mitteln aus dem Kapitel 3.3 durchgeführt werden. Eine andere Möglichkeit wäre auch, das Doppelschema als eine Zeitscheibe zu behandeln, so

[1] Um eine zeitscheibenüberspringende Kante zu induzieren, könnten auch die Schemata $\mathscr{B}_2$ und $\mathscr{A}_1$ bzw. $\mathscr{B}_2$ und $\mathscr{A}_2$ involviert sein.

dass immer der Rollup eines Doppelschemas durchgeführt wird. Dabei würden keine zeitscheibenüberspringende Kanten entstehen, die vom Doppelschema herrühren.

Ein anderer Spezialfall der zeitscheibenüberspringenden Kanten sind die statischen Knoten, wie wir im folgenden Abschnitt sehen werden.

### 5.1.3 Statische Knoten

Mit statischen Knoten lassen sich *typische* Eigenschaften modellieren, die während einer Sitzung als unveränderlich angesehen werden können. Mit jeder Evidenz werden sie zunehmend genauer eingeschätzt. Durch ihre Modellierung als statische Knoten haben sie auf jede Zeitscheibe denselben Einfluss.

In diesem Abschnitt wird beschrieben, warum statische Knoten für dynamische Bayessche Netze und insbesondere für die Rollup-Verfahren aus Kapitel 3.3 problematisch sind. Desweiteren wird gezeigt, wie sich die statischen Knoten auf die zeitscheibenüberspringenden Kanten zurückführen lassen und sich die Probleme lösen lassen, die durch die statischen Knoten für die Rollup-Verfahren aufgeworfen werden.

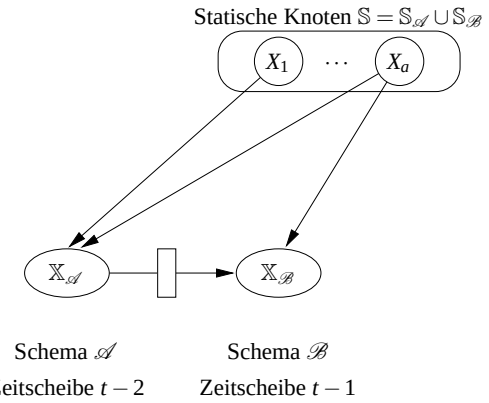
Der Knoten Wissensniveau des Schülers aus Beispiel 3.1.1 ist ein typisches Beispiel für einen statischen Knoten, mit dem Unterschied, dass der Knoten Wissensniveau des Schülers in jeder Zeitscheibe modelliert ist.

Es sei nun ein dynamisches Bayessches Netz zum Zeitpunkt $t-3$ gegeben, das nicht aufgerollt wird. Werde nun als Zeitscheibe $t-2$ das Zeitscheibenschema $\mathscr{A}$ und als Zeitscheibe $t-1$ das Zeitscheibenschema $\mathscr{B}$ instantiiert (siehe Abbildung 5.4(a)). Diese beiden Zeitscheibenschemata seien von den statischen Knoten $X_1,\ldots,X_a$ abhängig. Zu jedem Zeitpunkt $h \leq t-1$ seien die Zeitscheibenschemata des dynamischen Bayesschen Netzes so instantiiert, dass keine zeitscheibenüberspringende Kanten induziert werden[2]. Dieses wird im generischen Schema durch die gerichteten Kanten zwischen den konsekutiven Zeitscheiben angedeutet. Die Zeitscheiben erfüllen damit jedoch nicht die Markov-Eigenschaft, die für dynamische Bayessche Netze verlangt wird, da die Zeitscheiben auch von den statischen Knoten abhängig sind. Die Problematik wird im folgenden dargestellt.

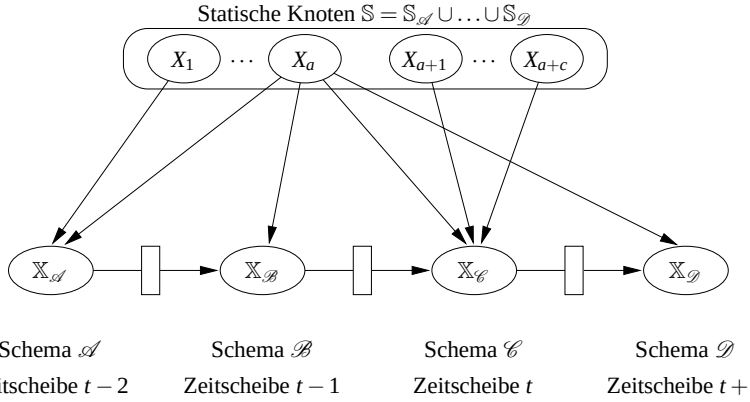
In unserem Beispiel beeinflussen nicht alle dargestellten statischen Knoten jedes Zeitscheibenschema. So beeinflusst der statische Knoten $X_1$ nur direkt das Zeitscheibenschema $\mathscr{A}$.

Beispielhaft werden noch die Zeitscheibenschemata $\mathscr{C}$ und $\mathscr{D}$ als Zeitscheibe $t$ bzw. $t+1$ instantiiert. Wie in Abbildung 5.4(b) zu sehen ist, induzieren manche

[2] Diese Annahme kann vorausgesetzt werden, da ansonsten das Verfahren aus Abschnitt 5.1.1 zur Anwendung kommt.




(a) DBN zum Zeitpunkt $t-1$: Die Schemata $\mathscr{A}$ und $\mathscr{B}$ und ihre statischen Knoten $\mathbb{S}_{\mathscr{A}} \cup \mathbb{S}_{\mathscr{B}}$ sind instantiiert. (Die statischen Knoten für das Schema $\mathscr{B}$ wurden schon vom Schema $\mathscr{A}$ instantiiert. Es gilt: $\mathbb{S}_{\mathscr{A}} \cap \mathbb{S}_{\mathscr{B}} = \mathbb{S}_{\mathscr{B}}$.)




(b) DBN zum Zeitpunkt $t+1$: Die Schemata $\mathscr{C}$ und $\mathscr{D}$ und die statischen Knoten $\mathbb{S}_{\mathscr{C}} \cup \mathbb{S}_{\mathscr{D}}$ wurden instantiiert. Das Schema $\mathscr{C}$ wird auch von den statischen Knoten in $\mathbb{S}_{\mathscr{A}} \cup \mathbb{S}_{\mathscr{B}}$ beeinflusst.

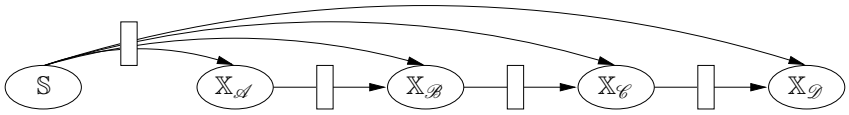
Abbildung 5.4: Statische Knoten: Die Knoten der Schemata sind nur von gewissen statischen Knoten abhängig.

Zeitscheibenschemata neue statische Knoten, die noch nicht instantiiert wurden, andere Zeitscheibenschemata greifen auf vorhandene statische Knoten zurück. Ebenso beeinflussen manche statische Knoten immer nur Zeitscheiben in denen ein bestimmtes Zeitscheibenschema instantiiert wurde (z. B. der statische Knoten $X_1$ das Zeitscheibenschema $\mathscr{A}$). Andere statische Knoten beeinflussen jede Zeitscheibe.

Wie im Fall der zeitscheibenüberspringenden Kanten ergeben sich die Probleme mit den statischen Knoten erst, wenn der Rollup (siehe Kapitel 3.3) immer dann durchgeführt wird, wenn eine neue Zeitscheibe etabliert wurde. Werden keine Zeitscheiben aufgerollt, so muss der zugehörige Junction Tree des dynami-

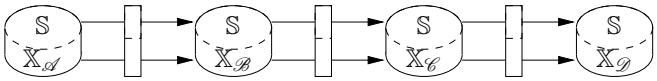
schen Bayesschen Netzes zwar insgesamt neu berechnet werden, aber es ergeben sich keine weiteren Probleme (vorausgesetzt, dass sich der Junction Tree des dynamischen Bayesschen Netzes noch berechnen lässt.). Die Situation stellt sich allerdings anders dar, wenn der Rollup immer dann durchgeführt wird, wenn eine neue Zeitscheibe etabliert wurde (siehe Kapitel 3.3). Wird die Zeitscheibe $t-2$ des Bayesschen Netzes in Abbildung 5.4(b) aufgerollt, so stellt sich die Frage, ob die statischen Knoten $X_1, \ldots, X_a$ auch zur Zeitscheibe $t-2$ gehören und mit aufgerollt werden. Zumindest der Knoten $X_a$ kann nicht ohne weiteres aufgerollt werden, da er auch noch andere Zeitscheiben beeinflusst. Auch der Knoten $X_1$ darf nicht aufgerollt werden, denn in ferner Zukunft könnte wieder eine Zeitscheibe mit dem Zeitscheibenschema $\mathscr{A}$ instantiiert werden. Ebenso könnte man die ganze Zeit auch am statischen Knotens $X_1$ interessiert sein, dessen BEL-Wert sich ebenfalls durch das Eintragen von Evidenz in den anderen Zeitscheiben ändert.

Die statischen Knoten dürfen also nie aufgerollt werden, und sie benötigen immer eine Verbindung zu den Zeitscheiben, damit sie ihren BEL-Wert an die jeweilige Situation anpassen können.




Statische Knoten | Schema $\mathscr{A}$ Zeitscheibe $t-2$ | Schema $\mathscr{B}$ Zeitscheibe $t-1$ | Schema $\mathscr{C}$ Zeitscheibe $t$ | Schema $\mathscr{D}$ Zeitscheibe $t+1$

(a) DBN zum Zeitpunkt $t+1$: Die statischen Knoten werden als eigene Zeitscheibe betrachtet. Man erhält den allgemeinen Fall von zeitscheibenüberspringenden Kanten.




Schema $\mathscr{A}$ Zeitscheibe $t-2$ | Schema $\mathscr{B}$ Zeitscheibe $t-1$ | Schema $\mathscr{C}$ Zeitscheibe $t$ | Schema $\mathscr{D}$ Zeitscheibe $t+1$

(b) DBN zum Zeitpunkt $t+1$: Die statischen Knoten werden in die einzelnen Schemata eingebettet, so dass es keine zeitscheibenüberspringende Kanten mehr gibt.

Abbildung 5.5: Statische Knoten: Die statischen Knoten werden in die Schemata eingebettet.

Betrachtet man die Menge aller möglichen statischen Knoten als besondere Zeitscheibe $-1$ (siehe Abbildung 5.5(a)), dann lassen sich die Probleme, die die statischen Knoten beim Rollup verursachen, ganz genauso wie die Probleme lösen, die die zeitscheibenüberspringenden Kanten herbeigeführt haben: Wir ergänzen jedes Schema um alle statischen Knoten (siehe Abbildung 5.5(b)) und definieren die bedingten Wahrscheinlichkeiten der ergänzten statischen Knoten, wie

es im Abschnitt 5.1.1 beschrieben wurde. In Abbildung 5.5 ist auch dargestellt, dass sich die Schemata und die Zustandsentwicklungsmodelle zwischen den Zeitscheiben erweitern. Man sieht, dass temporale Kanten hinzukommen. Jetzt kann der Rollup wie gewohnt mit den Mitteln aus Kapitel 3.3 durchgeführt werden, und es müssen maximal immer nur zwei Zeitscheiben vorhanden sein.

### 5.1.4 Zusammenfassung

Für den Modellierungsprozess und eine übersichtlichere Darstellung dynamischer Bayesscher Netze haben wir neue Modellierungsstrukturen eingeführt und somit die dynamischen Bayesschen Netze zu dynamischen Bayesschen Netze *n*-ter Ordnung erweitert.

Wie wir jedoch an diversen generischen Schemata gesehen haben, verursachen diese neuen Modellierungsstrukturen für die Rollup-Verfahren massive Probleme. An den generischen Schemata konnten wir zeigen, wie sich durch graphentheoretische Umformungen des dynamischen Bayesschen Netzes *n*-ter Ordnung die Probleme beheben lassen, so dass wieder ein Rollup durchgeführt werden kann.

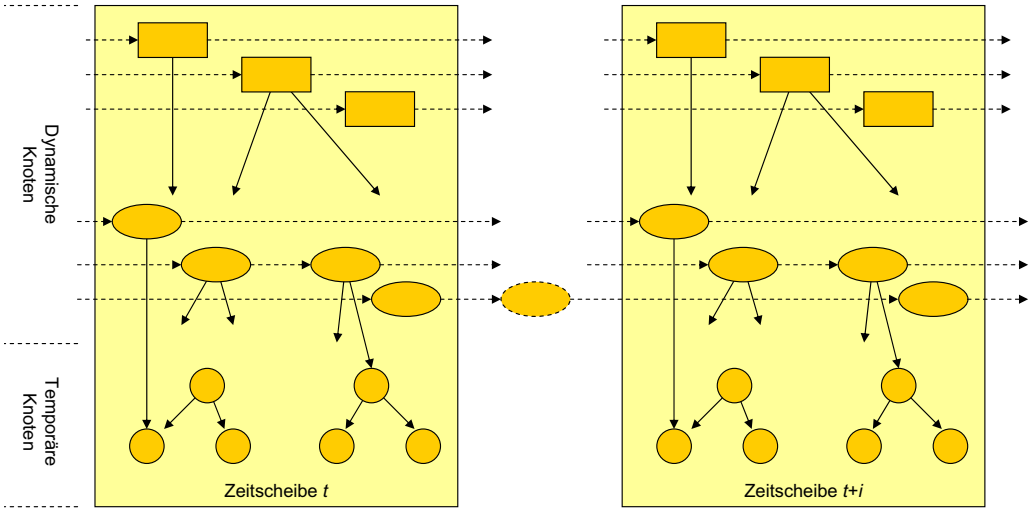



Abbildung 5.6: Prototypische Darstellung eines umformulierten dynamischen Bayesschen Netzes *n*-ter Ordnung.

Im Fall der zeitscheibenüberspringenden Kanten werden solange Zwischenknoten in der betroffenen Kante eingefügt, bis die Kante in jeder der Zeitscheiben zu liegen kommt, die sie übersprungen hat. Die Probleme, die die Doppelschemata und die statischen Knoten verursachen, wurden gelöst, indem die beiden Modellierungsstrukturen auf die zeitscheibenüberspringenden Kanten zurückgeführt

wurden. In Abbildung 5.6 sind die graphentheoretischen Umformungen prototypisch dargestellt, die sich dem Leser durch Vergleich mit Abbildung 5.1 leicht erschließen werden.

Im weiteren möchten wir die Knoten und Zeitscheiben mit zusätzlichen Informationen versehen. Bislang ist der zeitliche Abstand zwischen der Instantiierung zweier Zeitscheiben nicht explizit angegeben, sondern nur implizit in den Übergangswahrscheinlichkeiten der dynamischen Knoten modelliert.

## 5.2 Sensorverarbeitung

Wir werden im folgenden sehen, dass es u. a. bei der Sensormodellierung wichtig sein kann, im voraus den zeitlichen Abstand zwischen den Instantiierungen zweier Zeitscheiben zu kennen. Jedoch gibt es aber auch Ereignisse, die nicht vorhersehbar und unregelmäßig auftreten, so dass man keinen zeitlichen Abstand zwischen den Instantiierungen zweier Zeitscheiben angeben kann.

Das führt uns zu den beiden Begriffen *ereignis-* und *zeitgesteuerte Instantiierung*, die wir im folgenden Abschnitt genauer betrachten werden.

### 5.2.1 Ereignis- und zeitgesteuerte Instantiierung

Wir wollen weiterhin beim dynamischen Bayesschen Netz aus Abschnitt 7.1 bleiben, mit dem sich sprachliche Symptome und die Eigenschaften manuellem Eingabeverhaltens interpretieren lassen, um die ereignis- und zeitgesteuerte Instantiierung zu motivieren. Wir wollen es rein hypothetisch um die Fähigkeit erweitern, physiologische Daten interpretieren zu können. Die sprachlichen Symptome, das manuelle Eingabeverhalten und die physiologischen Daten werden durch entsprechende Sensoren gemessen und gegebenenfalls durch Vorverarbeitungsverfahren und Klassifizierer so aufbereitet, dass sie als Evidenz in den Sensorknoten der Zeitscheibe eingetragen werden können.

Bei der Kombination von physischen und verhaltensabhängigen Symptomen ist dabei zu beachten, dass beispielsweise unregelmäßige Ereignisse in relativ großen, nicht vorhersagbaren Abständen (wie beispielsweise sprachliche Äußerungen) oder regelmäßige Ereignisse in kurzen, vorhersagbaren Abständen (wie Herzschlag oder Atmung) auftreten.

Hier kann man einmal die *ereignisgesteuerte Instantiierung* einsetzen und dabei unrelevante Ereignisse herausfiltern, die zuvor durch eine Offline-Sensitivitätsanalyse bestimmt wurden. Und im anderen Fall ist die *zeitgesteuerte Instantiierung* von Durchschnittswerten (z. B. Herzschlagfrequenz) denkbar, wobei sich natürlich nicht alle Sensordaten zu Durchschnittswerten umrechnen lassen und in

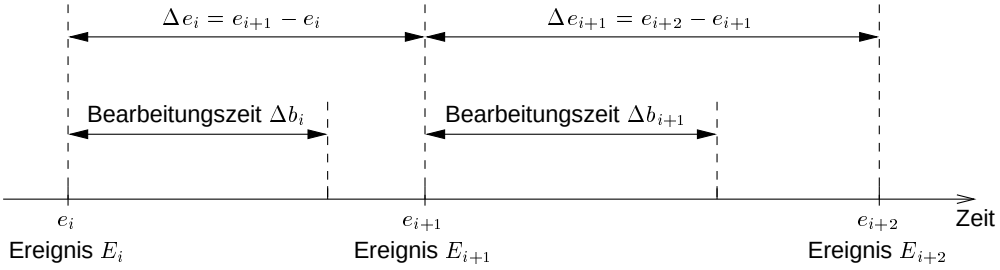



Abbildung 5.7: Ereignisgesteuerte Instantiierung von Zeitscheiben.

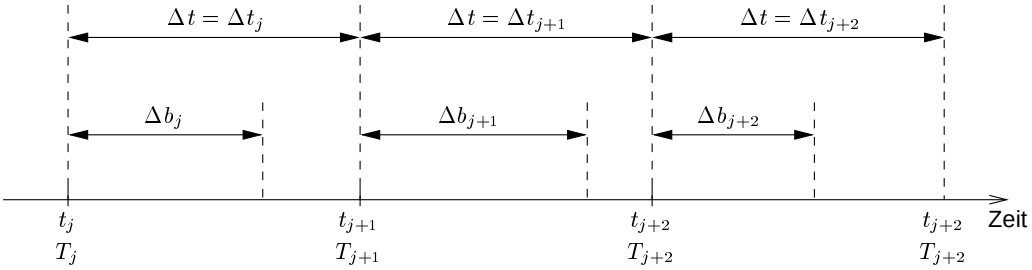



Abbildung 5.8: Zeitgesteuerte Instantiierung von Zeitscheiben.

diesem Fall nur eine ereignisgesteuerte Instantiierung in Frage kommt. Schauen wir uns die beiden Verfahren einmal genauer an:

Betrachten wir als Auslöser für die Instantiierung einer Zeitscheibe ein Ereignis wie beispielsweise eine sprachliche Äußerung. Diese sogenannte *ereignisgesteuerte Instantiierung* (siehe Abbildung 5.7) kommt beispielsweise bei AGENDER (siehe Abschnitt 7.4) zum Einsatz. Zu einem Zeitpunkt $e_i$ tritt ein Ereignis $E_i$ ein (im Beispiel AGENDER eine sprachliche Äußerung). Mit dem Zeitintervall $\Delta e_i$ bezeichnen wir die Zeit, die verstreicht, bis nach dem Ereignis $E_i$ das Ereignis $E_{i+1}$ eintritt. Mit $\Delta b_i$ bezeichnen wir die Bearbeitungszeit, die die Klassifizierer und das dynamische Bayessche Netz zur Berechnung benötigen. Idealerweise gilt: $\forall \Delta e_i : \Delta e_i > \Delta b_i$, so dass es schon vor jedem neuen Ereignis eine Rückmeldung gibt. Dies muss aber nicht immer erfüllt sein und hängt erheblich von der Anwendungsdomäne ab. Beispiele für die ereignisgesteuerte Instantiierung wären sprachliche Äußerungen (wie schon erwähnt) oder ein plötzlicher äußerer Reiz.

Bei der *zeitgesteuerten Instantiierung* ist der Auslöser für die Instantiierung einer Zeitscheibe ein vorgegebener Zeitpunkt (siehe Abbildung 5.8). Mit dem Zeitintervall $\Delta t_j$ bezeichnen wir die Zeit, zwischen zwei zeitgesteuerten Instantiierungen $T_j$ und $T_{j+1}$. Mit $\Delta b_i$ bezeichnen wir wie vor die Bearbeitungszeit. Idealerweise gilt auch hier: $\forall \Delta t_j : \Delta t_j > \Delta b_j$. Im Regelfall sind die Zeitintervalle $\Delta t_j$ alle gleich groß, d. h. $\forall j : \Delta t_j = \Delta t$. Wenn man nun die maximale Bearbeitungszeit abschätzen kann, so lässt sich $\Delta t$ sehr leicht bestimmen. Sensordaten für die

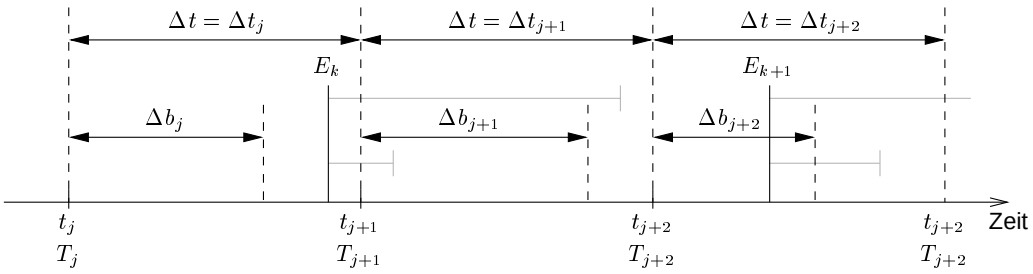



Abbildung 5.9: Ereignis- und zeitgesteuerte Instantiierung von Zeitscheiben.

beispielsweise eine zeitgesteuerte Instantiierung in Frage käme, wären die Herzschlagfrequenz, der über einem bestimmten Zeitraum gemittelte Hautleitwert oder die Anzahl von Hautleitwertsspontanfluktuationen in einem bestimmten Zeitabschnitt. Im allgemeinen sind also für die zeitgesteuerte Instantiierung Sensordaten geeignet, die sich über einen gewissen Zeitraum mitteln oder anderweitig zusammenfassen lassen.

Bei der *kombinierten ereignis- und zeitgesteuerten Instantiierung* ist der Auslöser für die Instantiierung einer Zeitscheibe ein Ereignis oder ein vorgegebener Zeitpunkt. Wie in Abbildung 5.9 zu sehen ist, kann es dabei zur Interferenz zwischen der ereignis- und der zeitgesteuerten Instantiierung kommen.

Beispielsweise tritt das Ereignis $E_k$ kurz vor der zeitgesteuerten Instantiierung ein. Die Zeitscheibe, die zum Zeitpunkt $t_j$ instantiiert wurde, wurde aber schon berechnet (Bearbeitungszeit $\Delta b_j$). Die Instantiierung der Zeitscheibe nur mit dem Ereignis würde im Fall des oberen Balkens des Ereignisses $E_k$ in der Berechnung zu lange dauern, so dass die zeitgesteuerte Instantiierung nicht mehr rechtzeitig erfolgen könnte. Eine mögliche Lösung wäre, die zeitgesteuerte Instantiierung variabel zu gestalten und die Instantiierung zeitlich nach vorne zu verschieben. Im Falle des unteren Balkens des Ereignisses $E_k$ könnte man bei einer variablen zeitgesteuerten Instantiierung die zeitgesteuerte Instantiierung zeitlich nach hinten verschieben. Eine andere Möglichkeit wäre es, die Ereignisse bis zur nächsten zeitgesteuerten Instantiierung aufzusammeln. Das Problem ist hier, dass sich Ereignisse in einer Zeitscheibe widersprechen könnten.

Es könnte aber auch dazu kommen, dass ein Ereignis eintritt, während die zeitgesteuerte Instantiierung noch berechnet wird, wie es durch das Ereignis $E_{k+1}$ in Abbildung 5.9 dargestellt wird. Je nachdem wie stark das Ereignis die Berechnung beeinflussen würde, könnte es notwendig sein, dass die Berechnung der zeitgesteuerten Instantiierung abgebrochen und stattdessen die Zeitscheibe der ereignisgesteuerten Instantiierung berechnet wird. Hierzu ist eine Offline-Sensitivitätsanalyse notwendig, die uninteressante Ereignisse (bzw. Ereigniskombinationen) herausfiltert, so dass diese nicht mehr instantiiert werden müssen.

Wenn jedoch die Bearbeitungszeiten $\Delta b$ sowohl bei der ereignisgesteuerten als auch bei der zeitgesteuerten Instantiierung gering genug sind, so kann auch $\Delta t$ der zeitgesteuerten Instantiierung verkleinert werden, so dass ohne große zeitliche Verzögerung eine zeitgesteuerte Instantiierung erfolgt, wenn ein Ereignis eintritt und somit nur noch die zeitgesteuerte Instantiierung notwendig wird.

Bei der bisherigen theoretischen Betrachtung der ereignis- und zeitgesteuerten Instantiierung wurde noch nicht berücksichtigt, wie die Einträge in den Tabellen bedingter Wahrscheinlichkeiten angepasst werden müssen, so dass bei vergleichbaren Messungen auch vergleichbare Ergebnisse erzielt werden. Nehmen wir als Beispiel ein EKG, das wir beispielsweise sowohl mit einer Frequenz von 512 Hz als auch mit einer Frequenz von 256 Hz aufnehmen. In beiden EKG-Kurven, die wir erhalten, sind alle Charakteristika einer EKG-Kurve (wie P-Zacke, QRS-Komplex oder T-Zacke) enthalten und beide Kurven lassen sich noch einander zuordnen. Wenn wir das EKG allerdings mit 1 Hz aufnehmen, so wird die EKG-Kurve noch nicht einmal mehr als eine solche erkannt werden, geschweige denn mit den beiden anderen EKG-Kurven vergleichbar sein. Nehmen wir nun an, dass wir ein dynamisches Bayessches Netz haben, dessen BEL-Werte seiner Knoten bei der zeitgesteuerten Instantiierung mit einem konstanten Zeitintervall $\Delta t$ für alle Zeitscheiben adäquat berechnet werden. Wenn wir dasselbe Netz bei der doppelten Frequenz mit Evidenz instantiieren würden, d. h. bei einem Zeitintervall $\frac{1}{2}\Delta t$, so wären die Berechnungen ohne eine entsprechende Anpassung der Tabellen bedingter Wahrscheinlichkeiten allerdings nicht mehr adäquat, d. h. um die beiden Kurven ähnlich der beiden EKG-Kurven einander zuordnen zu können, müssen bestimmte Tabellen bedingter Wahrscheinlichkeiten angepasst werden. Gegebenenfalls sind auch die zugehörigen Klassifizierer und Vorverarbeitungsalgorithmen an die veränderte Frequenz zu adaptieren.

In der bisherigen Betrachtung der Sensordaten wurde bisher immer davon ausgegangen, dass die zusammengehörenden Daten auch gleichzeitig anliegen. Jedoch reagiert beispielsweise die Herztätigkeit sehr viel schneller als der Hautleitwert auf Reize wie beispielsweise ein plötzliches Erschrecken. Diese Verzögerung zwischen dargebotenem Reiz und dem Einsetzen einer beobachtbaren Reaktion bezeichnet man als *Latenz*. Im folgenden Abschnitt werden wir dazu ein paar Lösungsvorschläge machen, um die Latenz im dynamischen Bayesschen Netz verarbeiten zu können.

### 5.2.2 Latenz

Mit *Latenz* bezeichnet man die Verzögerungszeit, die zwischen dem dargebotenem Reiz und dem Einsetzen einer beobachtbaren Reaktion liegt. Im allgemeinen spielt die Latenz keine Rolle, wenn alle Sensordaten eingetragen werden können, bevor der Rollup aufgerufen wird, d. h. dass die größte Latenz zusammen mit der

Bearbeitungszeit $\Delta b$ kleiner ist als $\Delta t$ (siehe den vorhergehenden Abschnitt 5.2.1).

Nehmen wir beispielsweise an, dass wir zwei Signale messen und durch ein dynamisches Bayessches Netz interpretieren wollen. Dabei soll die Latenz des einen Signals vernachlässigbar klein sein, und die des anderen Signals bei etwa 3 Sekunden liegen. Der Rollup soll jede Sekunde einmal durchgeführt werden. So kann man die zusammengehörenden Signale nie gleichzeitig in einer Zeitscheibe eingeben. Wenn man jedoch die Latenz vernachlässigt und die aktuell anliegenden Signale in einer Zeitscheibe instantiiert, so könnte es sein, dass sich die Werte widersprechen und es zu einer falschen Einschätzung kommt.

Im folgenden stellen wir drei Lösungsansätze für das Problem der Latenz vor:

1. Wenn nun sowohl die Latenz der Biosignale als auch das Zeitintervall $\Delta t$ der zeitgesteuerten Instantiierung bekannt sind, könnte die Latenz im dynamischen Bayesschen Netz durch entsprechende zeitscheibenüberspringende Kanten hart verdrahtet modelliert werden. Hierbei könnte allerdings die Komplexität des Netzes so groß werden, dass es nicht mehr verarbeitet werden kann, da durch diese Modellierung auch das Zustandsentwicklungsmodell des dynamischen Bayesschen Netzes komplexer wird. Hinzu kommt, dass die Biosignale mit geringer oder keiner Latenz die Einschätzung durch das dynamische Bayessche Netz beeinflussen. Denn wenn das Biosignal mit der Latenz anliegt und in der aktuellen Zeitscheibe instantiiert wird, dann wurden schon alle Biosignale mit geringerer Latenz in vorhergehenden Zeitscheiben als Evidenz eingetragen und gehen somit in die Einschätzung durch das dynamische Bayessche Netz mit ein.

   In Abbildung 5.10 ist prototypisch in einem dynamischen Bayesschen Netz mit vier Zeitscheiben dargestellt, wie durch den vorgeschlagenen Lösungsansatz zeitscheibenüberspringende Kanten entstehen. Vereinfachend wurde in der Darstellung angenommen, dass nur ein Biosignal eine Latenz aufweist. Der Sensorknoten des Biosignales, das in der Darstellung eine Latenz von drei Zeitscheiben aufweist, ist gestrichelt eingezeichnet. Zum Zeitpunkt $t$ tritt der Reiz auf, und erst nach einer Latenz von drei Zeitscheiben wird das dem Reiz zugehörige Biosignal (die Reaktion auf das Reiz) durch den Sensor gemessen. Das Problem der zeitscheibenüberspringenden Kanten muss –ähnlich wie in Abschnitt 5.1.1 geschildert wird– behoben werden, indem Zwischenknoten eingefügt werden, so dass die Kante in jeder Zeitscheibe zu liegen kommt, die sie überspringt. Das bedeutet, dass durch einen Sensorknoten, dessen Biosignal eine Latenz von $n$ Zeitscheiben besitzt, insgesamt $n-1$ Zwischenknoten in einer(!) Zeitscheibe eingefügt werden.

2. Eine andere Möglichkeit wäre beispielsweise, dass man die Sensordaten so lange aufsammelt, bis zusammengehörende Sensordaten gemeinsam in

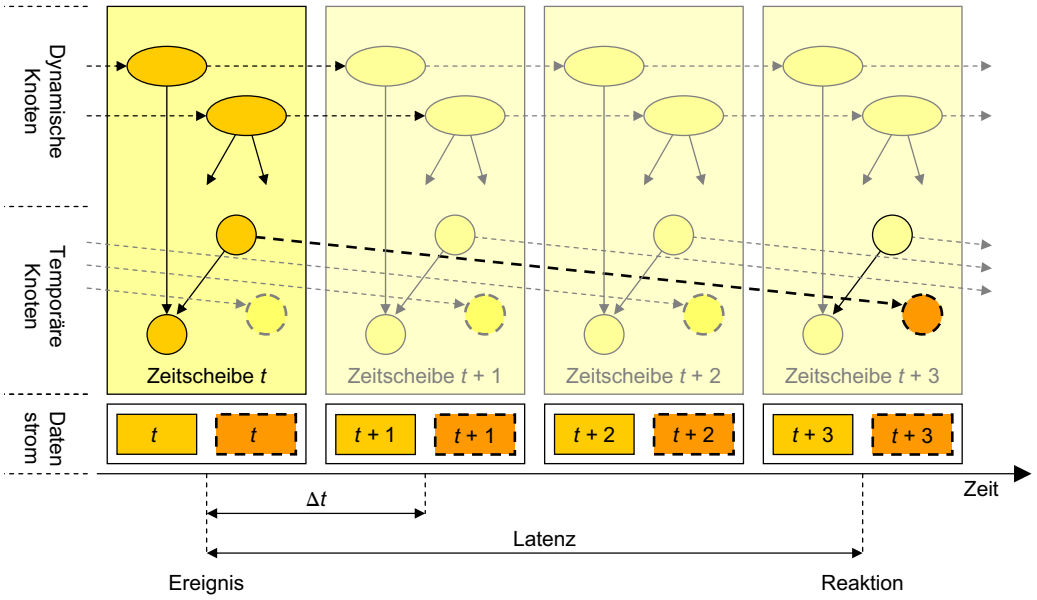



Abbildung 5.10: Latenz: Lösung mit zeitscheibenüberspringenden Kanten

einer Zeitscheibe ausgewertet werden können. Dazu müssen in den einzelnen Sensorknoten die Latenzzeiten angegeben werden, so dass berechnet werden kann, wie lange jeweils die einzelnen Sensordaten gesammelt werden müssen und in welchem Zeitabstand $\Delta t$ die zeitgesteuerte Instantiierung vorgenommen werden kann. Bei dieser Vorgehensweise verzögert sich natürlich die Einschätzung durch das dynamische Bayessche Netz um die größte Latenz der gemessenen Signale.

In Abbildung 5.11 ist prototypisch dargestellt, wie das Problem durch einen Datenpuffer gelöst werden kann. Wie zuvor ist auch diesmal der Sensorknoten des Biosignals, das eine Latenz aufweist, gestrichelt eingezeichnet. Für die Sensorknoten, deren Biosignal keine oder eine geringere Latenz aufweist als das Biosignal des gestrichelt eingezeichneten Sensorknotens, muss ein Datenpuffer für die einkommenden Sensordaten angelegt werden, da diese Daten erst nach der größten Latenz abgearbeitet werden können.

3. Es könnte allerdings sein, dass das Biosignal mit der größten Latenz nicht viel und die anderen Biosignal mit geringerer Latenz schon ausreichend zur Einschätzung beitragen, so dass man auf das Biosignal mit der größten Latenz für die Einschätzung verzichten könnte, um eine schnellere Einschätzung zu erreichen. Dieses Biosignal könnte aber auch von Zeit zu Zeit das sogenannte Zünglein an der Waage sein, so dass man es nicht mit völliger Gewissheit herausnehmen kann. Eine Lösung dieses Problems wären meh-

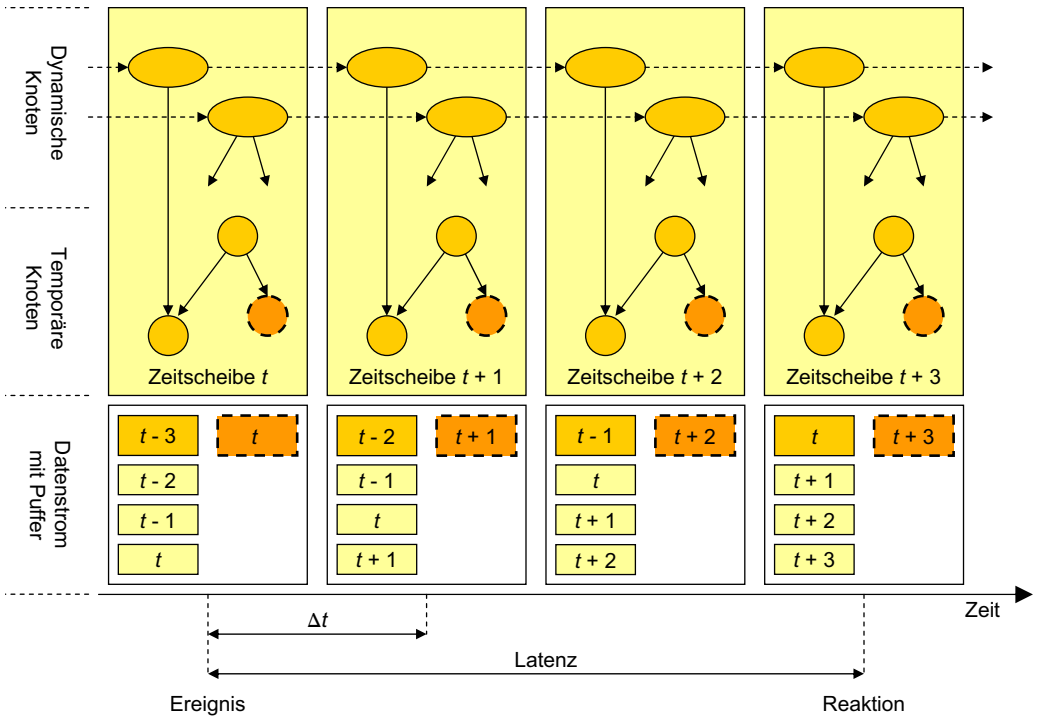



Abbildung 5.11: Latenz: Lösung mit Datenpuffer

rere dynamische Bayessche Netze, die in unterschiedlichen Stufen bzgl. der verschiedenen Latenzen eine Einschätzung vornehmen. Dabei würde das erste dynamische Bayessche Netz die Sensordaten mit der geringsten Latenz als Evidenz zur Eingabe erhalten, so dass sich die Einschätzung nur um die geringste Latenz verzögert. Das zweite dynamische Bayessche Netz würde die Sensordaten mit der geringsten und der zweitgeringsten Latenz als Evidenz zur Eingabe erhalten, wobei sich hier die Einschätzung schon um die zweitgeringste Latenz verzögert und die Sensordaten der geringsten Latenz gesammelt werden müssen. Das letzte dynamische Bayessche Netz würde dann alle Sensordaten als Evidenz zur Eingabe erhalten, was Fall 2 entspricht. Die Menge aller dynamischen Bayesschen Netze entspricht Fall 1, wobei jedoch die Komplexität des einzelnen dynamischen Bayesschen Netzes im Fall 1 nicht erreicht wird. Als Zusatzaufwand hat man allerdings mehrere dynamische Bayessche Netze mit ihren zugehörigen Sensordatenströmen zu verwalten.

Wie zuvor schon ausgeführt wurde, hat jeder der drei Ansätze seine Vor- und Nachteile. Es wäre deshalb wünschenswert, wenn man die Latenz an den entsprechenden Sensorknoten im dynamischen Bayesschen Netz angeben könnte, so dass

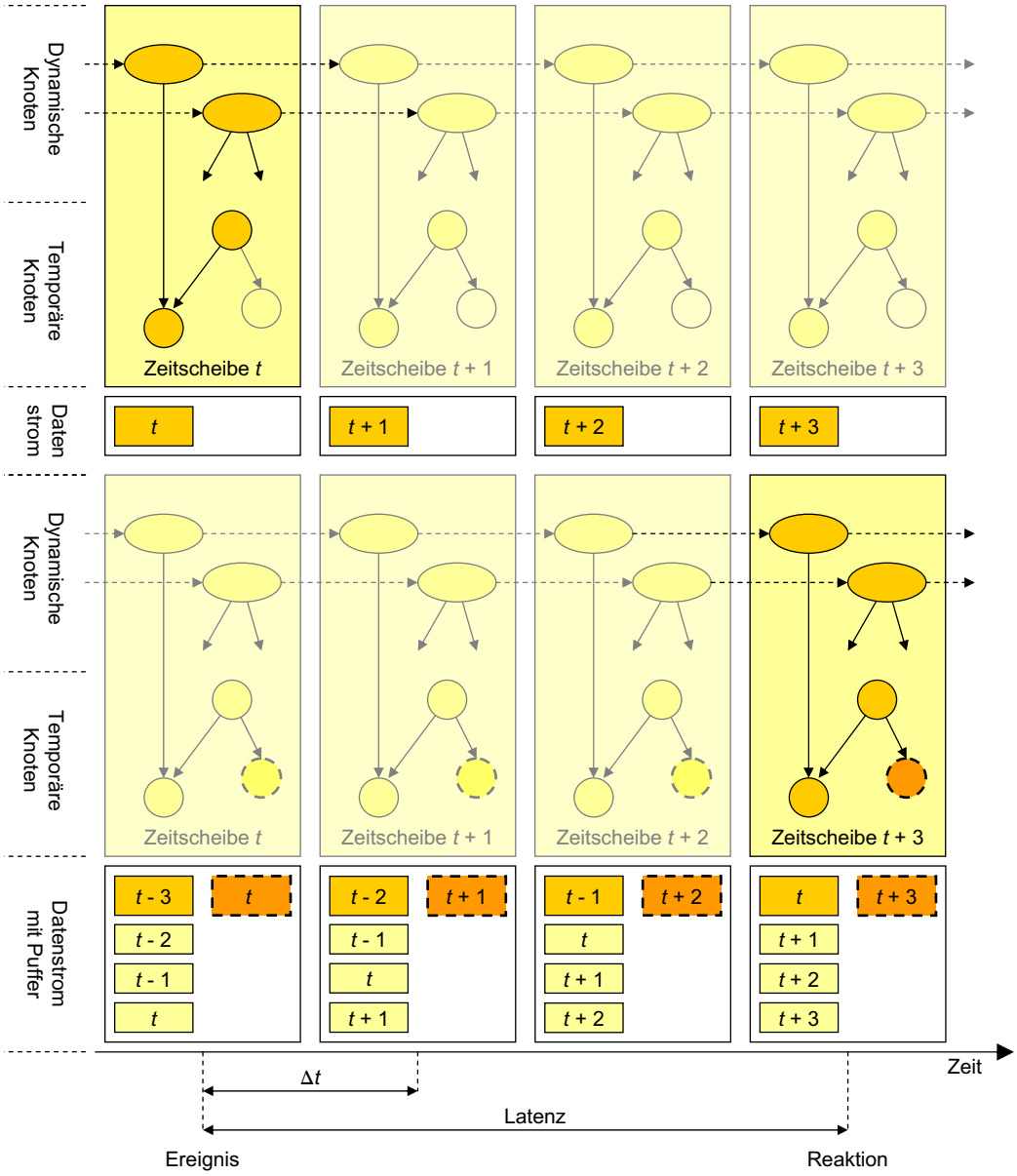



Abbildung 5.12: Latenz: Lösung mit Datenpuffer und mehreren dynamischen Bayesschen Netzen

das System selbsttätig die passende Lösung auswählt.

Die dynamischen Bayesschen Netze der im vorhergehenden Abschnitt 7 vorgestellten Anwendungen verwenden momentan ausschließlich Sensordaten, die gleichzeitig gültig sind, so dass die Latenz in der Modellierung ihrer dynamischen Bayesschen Netze nicht berücksichtigt werden musste.

## 5.3 Zusammenfassung

Wir haben neue Modellierungsstrukturen eingeführt, die den Modellierungsprozess der dynamischen Bayesschen Netze einfacher und aussagekräftiger und ihre Darstellung übersichtlicher gestalten. Es sind dies *statische Knoten*, *zeitscheibenüberspringende Kanten* und *Doppelschemata*. Diese neuen Modellierungsstrukturen erlauben nun Markov-Prozesse *n*-ter Ordnung und erweitern so die dynamischen Bayesschen Netze zu dynamischen Bayesschen Netzen *n*-ter Ordnung.

Wir haben gezeigt, dass diese neuen Modellierungsstrukturen die direkte Anwendung der Rollup-Verfahren aus Abschnitt 3.3 verhindern, aber dass im allgemeinen durch graphentheoretische Umformungen der Rollup wieder ermöglicht wird.

Im Fall der zeitscheibenüberspringenden Kanten werden solange Zwischenknoten in der betroffenen Kante eingefügt, bis die Kante in jeder der Zeitscheiben zu liegen kommt, die sie übersprungen hat. Die Probleme, die die Doppelschemata und die statischen Knoten verursachen, wurden gelöst, indem die beiden Modellierungsstrukturen auf die zeitscheibenüberspringenden Kanten zurückgeführt wurden.

Desweiteren haben wir für das Problem der *Latenz* bei Sensordaten unter Berücksichtigung der *ereignis-* und *zeitgesteuerten Instantiierung* verschiedene Lösungsansätze vorgeschlagen, die sich je nach Anwendungsgebiet des dynamischen Bayesschen Netzes unterschiedlich gut eignen. Auch hier treten zeitscheibenüberspringende Kanten auf, die mit den in diesem Kapitel vorgestellten Mitteln umgewandelt werden können, um wieder einen Rollup durchführen zu können.

# Kapitel 6

# JavaDBN: Codeerzeugung für eingebettete Systeme

In diesem Kapitel stellen wir das System *JavaDBN* vor, das unserem Wissen nach das erste System ist, das dynamische Bayessche Netze auf eingebetteten Systemen mit geringer Rechenleistung zum Laufen bringt. Dazu wird von JavaDBN Quellcode erzeugt (*Java* oder *C++*), der auf vom Anwender angegebene Anfragen an ein dynamisches Bayessches Netz spezialisiert ist. Wir zeigen, wie der generierte Quellcode durch den Anwender modifiziert werden kann, um beispielsweise *zeitabhängige bedingte Wahrscheinlichkeiten* (d. h. der quantitative Teil des Bayesschen Netzes) zu modellieren. Ebenso können mit JavaDBN Sensitivitätsanalysen durchgeführt werden, wie an einem Beispiel gezeigt wird.

Wir werden eine Auswahl der Anwendungen, die mit JavaDBN entwickelt wurden, im Kapitel 7 vorstellen, um verschiedene Aspekte wie die Einbettung des Quellcodes oder die Parametrisierung der Einträge einer Tabelle bedingter Wahrscheinlichkeiten zu beleuchten.

Als *eingebettete Systeme* bezeichnet man Rechnersysteme, die in einer Vielzahl von Geräten wie Autos, Flugzeugen, Waschmaschinen, Kühlschränken oder Handys ihren Dienst versehen. Bei eingebetteten Systemen mit wenig Ressourcen wird oftmals kein *Betriebssystem* oder nur ein Betriebssystem *ohne Speicherschutz* eingesetzt, so dass sich der Entwickler um die Speicherverwaltung kümmern muss. Als Programmiersprachen werden vorzugsweise *C* und *C++* und vereinzelt auch *Java* eingesetzt. Bei Systemen ohne Betriebssystem und vor allem bei massiven *Speichereinschränkungen* oder für *zeitkritische* Anwendungen wird häufig auch *Assembler* verwendet.

Oftmals müssen eingebettete Systeme innerhalb eines vorher fest vorgegebenen Zeitintervalls ein Ergebnis garantiert berechnet haben. Solche Systeme bezeichnet man als Echtzeitsysteme. Die Größe des Zeitintervalls ist dabei von der

Anwendung abhängig und kann von wenigen Nanosekunden bis Tage reichen. So muss beispielsweise die Reaktionszeit des Airbags im Auto bei etwa einer Millisekunde liegen, damit die Fahrzeuginsassen optimal geschützt werden können.

In bisherigen Systemen dient immer ein (dynamisches) Bayessches Netz als Eingabe für den Inferenzalgorithmus. Dies ist jedoch für eingebettete Systeme, die nur über wenig Speicher verfügen, nicht akzeptabel, da das Bayessche Netz erst noch kompiliert werden muss, um auf ihm rechnen zu können. Dies hieße, dass Programmcode für das Kompilieren des Bayesschen Netzes benötigt wird, der den knappen Speicher des eingebetteten Systems noch zusätzlich belegt.

Wir stellen jetzt ein System vor, mit dem dynamische Bayessche Netze auf eingebetteten Systemen verarbeitet werden können. Dazu wird von diesem System Quellcode erzeugt, für den zusätzlich Laufzeitgarantien gegeben werden können.

Anhand des Beispiels aus Abschnitt 4 (siehe Abbildung 4.1 des dynamischen Bayesschen Netzes, Abbildung 4.2 der verwendeten Zeitscheibenschemata und Abbildung 4.5 des gerichteten azyklischen Graphen (nur die durchgezogenen Linien)) wird nun die Anwendung *JavaDBN* vorgestellt. In Abbildung 6.1 ist ein Bildschirmabzug von JavaDBN mit den Zeitscheibenschemata und dem gerichteten Graphen und ein Beispielausschnitt des dazugehörigen arithmetischen Schaltkreises dargestellt.

Dieser von JavaDBN bestimmte arithmetische Schaltkreis erlaubt sowohl die Inferenz im Netz als auch den Rollup vorhergehender Zeitscheiben. Blattknoten repräsentieren dabei die Evidenzvektoren der dazugehörigen Knoten (z. B. evidenceC1) und die Tabellen mit den bedingten Wahrscheinlichkeiten (z. B. cptC1). Knoten für eine Addition sind hellgrau (cpt2_5), und Knoten für eine Multiplikation sind dunkelgrau (cptClique_A0_B0_A1_B1).

Für solche arithmetische Schaltkreise erzeugt JavaDBN Quellcode für Inferenzen oder Sensitivitätsanalysen im dazugehörigen dynamischen Bayesschen Netz wie es im folgenden Abschnitt beschrieben wird.

## 6.1 Ausgabe

JavaDBN kann Quellcode für die Inferenz im dynamischen Bayesschen Netz sowohl in *Java* als auch *C++* erzeugen. Es ist aber nicht auf diese Programmiersprachen eingeschränkt, sondern durch entsprechende Erweiterungen kann es Quellcode in jeder beliebigen Programmiersprache erzeugen.

Weiterhin erzeugt JavaDBN Quellcode für die Sensitivitätsanalyse und unterstützt *MatLab* (siehe [MatLab 06]). Zuerst wollen wir aber genauer auf die Ausgabe des Quellcodes und seine Vorteile für die Inferenz anhand der Programmiersprache Java eingehen.

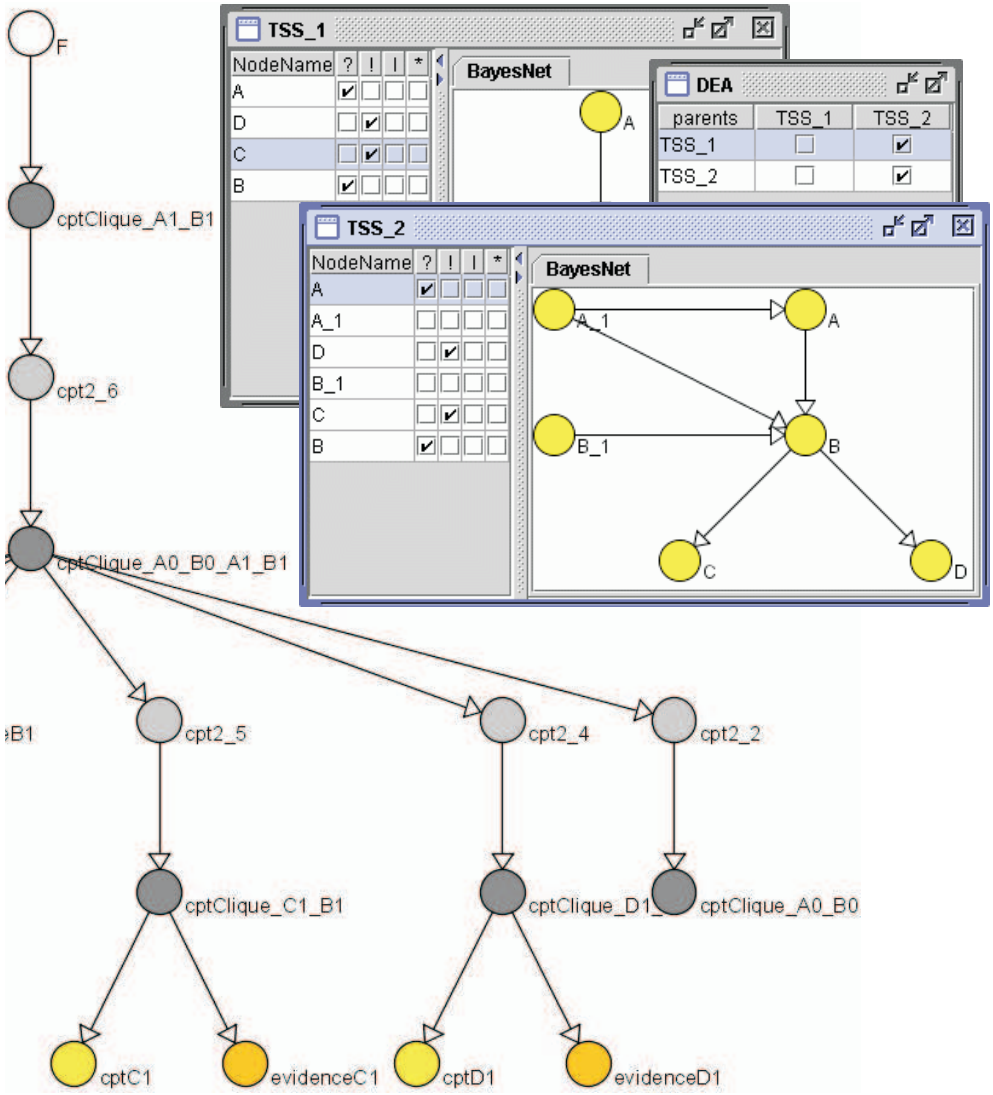



Abbildung 6.1: JavaDBN: Gerichteter Graph und Zeitscheibenschemata (oben rechts) und Beispielausschnitt des dazugehörigen arithmetischen Schaltkreises.

### 6.1.1 Quellcode für Inferenz

Zur Inferenz in dynamischen Bayesschen Netzen wird Quellcode erzeugt, der einen arithmetischen Schaltkreis auswertet und differenziert. Dabei wurde der in Abschnitt 2.4.2 beschriebene Algorithmus bewusst ausgewählt, da er relativ einfach ist und somit der resultierende Quellcode sowohl lesbar als auch verständlich bleibt, so dass bei Bedarf Änderungen im Quellcode relativ einfach vom Anwender vorgenommen werden können. Der Quellcode ist dabei wie folgt strukturiert:

```
/**
 * ABCD-DBN-Polynomial.java
 *
 * @author  Boris Brandherm
 */
public class ABCD-DBN-Polynomial {

    // time slice and normalizing factor
    public double normalize = 0;
    public int timeslice = 1;

    /* variable declarations for
     * each node, clique, and separator */

    // Creates a new instance of ABCD-DBN-Polynomial
    public ABCD-DBN-Polynomial() {
    }

    // Set and reset of evidence

    // collect cliques and separators
    // distribute cliques and separators

    // Inference in the dynamic Bayesian network

    // Rollup
}
```

Im folgenden werden nun

- die Variablenvereinbarungen,
- die Funktionen zum Eintragen von Evidenz,
- die Funktionen für die beiden Phasen Collect und Distribute Evidence
- die Inferenz
- der Rollup und
- die Systemeinbindung

in den einzelnen Abschnitten teilweise mit Beispiel-Quellcode vorgestellt.

**Variablenvereinbarungen** Alle Variablen, die zur Berechnung benötigt werden, werden erzeugt, wenn die Anwendung gestartet wird. Für jeden Knoten des Zeitscheibenschemas wird eine Kontrollvariable counter<nodeName> erzeugt, so dass durch alle Zustände des Knotens gegangen werden kann. Weiterhin wird für jeden Knoten des arithmetischen Schaltkreises eine Variable <nodeName>_vr für den Aufwärtsdurchgang und eine Variable <nodeName>_dr für den Abwärtsdurchgang erzeugt. Zum Beispiel werden für den Knoten C1 die Variablen counterC1, beliefC1_vr, evidenceC1_vr, cptC1_vr, evidenceC1_dr und cptC1_dr, für die Clique Clique_C1_B1 werden die Variablen cptClique_C1_B1_vr und cptClique_C1_B1_dr, und für den Separator 2_5 werden die Variablen cpt2_5_vr und cpt2_5_dr erzeugt. Während der weiteren Laufzeit wird kein Speicher mehr allokiert oder deallokiert, um den Aufruf einer automatischen Speicherbereinigung zu vermeiden.

**Funktionen zum Eintragen von Evidenz** Es werden für ausgewählte Knoten besondere Funktionen, um Evidenz eintragen zu können, zur Verfügung gestellt. Da auf die Variablen auch von außerhalb der Klasse frei zugegriffen werden kann, kann Evidenz für einen Knoten auch direkt eingetragen werden. Um beispielsweise anzugeben, dass der Zustand 2 des Knotens D1 eingetreten ist und dass Zustand 0 und Zustand 1 nicht mehr eintreten werden, müssen Zustand 0 und Zustand 1 mit dem Wert 0 und Zustand 2 mit dem Wert 1 belegt werden, d. h. evidenceD1_vr = $\{0.0, 0.0, 1.0\}$.

```
// Set evidence for: D1
public void setEvidence_D1_State1() {
  for (counterD1 = 0; counterD1 < 2; counterD1++) {
    evidenceD1_vr[counterD1] = 0.0;
  }
  evidenceD1_vr[1] = 1.0;

  return;
}
```

Andererseits kann man auch alle Zustände ausschließen, von denen man weiß, dass sie nicht mehr eintreten. Beispielsweise kann beim Knoten D1 Zustand 1 mit dem Wert 0 und Zustand 0 und Zustand 2 mit dem Wert 1 belegt werden, d. h. evidenceD1_vr = $\{1.0, 0.0, 1.0\}$. Ebenso können die Zustände eines Knotens nicht nur mit den Werten 0.0 und 1.0 sondern auch mit Werten zwischen 0.0 und 1.0 belegt werden.

**Funktionen für die beiden Phasen Collect und Distribute Evidence** Für jeden inneren Knoten des arithmetischen Schaltkreises werden besondere Methoden zur Auswertung (Aufwärtsdurchgang) und Ableitung (Abwärtsdurchgang)

des arithmetischen Schaltkreises generiert. Wenn der arithmetische Schaltkreis ausgewertet wird, wird beispielsweise die Variable cptClique_C1_B1_vr berechnet, indem die beiden Variablen cptC1_vr und evidenceC1_vr miteinander multipliziert werden (siehe Abbildung 6.1). Der dazugehörige Quellcode sieht wie folgt aus:

```
// Multiply Clique: cptClique_C1_B1
public void collect_cptClique_C1_B1_vr() {
  for (counterC1 = 0; counterC1 < 2; counterC1++) {
    for (counterB1 = 0; counterB1 < 2; counterB1++) {
      cptClique_C1_B1_vr[counterC1][counterB1] =
        cptC1_vr[counterB1][counterC1] *
        evidenceC1_vr[counterC1];
    }
  }
  return;
}
```

Wie man am Quellcode erkennen kann, wird während der Multiplikation darauf geachtet, dass nur konsistente Tabelleneinträge verarbeitet werden.

Zwei benachbarte Cliquen tauschen sich untereinander durch einen Separator-Knoten aus, der beide miteinander verbindet und ihre gemeinsamen Knoten enthält, so dass nach einer Berechnung für jeden einzelnen Knoten in jeder Clique und jedem Separator dieselbe Information gespeichert ist. Separator cpt2_6 befindet sich zwischen den beiden Cliquen Clique_A0_B0_A1_B1 und Clique_A1_B1. Diese zwei Cliquen haben die Knoten A1 und B1 gemeinsam, so dass der Separator die Knoten A1 und B1 enthält. Die für die Knoten A1 und B1 relevante Information wird aus der Clique Clique_A0_B0_A1_B1 extrahiert und normalisiert, so dass sich die Zahlen zu Eins addieren, wie im folgenden Quellcode zu sehen ist:

```
// Marginalize Separator: cpt2_6
public void collect_cpt2_6_vr() {
  for (counterA1 = 0; counterA1 < 2; counterA1++) {
    for (counterB1 = 0; counterB1 < 2; counterB1++) {
      cpt2_6_vr[counterA1][counterB1] = 0.0;
      for (counterA0 = 0; counterA0 < 2; counterA0++) {
        for (counterB0 = 0; counterB0 < 2; counterB0++) {
          cpt2_6_vr[counterA1][counterB1] +=
          cptClique_A0_B0_A1_B1_vr
            [counterA0][counterB0][counterA1][counterB1];
        }
      }
```

```
    }
  }
  normalize = 0;
  for (counterA1 = 0; counterA1 < 2; counterA1++) {
    for (counterB1 = 0; counterB1 < 2; counterB1++) {
      normalize += cpt2_6_vr[counterA1][counterB1];
    }
  }
  for (counterA1 = 0; counterA1 < 2; counterA1++) {
    for (counterB1 = 0; counterB1 < 2; counterB1++) {
      cpt2_6_vr[counterA1][counterB1] /= normalize;
    }
  }

  return;
}
```

**Inferenz** Diese Funktionen zur Auswertung und Ableitung des arithmetischen Schaltkreises müssen in einer gewissen Reihenfolge aufgerufen werden, welches die Funktion initialize() sicherstellt. Der folgende Beispiel-Quellcode zeigt die Reihenfolge der Funktionsaufrufe für die graphische Darstellung des Polynoms in Abbildung 6.1:

```
/** Inference in the dynamic Bayesian network */
public void initialize() {
  collect_cpt2_2_vr();
  collect_cptClique_D1_B1_vr();
  collect_cpt2_4_vr();
  collect_cptClique_C1_B1_vr();
  collect_cpt2_5_vr();
  collect_cptClique_A0_B0_A1_B1_vr();
  collect_cpt2_6_vr();
  collect_cptClique_A1_B1_vr();
  distribute_cptClique_A1_B1_dr();
  distribute_cpt2_6_dr();
  distribute_cptClique_A0_B0_A1_B1_dr();
  distribute_cpt2_5_dr();
  distribute_cptClique_C1_B1_dr();
  distribute_cpt2_4_dr();
  distribute_cptClique_D1_B1_dr();
  distribute_cpt2_2_dr();

  return;
```

```
}
```

So muss nur die Funktion initialize() aufgerufen werden, um die Inferenz im dynamischen Bayesschen Netz durchzuführen. Im Anschluss daran lassen sich die BEL-Werte der interessierenden Knoten mit den dazugehörigen Funktionen auslesen, die durch den Quellcode zur Verfügung gestellt werden.

**Rollup** Der Rollup alter Zeitscheiben wird durchgeführt, indem die Tabelleneinträge der Clique, die als Schnittstelle zur nachfolgenden Zeitscheibe fungiert (hier cptClique_A1_B1), in die Tabelle der Clique eingetragen werden, die als Schnittstelle zur vorhergehenden Zeitscheibe fungiert (hier cptClique_A0_B0):

```
// Rollup PreBeliefClique: cptClique_A0_B0
public void rollup() {
  for (counterA0 = 0; counterA0 < 2; counterA0++) {
    for (counterB0 = 0; counterB0 < 2; counterB0++) {
      cptClique_A0_B0_vr[counterA0][counterB0] =
      cptClique_A1_B1_vr[counterA0][counterB0];
} } }
```

Neue Evidenz kann nach einem Rollup eingegeben werden, so dass die neuen Einschätzungen berechnet werden können, wobei die vorhergehenden (aufgerollten) Zeitscheiben und ihr Einfluss ohne Approximation berücksichtigt wird.

**Systemeinbindung** Das soeben vorgestellte dynamische Bayessche Netz lässt sich nun auf einfache Art und Weise in vorhandene Anwendungen einbetten, wie es im folgenden gezeigt wird:

```
import JavaDBN.EmbeddedJavaDBN;

public class Example {
    [...]
    public static void main(String args[]) {
        [...]
        ABCD-DBN-Polynomial jDBN =
            new ABCD-DBN-Polynomial();
        [...]
        while([...]) {
            [...]
            jDBN.setEvidence_D1_State1();
            [...]
            jDBN.initialize();
```

```
            [...]
            jDBN.rollup();
            [...]
        }
        [...]
        return;
    }
}
```

### 6.1.2 Vorteile des Quellcodes

Der Hauptvorteil des erzeugten Quellcodes liegt darin, dass er nicht nur das dynamische Bayessche Netz repräsentiert sondern auch die Inferenz mit dem Rollup alter Zeitscheiben im dynamischen Bayesschen Netz ermöglicht, wie es im vorhergehenden Abschnitt dargestellt wurde. Da zur Laufzeit keine Variablen allokiert oder deallokiert werden, wird keine automatische Speicherbereinigung aufgerufen werden. Dadurch lässt sich der Ressourcenbedarf bezüglich Speicher und Rechenzeit eindeutig abschätzen bzw. bestimmen.

Ein weiterer großer Vorteil des Quellcodes liegt darin, dass er leicht lesbar ist, so dass das Verfahren der Inferenz und des Rollups für den Anwender transparent bleibt. Wie wir im folgenden zeigen werden, lassen sich dadurch Veränderungen im Quellcode vornehmen, so dass sich Zusatzfunktionen wie ein Datenbankzugriff oder allgemeiner ein Methodenaufruf realisieren lässt, der nicht oder nur sehr schwer möglich wäre, wenn der Inferenzalgorithmus vorkompiliert ist und das dynamische Bayessche Netz zur Verarbeitung als Eingabe erhält. Wir wollen nun beispielhaft anhand des Knotens `B1` des Beispielnetzes (siehe Abbildung 6.1) zeigen, wie sich konstante zu variablen Einträgen in der Tabelle bedingter Wahrscheinlichkeiten umwandeln lassen, indem Arrays durch Methodenaufrufe ersetzt werden.

Dies ist z. B. dann sinnvoll, wenn Einträge in der Tabelle bedingter Wahrscheinlichkeiten im Voraus nicht bekannt sind oder sich zur Laufzeit verändern können. Ein Methodenaufruf könnte beispielsweise einen Datenbankzugriff realisieren (um beispielsweise die Einträge für die Tabelle bedingter Wahrscheinlichkeiten aus einem Benutzerprofil einzulesen) oder von der abgelaufenen Zeit seit der letzten Instantiierung abhängig sein. Im folgenden werden die Variablen gezeigt, die für den Knoten erzeugt wurden:

```
public int counterB1;
public double[] beliefB1_vr = {1.0, 1.0};
public double[] evidenceB1_vr = {1.0, 1.0};
public double[][][][] cptB1_vr =
{{{{0.85, 0.15}, {0.75, 0.25}},
```

```
  {{0.65, 0.35}, {0.55, 0.45}}}, [...] };
public double[] evidenceB1_dr = {1.0, 1.0};
public double cptB1_dr[][][][] = new double[2][2][2][2];
```

Der unterstrichene Programmteil, muss durch den folgenden Quellcode ersetzt werden:

```
public double cptB1_vr(int
    counterA0, counterB0, counterA1, counterB1; [...]) {
  double value = [...];
  return(value);
}
```

Wie sich der Wert berechnet, der von dieser Methode zurückgegeben wird, ist von der Problemstellung abhängig.

In allen Methoden, die auf das Array cptB1_vr zugreifen, muss nun der Arrayzugriff durch den entsprechenden Funktionsaufruf ersetzt werden, der in unserem Beispiel wie folgt lautet:

```
cptB1_vr(counterA0, counterB0,
         counterA1, counterB1, [...])
```

In der nachfolgenden Methode ist also der unterstrichene Text durch den vorhergehenden Methodenaufruf zu ersetzen:

```
// Multiply Clique: cptClique_A0_B0_A1_B1
public void collect_cptClique_A0_B0_A1_B1_vr() {
  for (counterA0 = 0; counterA0 < 2; counterA0++) {
    for (counterB0 = 0; counterB0 < 2; counterB0++) {
      for (counterA1 = 0; counterA1 < 2; counterA1++) {
        for (counterB1 = 0; counterB1 < 2; counterB1++) {
          cptClique_A0_B0_A1_B1_vr
            [counterA0][counterB0]
            [counterA1][counterB1] =
          cptA1_vr[counterA0][counterA1] *
          evidenceA1_vr[counterA1] *
          cptB1_vr
            [counterA0][counterB0]
            [counterA1][counterB1] *
          evidenceB1_vr[counterB1] *
          cpt2_5_vr[counterB1] *
          cpt2_4_vr[counterB1] *
          cpt2_2_vr[counterA0][counterB0];
  } } } }
  return;
}
```

Im folgenden Abschnitt werden nun einige Anwendungsmöglichkeiten der Sensitivitätsanalyse an einem Beispiel konkret vorgestellt.

### 6.1.3 Quellcode für Sensitivitätsanalyse

Für die Sensitivitätsanalyse wollen wir das Bayesssches Netz in Abbildung 2.1 mit den Wahrscheinlichkeiten in Tabelle 2.1 aufgreifen. Dazu haben wir zur besseren Lesbarkeit das Bayessche Netz in Abbildung 6.2 mit den Wahrscheinlichkeiten in Tabelle 6.1 erneut dargestellt.

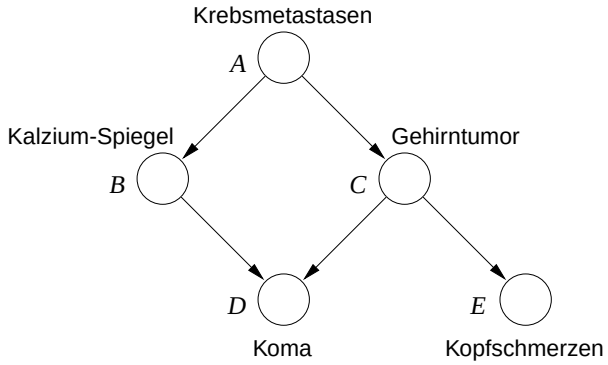



Abbildung 6.2: Beispielnetz Gehirntumor

**A-priori Parametrisierung** Wir wollen nun bestimmen, wie die BEL-Werte der Knoten im Netz durch die A-priori-Wahrscheinlichkeit des Knotens A beeinflusst werden, wenn keine Evidenz vorliegt. Dazu sei die A-priori-Wahrscheinlichkeit des Knotens $A$ nun nicht mehr $\Pr(A) = (0.20, 0.80)$ sondern $\Pr(A) = (A1, A2)$. Nach der Initialisierung des Bayesschen Netzes kann man nun anhand der BEL-Werte der einzelnen Knoten den Einfluss der A-priori-Wahrscheinlichkeit ablesen:

```
syms alpha A1 A2; A2 = 1 - A1; alpha = 1/(A1 + A2);
A = [A1; A2];
B = alpha * [0.8 * A1 + 0.2 * A2;
             0.2 * A1 + 0.8 * A2]
C = alpha * [0.2 * A1 + 0.05 * A2;
             0.8 * A1 + 0.95 * A2]
D = alpha * [0.74 * A1 + 0.245 * A2;
             0.26 * A1 + 0.755 * A2]
E = alpha * [0.64 * A1 + 0.61 * A2;
             0.36 * A1 + 0.39 * A2]
x = 0:1/10:1;
```

| | | |
|---|---|---|
| Krebsmetastasen (nein, ja) | | |
| $\Pr(a)$: | $\Pr(a_1) = 0.80$ | $\Pr(a_2) = 0.20$ |
| Kalzium-Spiegel (normal, erhöht) | | |
| $\Pr(b \mid a)$: | $\Pr(b_1 \mid a_1) = 0.80$ | $\Pr(b_2 \mid a_1) = 0.20$ |
| | $\Pr(b_1 \mid a_2) = 0.20$ | $\Pr(b_2 \mid a_2) = 0.80$ |
| Gehirntumor (nein, ja) | | |
| $\Pr(c \mid a)$: | $\Pr(c_1 \mid a_1) = 0.95$ | $\Pr(c_2 \mid a_1) = 0.05$ |
| | $\Pr(c_1 \mid a_2) = 0.80$ | $\Pr(c_2 \mid a_2) = 0.20$ |
| Koma (nicht im Koma, komatös) | | |
| $\Pr(d \mid b,c)$: | $\Pr(d_1 \mid b_1,c_1) = 0.95$ | $\Pr(d_2 \mid b_1,c_1) = 0.05$ |
| | $\Pr(d_1 \mid b_1,c_2) = 0.20$ | $\Pr(d_2 \mid b_1,c_2) = 0.80$ |
| | $\Pr(d_1 \mid b_2,c_1) = 0.20$ | $\Pr(d_2 \mid b_2,c_1) = 0.80$ |
| | $\Pr(d_1 \mid b_2,c_2) = 0.20$ | $\Pr(d_2 \mid b_2,c_2) = 0.80$ |
| Kopfschmerzen (keine, starke) | | |
| $\Pr(e \mid c)$: | $\Pr(e_1 \mid c_1) = 0.40$ | $\Pr(e_2 \mid c_1) = 0.60$ |
| | $\Pr(e_1 \mid c_2) = 0.20$ | $\Pr(e_2 \mid c_2) = 0.80$ |

Tabelle 6.1: Bedingte Wahrscheinlichkeiten bzw. A-priori-Wahrscheinlichkeiten für die Knoten des Beispielnetzes Gehirntumor in Abbildung 6.2.

```
plot(x, subs(A(1), x), x, subs(B(1), x), [...]);
```

Wenn man den Quellcode in MatLab einliest, erhält man die folgenden Ergebnisse für die Werte der Knoten B, C, D und E und ihre Visualisierung wie in Abbildung 6.3 dargestellt wird.

```
B = [  3/5 * A1 + 1/5]          C = [  3/20 * A1 + 1/20]
    [ -3/5 * A1 + 4/5]              [ -3/20 * A1 + 19/20]
D = [  99/200 * A1 + 49/200]    E = [  3/100 * A1 + 61/100]
    [ -99/200 * A1 + 151/200]       [ -3/100 * A1 + 39/100]
```

Der BEL-Wert eines jeden Knotens ist somit linear abhängig von A1. Die Knoten C und E werden am wenigsten durch die A-priori-Wahrscheinlichkeit des Knotens A beeinflusst. Knoten E hat einen maximalen Wert von sogar nur 0.03. Die Winkelhalbierende stellt die maximale Abhängigkeit dar, so dass eine Gerade mit höherer Steigung nicht möglich ist. Wie man sehen kann, vereinfacht MatLab die Werte der Knoten, indem die Variable A2 mit ihrem Wert 1 - A1 und die Variable alpha mit ihrem Wert 1/(A1 + A2) belegt wird.

In den nächsten Beispielen werden die Wahrscheinlichkeitentabellen der Knoten parametrisiert.

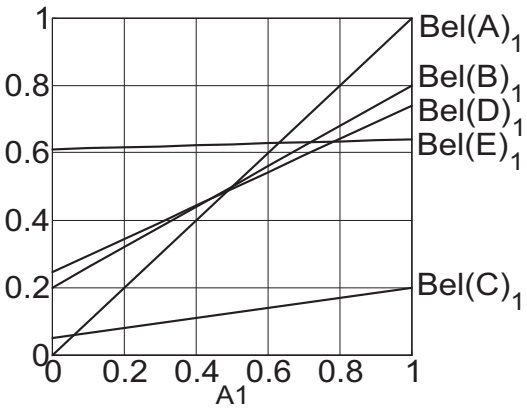



Abbildung 6.3: Der BEL-Wert in Abhängigkeit von $A1$ beschreibt eine Gerade

**CPT-Parametrisierung** Wir wollen nun die bedingten Wahrscheinlichkeiten des Knotens $C$ des Bayesschen Netzes in Abbildung 6.2 parametrisieren. Die A-priori- und bedingten Wahrscheinlichkeiten seien wieder so, wie sie in Tabelle 6.1 angegeben sind. Die Tabelle bedingter Wahrscheinlichkeiten des Knotens $C$ setzen wir auf

$$\mathrm{CPT}(C) = \theta_{(C|A)} = \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline a_1 & X1 & X2 \\ a_2 & Y1 & Y2 \end{array}\right].$$

Es gilt: $X2 = 1 - X1$ und $Y2 = 1 - Y1$. Dieses wollen wir jedoch noch nicht ausnutzen, damit wir sehen können, wie sich Knoten mit mehr als zwei Hypothesen auf das Netz auswirken.

Nach der Initialisierung des Bayesschen Netzes können wir anhand der BEL-Werte der Knoten den Einfluss der bedingten Wahrscheinlichkeiten des Knotens $C$ ablesen:

$$\begin{aligned}
\mathrm{BEL}(A) &= \alpha \begin{pmatrix} 0.20\,X1 + 0.20\,X2 \\ 0.80\,Y1 + 0.80\,Y2 \end{pmatrix} \\
\mathrm{BEL}(B) &= \alpha \begin{pmatrix} 0.16\,X1 + 0.16\,X2 + 0.16\,Y1 + 0.16\,Y2 \\ 0.04\,X1 + 0.04\,X2 + 0.64\,Y1 + 0.64\,Y2 \end{pmatrix} \\
\mathrm{BEL}(C) &= \alpha \begin{pmatrix} 0.20\,X1 + 0.80\,Y1 \\ 0.20\,X2 + 0.80\,Y2 \end{pmatrix} \\
\mathrm{BEL}(D) &= \alpha \begin{pmatrix} 0.156\,X1 + 0.146\,X2 + 0.576\,Y1 + 0.176\,Y2 \\ 0.044\,X1 + 0.054\,X2 + 0.224\,Y1 + 0.624\,Y2 \end{pmatrix} \\
\mathrm{BEL}(E) &= \alpha \begin{pmatrix} 0.16\,X1 + 0.12\,X2 + 0.64\,Y1 + 0.48\,Y2 \\ 0.04\,X1 + 0.08\,X2 + 0.16\,Y1 + 0.32\,Y2 \end{pmatrix}.
\end{aligned}$$

Unter der Voraussetzung von $X1+X2=1$ und $Y1+Y2=1$ erhalten wir:

$$\begin{aligned}
\mathrm{BEL}(A) &= \begin{pmatrix} 0.20 \\ 0.80 \end{pmatrix} \\
\mathrm{BEL}(B) &= \begin{pmatrix} 0.32 \\ 0.68 \end{pmatrix} \\
\mathrm{BEL}(C) &= \begin{pmatrix} 0.20\,X1+0.80\,Y1 \\ 0.20\,X2+0.80\,Y2 \end{pmatrix} \\
\mathrm{BEL}(D) &= \begin{pmatrix} 0.156\,X1+0.146\,X2+0.576\,Y1+0.176\,Y2 \\ 0.044\,X1+0.054\,X2+0.224\,Y1+0.624\,Y2 \end{pmatrix} \\
\mathrm{BEL}(E) &= \begin{pmatrix} 0.16\,X1+0.12\,X2+0.64\,Y1+0.48\,Y2 \\ 0.04\,X1+0.08\,X2+0.16\,Y1+0.32\,Y2 \end{pmatrix}.
\end{aligned}$$

Wie wir sehen, werden die Knoten $A$ und $B$ von $X1$ und $Y1$ nicht beeinflusst. Für die anderen Knoten werden wir im folgenden ausnutzen, dass $X2=1-X1$ und $Y2=1-Y1$ gilt. Wir erhalten:

$$\begin{aligned}
\mathrm{BEL}(C) &= \begin{pmatrix} 0.20\,X1+0.80\,Y1 \\ 0.20\,(1-X1)+0.80\,(1-Y1) \end{pmatrix} = \begin{pmatrix} 0+0.20\,X1+0.80\,Y1 \\ 1-0.20\,X1-0.80\,Y1 \end{pmatrix} \\
\mathrm{BEL}(D) &= \begin{pmatrix} 0.156\,X1+0.146\,(1-X1)+0.576\,Y1+0.176\,(1-Y1) \\ 0.044\,X1+0.054\,(1-X1)+0.224\,Y1+0.624\,(1-Y1) \end{pmatrix} = \\
&= \begin{pmatrix} 0.322+0.01\,X1+0.4\,Y1 \\ 0.678-0.01\,X1-0.4\,Y1 \end{pmatrix} \\
\mathrm{BEL}(E) &= \begin{pmatrix} 0.16\,X1+0.12\,(1-X1)+0.64\,Y1+0.48\,(1-Y1) \\ 0.04\,X1+0.08\,(1-X1)+0.16\,Y1+0.32\,(1-Y1) \end{pmatrix} = \\
&= \begin{pmatrix} 0.60+0.04\,X1+0.16\,Y1 \\ 0.40-0.04\,X1-0.16\,Y1 \end{pmatrix}.
\end{aligned}$$

Wir erkennen, dass die BEL-Werte der Knoten $C$, $D$ und $E$ linear von $X1$ und $Y1$ abhängig sind und eine Ebene im Raum aufspannen. Dieses ist in Abbildung 6.4 dargestellt. $X1$ hat im Vergleich zu $Y1$ auf den BEL-Wert der Knoten $C$, $D$ und $E$ nur einen geringen Einfluss. Bei den Knoten $D$ und $E$ ist er sogar vernachlässigbar klein.

Will man nun eine bestimmte Wahrscheinlichkeitentabelle für den Knoten $C$ modellieren, so lässt sich ein lineares Gleichungssystem aufstellen. Kennt man die BEL-Werte der Knoten $D$ und $E$ (z. B. seien $\mathrm{BEL}(D) = \begin{pmatrix} 0.344 \\ 0.656 \end{pmatrix}$ und $\mathrm{BEL}(E)$

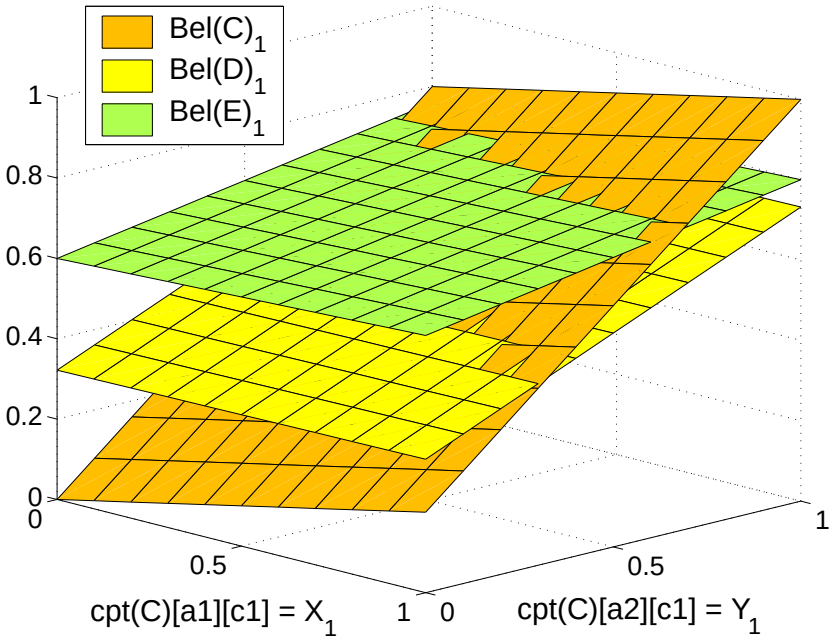



Abbildung 6.4: Die BEL-Werte in Abhängigkeit von $X1$ und $Y1$ beschreiben jeweils eine Ebene.

$= \begin{pmatrix} 0.616 \\ 0.384 \end{pmatrix}$), so erhält man:

$$0.344 = 0.322 + 0.01\,X1 + 0.40\,Y1 \Leftrightarrow 0.022 = 0.01\,X1 + 0.40\,Y1 \quad \text{(I)}$$
$$0.656 = 0.678 - 0.01\,X1 - 0.40\,Y1 \Leftrightarrow 0.022 = 0.01\,X1 + 0.40\,Y1$$
$$0.616 = 0.600 + 0.04\,X1 + 0.16\,Y1 \Leftrightarrow 0.016 = 0.04\,X1 + 0.16\,Y1 \quad \text{(II)}$$
$$0.384 = 0.400 - 0.04\,X1 - 0.16\,Y1 \Leftrightarrow 0.016 = 0.04\,X1 + 0.16\,Y1.$$

Dies lässt sich mit dem Gaußschen Algorithmus (siehe [Lamprecht 78]) lösen, und man erhält:
mit (II) := (II) - 4 * (I):

$$\begin{aligned} 0.022 &= 0.01\,X1 + 0.40\,Y1 && \text{(I)} \\ -0.072 &= -1.44\,Y1 && \text{(II)} \end{aligned}$$

und (II) := (II)/(−1.44):

$$\begin{aligned} 0.022 &= 0.01\,X1 + 0.40\,Y1 && \text{(I)} \\ 0.05 &= Y1 && \text{(II)} \end{aligned}$$

Einsetzen von (II) in (I) und Umformen von (I) liefert:

$$\begin{aligned} 0.20 &= X1 && \text{(I)} \\ 0.05 &= Y1 && \text{(II).} \end{aligned}$$

Die Tabelle bedingter Wahrscheinlichkeiten des Knoten $C$ bestimmt sich nun zu

$$\theta_{(C|A)} = \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline a_1 & 0.20 & 0.80 \\ a_2 & 0.05 & 0.95 \end{array}\right].$$

**Sensitivitätsanalyse** Bei der Sensitivitätsanalyse wird ein Eintrag einer Wahrscheinlichkeitentabelle systematisch verändert. Die anderen Einträge der Wahrscheinlichkeitentabelle werden an diese Veränderung (prozentual ihres Anteils) angepasst, so dass sich die entsprechenden Einträge zu eins addieren.

Analog zu [van der Gaag & Coupé 98] und [Kjærulff & van der Gaag 00] erhält man linear abhängige Ausdrücke bezüglich der parametrisierten Einträge der Wahrscheinlichkeitentabellen. Unser Verfahren erhält die linear abhängigen Ausdrücke jedoch direkt als Funktion aus der Inferenz mit den parametrisierten Einträgen der Wahrscheinlichkeitentabellen. Ebenso lassen sich mit ihrem Verfahren nur einzelne Parameter betrachten, wohingegen mit unserem Verfahren beliebig viele Parameter betrachtet werden können. In Abbildung 6.4 sind beispielsweise die BEL-Werte der Knoten $C$, $D$ und $E$ in Abhängigkeit zweier Parameter dargestellt.

## 6.2 Zusammenfassung

Das Softwarewerkzeug JavaDBN ermöglicht den Einsatz dynamischer Bayesscher Netze auf eingebetteten Systemen mit geringer Leistung, indem effizienter Quellcode erzeugt wird. Dieser Quellcode kann einfach angepasst werden, so dass beispielsweise Einträge in einer Tabelle bedingter Wahrscheinlichkeiten als zeitveränderliche Funktionsaufrufe realisiert werden können. Sensitivitätsanalysen sind eine weitere Zusatzfunktion, die durch dieses Werkzeug ermöglicht werden.

# Kapitel 7

# Anwendungen

Im folgenden werden exemplarisch Anwendungen vorgestellt, die mit dem eben vorgestellten Verfahren erstellt wurden. Dabei werden in den folgenden Anwendungen die verschiedenen Sensortypen und ihre Fusion demonstriert.

- **simulierte Biosensoren (Sprache und motorisches Eingabeverhalten):** Wir stellen das dynamische Bayessche Netz zur Behandlung von motorischen und sprachlichen Symptomen zur Einschätzung der aktuellen Benutzersituation bezüglich Zeitdruck und kognitiver Belastung (Projekt READY) vor. Dabei werden die Symptome nicht durch biophysiologische Sensoren und ihre Klassifizierer gemessen sondern über eine graphische Benutzeroberfläche spezifiziert, d. h. das dynamische Bayessche Netze wird ereignisgesteuert instantiiert. Wir zeigen weiterhin, dass sich dynamische Bayessche Netze zur Einschätzung der Ressourcenlage eines Benutzers anhand motorischer und sprachlicher Symptome auf eingebetteten Systemen einsetzen lassen.

- **Biosensoren (ACC, EMG, EOG):** Das prototypische System Alarm Manager arbeitet mit Biosensoren (Beschleunigungssensor (ACC-Sensor), Muskeltätigkeitssensor (EMG-Sensor), Augenbrauensensor (EOG-Sensor)), die ein dynamisches Bayessches Netz mit Evidenz versorgen, um festzustellen, ob der Benutzer einen Alarm wahrgenommen hat. Die Einschätzungen des dynamischen Bayesschen Netzes dienen zur Steuerung des Alarm-Levels. Das dynamische Bayessche Netz wird hierbei zeitgesteuert instantiiert.

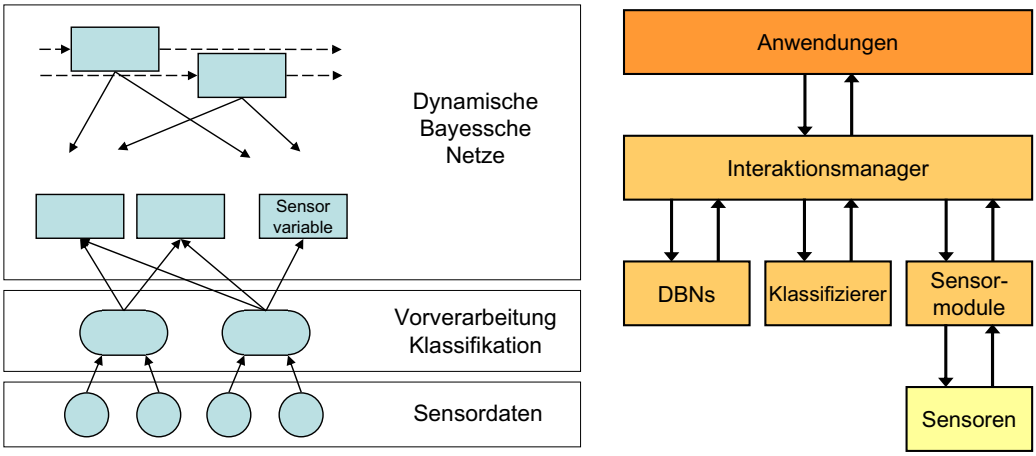
- **Umgebungssensoren (RFID, Infrarot):** Das multisensorielle Positionierungssystem LORIOT läuft auf einem persönlichen digitalen Assistenten und arbeitet mit Umgebungssensoren (RFID- und Infrarot-Sensoren), um einen Benutzer zu positionieren. Die Umgebung ist dabei mit RFID-Tags

und IR-Sendern ausgestattet. Jeder gemessene RFID-Tag und jede gemessene IR-Bake induziert eine Geokoordinate mit einem dynamischen Bayesschen Netz, so dass insgesamt mehrere dynamische Bayessche Netze (geo-DBNs) evaluiert werden, um die Position des Benutzers einzuschätzen. Das dynamische Bayessche Netz lässt sich einfach erweitern, so dass weitere verschiedene Sensoren (wie z. B. GPS-Sensor) integriert werden können. Zusätzlich wird mit einem weiteren Netz die Orientierung des Benutzers berechnet. Hier wird der bisher zurückgelegte Weg und gemessene IR-Baken (die eine Orientierung ausstrahlen) berücksichtigt. Die dynamischen Bayesschen Netze werden hierbei zeitgesteuert instantiiert.

- **Bio- und Umgebungssensoren:** In einer weiteren Arbeit wurde das multisensorielle Positionierungssystem LORIOT um einen Biosensor (Beschleunigungssensor) erweitert. Die durch den Beschleunigungssensor gemessene Schrittfrequenz wird dazu verwendet, um die bedingten Wahrscheinlichkeiten in der Übergangswahrscheinlichkeiten des dynamischen Knotens anzupassen. Diese bedingten Wahrscheinlichkeiten werden durch eine Funktion parametrisiert, die wie folgt arbeitet: Ist keine Schrittfrequenz feststellbar, so führen die bedingten Wahrscheinlichkeiten keine Abschwächung der vorhergehenden Zeitscheibe durch. Umso höher die gemessene Schrittfrequenz des Benutzers ist, umso stärker wird die Abschwächung durch die bedingten Wahrscheinlichkeiten. Die dynamischen Bayesschen Netze werden hierbei zeitgesteuert instantiiert.

- **Bio- und Umgebungssensor (Mikrofon):** Agender verwendet Sprachsymptome zur Einschätzung von Alter und Geschlecht des Anwenders. Als Sensor dient hier ein einfaches Mikrofon (Biosensor). Weiterhin dient dasselbe Mikrofon zur Einschätzung des Hintergrundlärms (Umgebungssensor). Je nachdem, ob der Hintergrundlärm stimmhaft oder stimmlos ist, werden verschiedene Klassifizierer ausgewählt, um unter den möglichen Lärmbedingungen das beste Ergebnis zu erreichen. Das dynamische Bayessche Netz wird ereignisgesteuert instantiiert.

Die Beziehung zwischen Sensordaten, ihrer Vorverarbeitung und Klassifizierung sowie die Instantiierung der Ergebnisse als Evidenz in einem dynamischen Bayesschen Netz ist in Abbildung 7.1(a) visualisiert. In Abbildung 7.1(b) wird gezeigt, wie die existierenden Komponenten in einem System integriert werden können, so dass Anwendungen Anfragen nur auf der Ebene des sogenannten Interaktionsmanagers zu stellen haben. Dieser Interaktionsmanager ist dann beispielsweise dafür verantwortlich, dass die unverarbeiteten Sensordaten an ihre zugehörigen Klassifizierer oder dynamischen Bayesschen Netze weitergeleitet werden. Ebenso muss der Interaktionsmanager den Rollup der dynamischen Bayesschen Netze




(a) Verarbeitungshierarchie (b) Systemarchitektur

Abbildung 7.1: Schematischer Überblick über den Datenfluss

mit dem Auslesen und dem Verarbeiten der Sensoren synchronisieren. Gleichfalls werden neue Klassifizierer und Sensoren beim Interaktionsmanager angemeldet, der ihren Zugriff abwickelt. So wird die Entwicklung von Anwendungen getrennt von der Erweiterung um Sensoren, Klassifizierer und dynamischen Bayesschen Netzen, auf die weiterhin unabhängig von allen anderen Modulen oder des Systems zugegriffen werden kann.

## 7.1 Motorische und sprachliche Symptome

Im Rahmen des READY-Projektes haben wir zu Demonstrationszwecken einen Prototypen entwickelt, der die Ressourcenlage eines Benutzers aufgrund seines symptomatischen Verhaltens mit dynamischen Bayesschen Netzen probabilistisch einschätzt.

Im folgenden stellen wir den Prototypen vor und zeigen, wie mit JavaDBN die Inferenz im dynamischen Bayesschen Netz und der Rollup alter Zeitscheiben realisiert wurde.

### 7.1.1 READY-Projekt

Die zentrale Problemstellung des Projekts READY[1] besteht in der Erkennung und Berücksichtigung von Beschränkungen der Ressourcenlage eines Benutzers durch ein mobiles Assistenzsystem, so dass Anweisungen und Informationen der Situation des Benutzers entsprechend präsentiert werden (s. [Bohnenberger et al. 02]). Insbesondere soll das Assistenzsystem die zeitlich veränderlichen Benutzereigenschaften wie *Arbeitsgedächtnisbelastung* und *subjektiven Zeitdruck* modellieren. Da diese Eigenschaften nicht direkt beobachtet werden können, müssen sie aufgrund indirekter Evidenz eingeschätzt werden. READY schätzt diese Ressourceneinschränkungen auf der Basis von *Symptomen* im Verhalten des Benutzers als auch auf der Basis *physiologischer Daten* probabilistisch ein.

Die Art der Unterstützung durch READY kann am einfachsten mit einem konkreten Beispiel verdeutlicht werden. Wir wollen dies am Beispiel eines Flughafenszenarios veranschaulichen.

In Abbildung 7.2 sind zwei Reisende an einem großen internationalen Flughafen dargestellt. Der erste Reisende hat sowohl Zeitdruck als auch ist er abgelenkt durch den sehr kurz bevorstehenden Abflug seines Flugzeuges. Für ihn ist sehr wahrscheinlich die zweite in der Abbildung dargestellte Präsentation geeignet. Da der zweite Reisende viel Zeit zu haben und aufmerksam zu sein scheint, dürfte die erste Präsentation für ihn am besten geeignet zu sein. Während der erste Reisende zum Flugsteig geführt wird, kann das System den zweiten Reisenden an ausgewählten Geschäften vorbei leiten, so dass er auf dem Weg zum Flugsteig noch einige nützliche Dinge erledigen kann.

In diesem Beispiel können die Unterschiede zwischen den beiden Reisenden hauptsächlich auf der Basis ihrer Spracheigenschaften wie die Länge ihrer Sprachäußerungen, die Anzahl der Pausen und die Artikulationsgeschwindigkeit erkannt werden. Die erste Person spricht schnell, laut und knapp, welches Anzeichen sowohl für den Zeitdruck als auch für die kognitive Belastung der Person sind.

[1] READY ist ein Akronym für **Re**ssourcen**a**daptives **D**ialogs**y**stem.

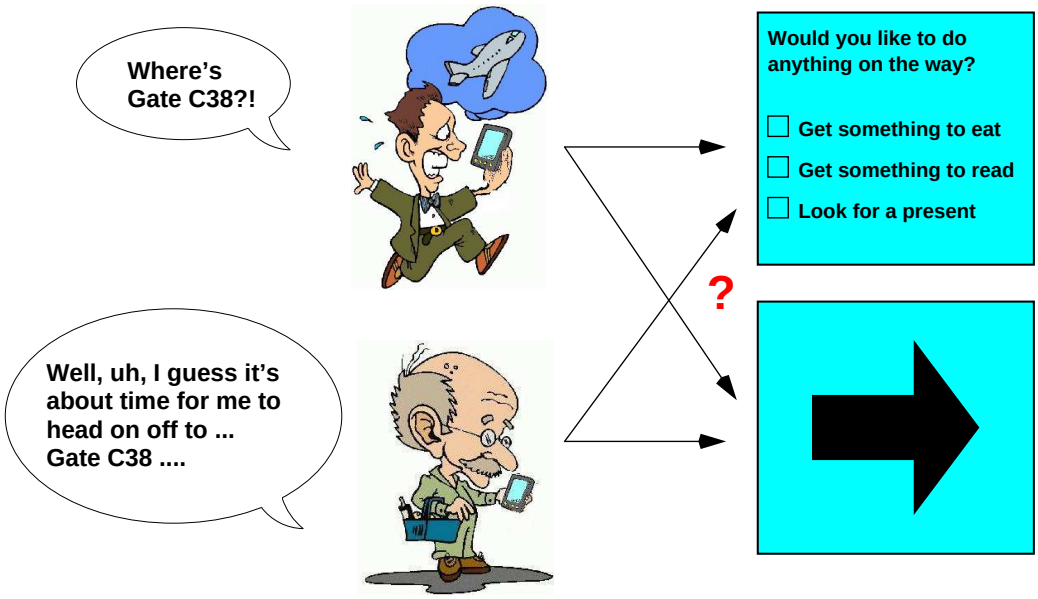



Abbildung 7.2: Beispiel wie das System READY sein Verhalten an die wahrgenommenen Ressourceneinschränkungen des Benutzers anpasst.

Evidenz dieser Art kann bei jeder sprachlichen Äußerung oder manuellen Eingabe des Benutzers erfasst werden. Im READY-Prototypen wird sie über eine grafische Benutzeroberfläche (siehe Abbildung 7.3) spezifiziert, die ein Handheld-Gerät auf einem PC-Bildschirm simuliert. So können wir die Verwendung einer breiten Vielfalt an Ein- und Ausgabearten simulieren. Der Prototyp enthält keine Komponenten wie echte Spracherkenner oder -erzeuger; hingegen müssen alle Ein- und Ausgaben über die graphische Benutzerschnittstelle mit ihren typischen Modalitäten (wie z. B. Menüs und graphische und textuelle Anzeige) spezifiziert werden.

In der oberen linken Ecke des Graphics / Text Fensters in Abbildung 7.3 zeigt die kleine Karte den Inhalt des simulierten PDA-Bildschirmes an. Die manuelle Eingabe in den PDA wird durch die Beschreibung ihres Inhaltes und die Art und Weise ihrer Eingabe erfasst. Zum Beispiel wurde in der rechten Ecke des Graphics / Text Fensters spezifiziert, dass dem Anwender eine Karte präsentiert wurde, in der er auf das Telefon klickte (Clicked). Darunter wurde beispielsweise angegeben, dass der Benutzer zunächst das Objekt beim Anklicken knapp verfehlte (Close). Die Spracheingabe wurde ähnlich realisiert, wie man am Sprachfenster (Speech) in der Abbildung 7.3 rechts erkennen kann. In unserem Beispiel äußerte sich der Benutzer mit dem Inhalt I'd like to use this... und einer Artikulationsgeschwindigkeit (Articulation rate) von 4 bis 8 Silben pro Sekunde (4–8 syll/sec).

Die manuelle Eingabe und die Spracheingabe sind zwei Teile einer multimo-


Abbildung 7.3: Bildschirmabzug von der Benutzerschnittstelle des READY-Prototypen.

dalen Äußerung, um einen Wunsch des Benutzers zu äußern. Die multimodale Ausgabe des Systems wurde analog durch das Interface angezeigt.

Die beiden kleinen Panelen, die in Abbildung 7.3 oben dargestellt sind (Intermodule communication und Monitoring - User modeling), zeigen den internen Zustand des Systems an. Insbesondere zeigt die rechte Panele die Einschätzung über den Zustand des Benutzers bezüglich Zeitdruck und Arbeitsgedächtnisbelastung an. Diese Einschätzung wurde durch das dynamische Bayessche Netz berechnet, das schematisch in Abbildung 7.5 dargestellt ist.

Im READY-Projekt wurde basierend auf zwei Experimenten (siehe Abbildung 7.4 (Experimente)) ein Bayessches Netz für die Interpretation von Sprachsymptomen gelernt (siehe [Müller et al. 01] und [Kiefer 02] und Abbildung 7.4 (Lernen der BNs)), während ein anderes Bayessches Netz für die Interpretation der Eigenschaften manuellen Eingabeverhaltens basierend auf einer umfassenden Literaturstudie (siehe [Lindmark 00]) erstellt wurde. Diese Bayesschen Netze wurden zu einem Zeitscheibenschema kombiniert, das als Vorlage für die Instantiierung der Zeitscheiben des dynamischen Bayesschen Netzes (siehe Abbildung 7.4 (Benutzermodellierung)) dient. Die Kombination dieser beiden Bayesschen Netze

erlaubt es READY nun auf der Basis multimodaler Eingaben zu schließen. In Abschnitt 7.1.2 wird das dynamische Bayessche Netz (siehe Abbildung 7.5) näher beschrieben.

Wie wir gesehen haben, kommen im READY-Prototypen verschiedene Komponenten zum Einsatz. In Abbildung 7.4 haben wir diese Komponenten und in welcher Art und Weise sie miteinander kommunizieren grafisch in einem Kuchendiagramm dargestellt. Die Kuchenstücke innerhalb des Kuchens repräsentieren die Online-Komponenten. Die Stücke, die aus dem Kuchen herausragen, repräsentieren Komponenten, die während der off-line stattfindenden Initialisierungsphase aktiv sind. In Experimente werden mittels Experimenten Informationen über das Verhalten und die Aktionen von Benutzern gesammelt. Die gewonnenen Daten werden sowohl für die Parametrisierung der entscheidungstheoretischen Dialogplanung (siehe [Bohnenberger 05]) als auch für das Lernen der Bayesschen Netze (siehe [Wittig 03]) für die individuelle Benutzermodellierung genutzt.

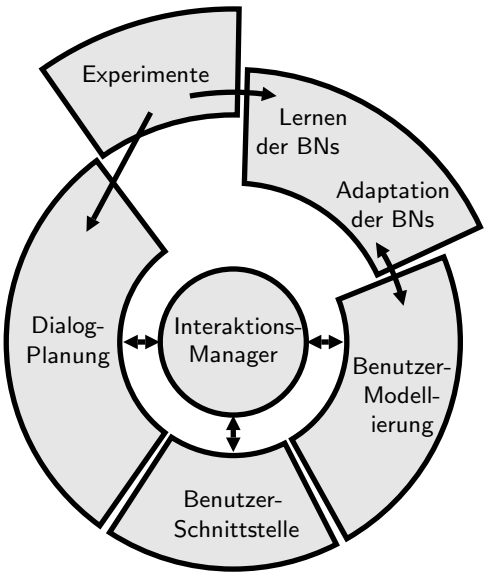



Abbildung 7.4: Systemarchitektur des READY-Prototyps. Die Pfeile stellen den Informationsfluss zwischen den Komponenten dar. Weitere Erläuterungen finden sich im Text.

Der Benutzer interagiert mit READY über die Benutzerinterschnittstelle, die die Eingabe an den Interaktionsmanager weiterleitet. Der Interaktionsmanager seinerseits führt eine erste ungefähre Aktions- und Antwort-Planung durch und versorgt die Benutzermodellierungs-Komponente mit den Informationen, wie der Benutzer seine Anfrage getätigt hat. Die Eingabeinformationen, d. h., was der An-

wender tun oder wissen möchte, werden an die Komponente Dialogplanung weitergeleitet. Die Kommunikation zwischen Benutzermodellierung und Dialogplanung wird durch den Interaktionsmanager erledigt. Insbesondere erhält die Dialogplanung von der Benutzermodellierung Informationen über den aktuellen Zustand des Benutzers. Die Bayesschen Netze für die Benutzermodellierung werden nach jeder Sitzung durch Adaption der BNs semi-on-line angepasst.

### 7.1.2 Benutzermodellierung

Wir wollen nun das dynamische Bayessche Netz genauer betrachten, das Symptome in Sprache und manueller Eingabe handhaben kann. In Abbildung 7.5 sind zwei Zeitscheiben dieses dynamischen Bayesschen Netzes dargestellt. Eine Zeitscheibe umfasst dabei an die 40 Knoten mit jeweils 2 bis 4 Hypothesen.

Schauen wir uns die Knoten in Zeitscheibe $N+1$ einmal genauer an. Die Knoten Cognitive Load und Time Pressure modellieren die Benutzereinschränkungen und sind als *dynamische Knoten* realisiert, da sie einen Einfluss auf die nächste Zeitscheibe haben. Diese Benutzereigenschaften variieren über die Zeit und können nicht direkt beobachtet werden. Sie müssen basierend auf den Merkmalen des Benutzerverhaltens wie Sprechpausen oder kräftiges Tippen auf dem Bildschirm eingeschätzt werden.

Sowohl die Knoten, die Verhaltensmerkmale repräsentieren, wie beispielsweise Filled Pauses, Disfluencies, Motor Error oder High Forces als auch die Knoten, die als Sensorknoten fungieren, wie z. B. die Knoten Difficulty of Speech Task und Difficulty of Motor Task wurden als *temporäre Knoten* modelliert.

Die *statischen Knoten* in der Abbildung modellieren die *typischen Eigenschaften* des Benutzers – z. B. wie oft dieser Benutzer typischerweise gefüllte Pausen (z. B. *uhm*) erzeugte. Diese Eigenschaften verändern sich nicht über die Zeit aber werden mit jeder neuen Äußerung bzw. Eingabe zunehmend genauer eingeschätzt. Durch ihre Modellierung als statische Knoten haben sie auf jede Zeitscheibe denselben Einfluss.

Zuerst wurde der Rollup nur approximiert durchgeführt, indem die vorhergehende Zeitscheibe abgeschnitten und die berechneten BEL-Werte der Knoten als A-priori-Wahrscheinlichkeiten in die Knoten der aktuellen Zeitscheibe eingetragen wurden. Dieser Rollup war naturbedingt fehlerbehaftet. Ohne Rollup merkte man jedoch schon nach ca. drei Zeitscheiben eine deutliche Antwortlatenz des Systems, da nach jedem Anhängen eine neue Eliminationsreihenfolge berechnet und ein neuer Junction Tree bestimmt werden mussten, in dem dann die gesammelten Evidenzen eingetragen wurden. Erst danach konnte die Inferenz über dem gesamten Junction Tree durchgeführt werden. Durch das in dieser Arbeit vorgestellte Verfahren konnten wir einen Rollup ohne Approximation realisieren. Wir ließen den von JavaDBN erzeugten Quellcode sowohl auf einem Laptop als auch

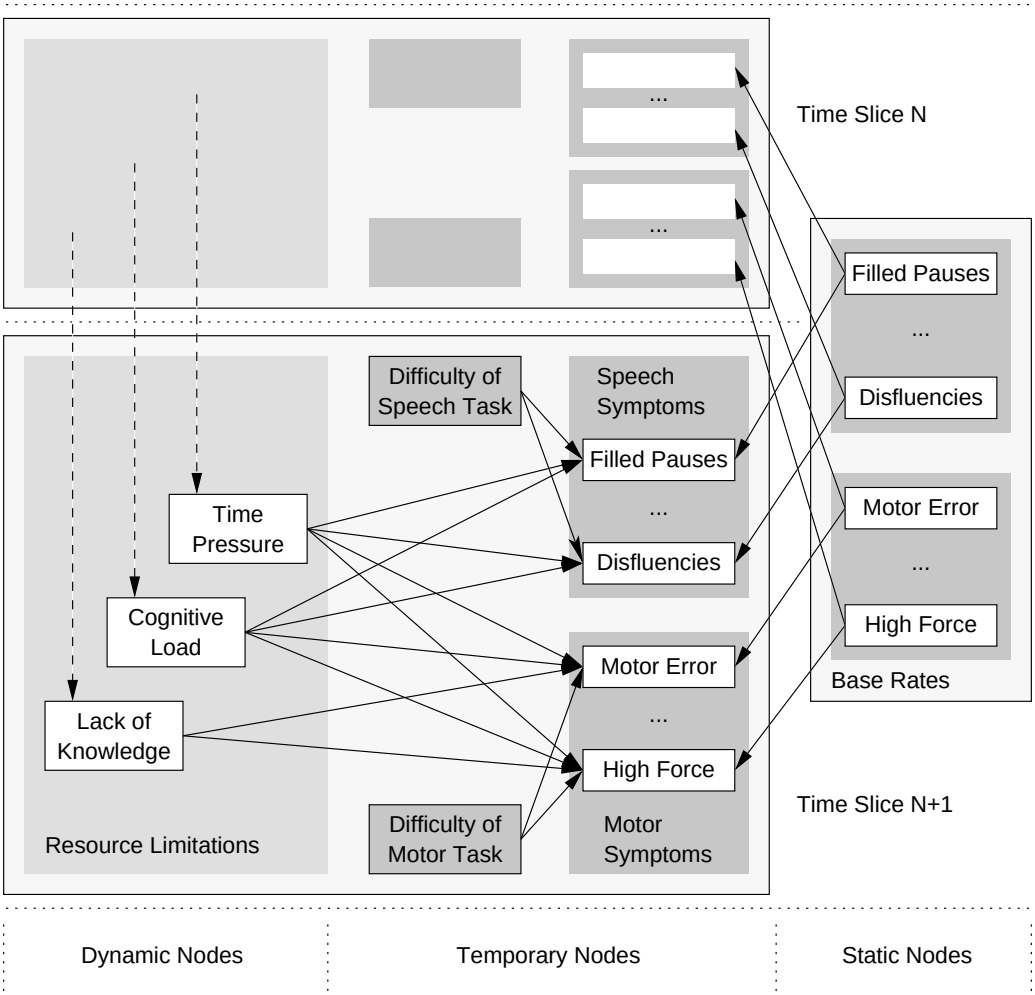



Abbildung 7.5: Übersicht eines dynamischen Bayesschen Netzes für die Erkennung von Benutzereinschränkungen wie Zeitdruck (Time Pressure) und kognitive Belastung (Cognitive Load) anhand sprachlicher und motorischer Symptome (Speech Symptoms und Motor Symptoms).

auf einem Zaurus SL-5500G laufen. Auf dem Laptop wurden dabei 25.000 Zeitscheiben und auf dem Zaurus SL-5500G 500 Zeitscheiben innerhalb einer Minute mit exaktem Rollup verarbeitet. Dabei wurde die simulierte multimodale Eingabe nicht verwendet und immer wieder dieselbe Evidenz eingetragen.

## 7.2 Alarm Manager

Der Alarm Manager ist ein prototypisches System zur Demonstration wie physiologische Daten eines Benutzers durch physiologische Sensoren erfasst und durch dynamische Bayessche Netze auf einem persönlichen digitalen Assistenten (PDA) mit geringen Ressourcen an Rechenzeit und Arbeitsspeicher interpretiert werden können, um dem Anwender Nachrichten an seinen aktuellen Zustand angepasst zu präsentieren.

Wir geben einen Überblick über den instrumentierten Raum an unserem Lehrstuhl (Saarland University Pervasive Instrumented Environment (SUPIE)) (siehe [Butz & Krüger 06]), indem wir ein Anwendungsszenario beschreiben, in dem der Anwender an einem Flughafen in der zollfreien Zone in einem Laden einkauft, während er auf seinen Abflug wartet. Wenn die Boarding Zeit naht, informiert ihn der Alarm Manager mit steigender Intensität, in Abhängigkeit der von ihm gemessenen Reaktion.

### 7.2.1 Systemüberblick

In diesem Abschnitt geben wir einen kurzen Überblick über das System, dem eine detailliertere Beschreibung der Komponenten im Hauptteil folgt.

Für die Erfassung der Daten über den Zustand des Benutzers verwenden wir einen Varioport (siehe Abschnitt C.1), ein mobiles Gerät, mit dem sich Signale von Umgebungs- und physiologischen Sensoren aufzeichnen lassen. In unserem Beispielszenario haben wir einen Elektromyogramm-Sensor (EMG) am Unterarm, einen Elektrookulogramm-Sensor (EOG) zwischen den Augenbrauen und einen Beschleunigungssensor (ACC) an einem Oberschenkel angelegt (siehe Abbildung 7.8).

Diese Daten werden vom Varioport über eine serielle Schnittstelle durch die Software *Javario* (siehe Abschnitt C.2) ausgelesen, die auf dem Laptop läuft. (Momentan wird an einer Portierung für den PDA Dell 600 gearbeitet.) Nach der Vorverarbeitung und Strukturierung in ein geeigneteres Format werden die Daten über Funk (WLAN) an einen Handheld –ein Zaurus SL-5500G, der mit einem Intel strongARM Prozessor ausgerüstet ist, der eine Taktfrequenz über 206 MHz hat– gesendet, wo sie als Evidenz in einem dynamischen Bayesschen Netz instantiiert werden. Das dynamische Bayessche Netz berechnet daraus eine Einschätzung über den Zustand des Benutzers und sendet die Ergebnisse kontinuierlich an den Event Heap.

Der Alarm Manager informiert den Benutzer über akustische und visuelle Meldungen mit steigender Intensität bis (in Abhängigkeit von der Einschätzung, die das dynamische Bayessche Netz berechnet) angenommen werden kann, dass der Benutzer auf die Nachricht reagiert hat (siehe Abbildung 7.6).

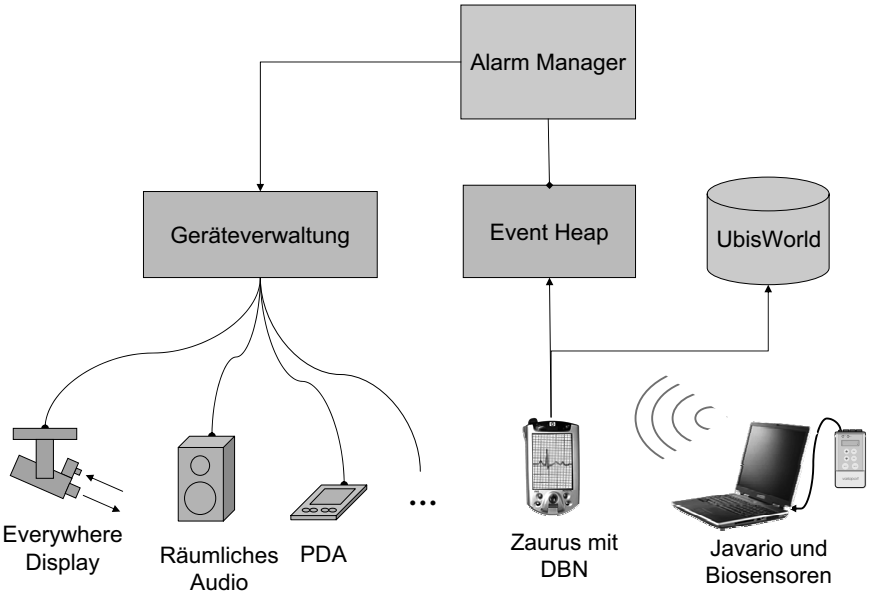



Abbildung 7.6: Infrastruktur des Szenarios

Der Alarm Manager ist ein Dienst, der verschiedene Arten von Benachrichtigungen in der instrumentierten Umgebung auslöst, wobei er den Benutzerzustand berücksichtigt, der durch das dynamische Bayessche Netz auf dem Zaurus eingeschätzt wird. Als erster Schritt des Verfahrens wird der Alarm Manager gestartet, wenn die Boarding Zeit nahe ist. Er wird nun eine Nachricht mit dem Boarding-Aufruf auf dem Handheld des Anwenders anzeigen und eine kurzes Audio-Signal abspielen, um Aufmerksamkeit zu erzielen. Nun werden die Arousal-Werte, die von JavaDBN an den Event Heap geschickt werden, so lange beobachtet wie die Benachrichtigung aktiv ist, um zu sehen, ob der Anwender reagiert. Wenn der Arousal-Wert unter einem gewissen Schwellwert bleibt, wird vom System angenommen, dass der Benutzer den Alarm nicht wahrgenommen hat und die nächste Benachrichtigungsstufe wird initialisiert: Der Alarm Manager spielt eine allgemeine Boarding-Ansage auf öffentlichen Lautsprechern ab, die sich in der Wahrnehmungsreichweite des Benutzers befinden – in unserem Fall ist dies das räumliche Audiosystem an unserem Lehrstuhl (siehe [Schmitz 03, Schmitz & Butz 06]). Wiederum wird der Arousal-Wert des Anwenders beobachtet und wenn notwendig, wird der dritte und letzte Benachrichtigungsschritt gestartet, indem ein virtueller Avatar (siehe Abbildung 7.7 und [Kruppa et al. 05], [Kruppa 05] und [Kruppa 06]) gestartet wird, der sich direkt an den Benutzer wendet und auffordert, sofort zum Gate zu gehen.

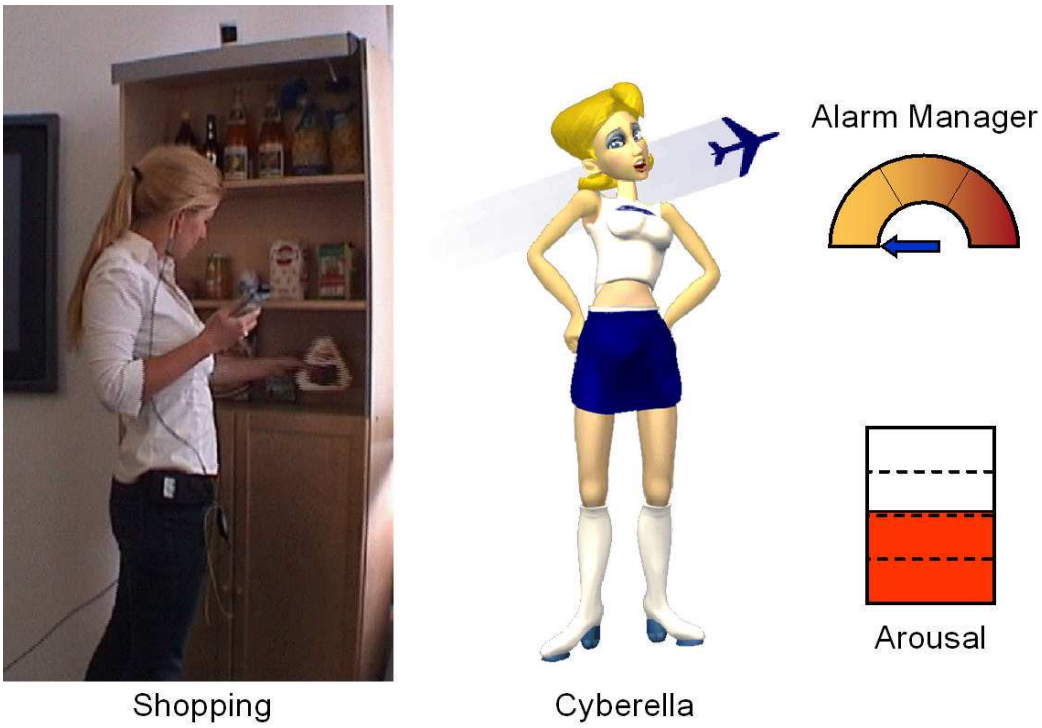



Abbildung 7.7: Durchsage eines Avatars

### 7.2.2 Infrastruktur

Der *Event Heap* (siehe [Johanson & Fox 02]) ist eine zentrale Kommunikationseinheit unserer instrumentierten Umgebung und wird von den Anwendungen für den Austausch und die Synchronisierung von Daten verwendet. Informationseinheiten werden als *Events* gespeichert. In unserem Beispiel sind dies die Inferenzen, die durch den Zaurus gemacht werden, der über WiFi mit dem lokalen Netzwerk verbunden ist, um auf den Event Heap zugreifen zu können. Diese Events verbleiben auf dem Event Heap für eine bestimmte Zeitdauer (zwei Minuten per Default) während dessen sie für Dienste und Anwendungen (in unserem Fall der Alarm Manager) verfügbar sind. Anwendungen melden im allgemeinen einen oder mehr Listener auf dem Event Heap an, die die interessierenden Events herausfiltern und an die Anwendung weiterreichen.

Die Ausgabegeräte der instrumentierten Umgebung werden durch einen zentralen Gerätemanager verwaltet, der es den Geräten erlaubt, sich mit ihren Ein- und Ausgabemöglichkeiten an bzw. abzumelden, wenn sie in den Raum gebraucht bzw. wieder aus dem Raum entfernt werden. Der Gerätemanager wurde im Projekt FLUIDUM (siehe [Fluidum ]) umgesetzt und wird vom Alarm Manager dazu verwendet, die Geräte mit den notwendigen Ausgabeeigenschaften anzufragen.

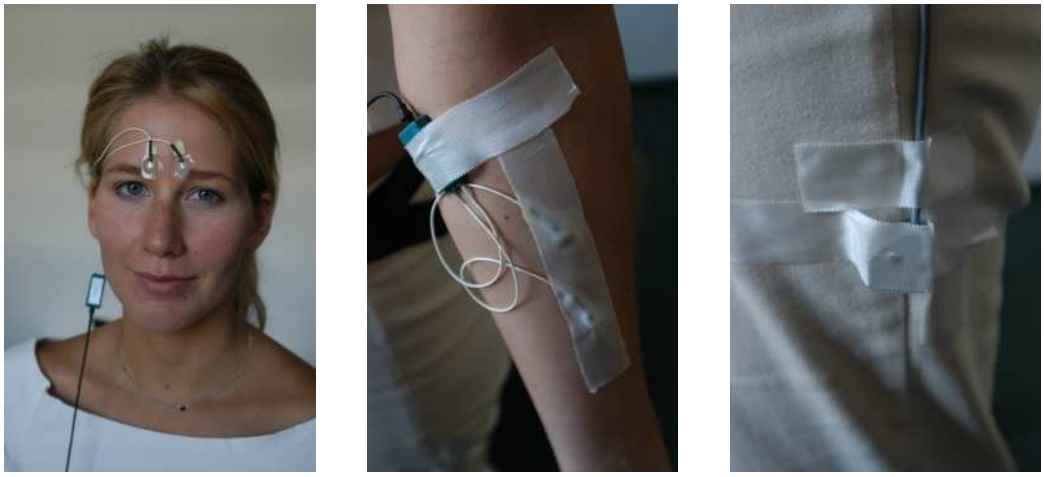


Abbildung 7.8: Elektrookulogramm-, Elektromyogramm- und Beschleunigungssensor (von links nach rechts)

Mit UbisWorld (siehe [Heckmann 06b]) können Benutzermodelle in ubiquitären Umgebungen verwaltet und Teile der realen Welt wie Personen, Objekte, Orte, Zeiten, Ereignisse und ihre Eigenschaften und Fähigkeiten repräsentiert werden. Über ein web-basiertes Interface können die zugrundeliegende Ontologie verändert, neue Äußerungen eingetragen und der Zugriff zu möglicherweise privaten Daten eingeschränkt werden (siehe [Stahl & Heckmann 04]). Unsere Anwendung verwendet sie, um die Sensordaten und die Inferenzen des dynamischen Bayesschen Netzes für eine mögliche spätere Verwendung oder Simulation zu speichern.

Die in Abbildung 7.1(a) visualisierte Verarbeitungshierarchie zwischen Sensordaten, ihrer Vorverarbeitung und Klassifizierung und die Instantiierung der Ergebnisse als Evidenz in einem dynamischen Bayesschen Netz stellt sich in dieser Anwendung konkret wie folgt dar: Die Sensordaten werden durch aktive Sensoren gemessen und auf dem Varioport bzw. durch die Javario-Anwendung vorverarbeitet. Für jeden Sensor werden Aktivierungsschwellwerte berechnet. Die Messungen des Beschleunigungssensors werden weitergehend interpretiert, um die horizontale Position des Sensors zu bestimmen. Schritte werden erkannt, indem gewisse Schwellwerte angewandt werden. So kann die Gehgeschwindigkeit anhand der Frequenz der festgestellten Schritte berechnet werden. (Die Javario-Bibliothek stellt für die Verwaltung und das Auslesen der Sensoren des Varioports mehrere Funktionen zur Verfügung (siehe Abschnitt C.2 für mehr Informationen).) Die Ergebnisse dienen dann als Eingabe als Evidenz in Knoten des dynamischen Bayesschen Netzes (siehe Abbildung 7.9), welches auf dem Zaurus läuft.

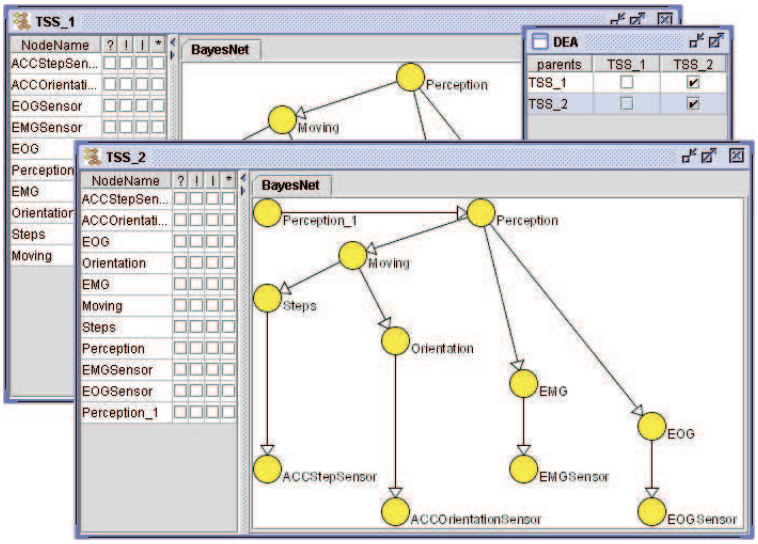



Abbildung 7.9: Bildschirmabzug von JavaDBN mit gerichtetem Graphen (DEA) und den Zeitscheibenschemata TSS_1 und TSS_2.

### 7.2.3 Zaurus mit dynamischem Bayesschen Netz

Über eine direkte WiFi-Verbindung zum Laptop erhält der Zaurus die Daten. Auf ihm läuft ein dynamisches Bayessches Netz, das die Einschätzung über den Benutzerzustand anhand der verfügbaren Messungen berechnet. Sobald neue Daten eintreffen, wird eine neue Zeitscheibe instantiiert und die Einschätzung berechnet gefolgt vom Rollup. Im folgenden wird nun das dynamische Bayessche Netz vorgestellt und gezeigt, wie es in das System eingebunden wird.

**Dynamisches Bayessches Netz** Das dynamische Bayessche Netz besteht aus zwei Zeitscheibenschemata (Zeitscheibenschema 1 (TSS_1) für die Instantiierung der initialen Zeitscheibe und Zeitscheibenschema 2 (TSS_2) für die Instantiierung der nachfolgenden Zeitscheiben), deren Graphen in Abbildung 7.9 dargestellt sind. In TSS_2 befinden sich die Knoten Perception_1, Perception, Movement, Steps, Orientation, EMG, EOG, ACCStepSensor, ACCOrientationSensor, EMGSensor und EOGSensor. TSS_1 unterscheidet sich von TSS_2 nur in dem Punkt, dass es keinen Knoten Perception_1 enthält, der den Einfluss der vorhergehenden Zeitscheibe auf die aktuelle Zeitscheibe modelliert. Die Knoten ACCStepSensor, ACCOrientationSensor, EMGSensor und EOGSensor entsprechen den Sensorvariablen des dynamischen Bayesschen Netzes in Abbildung 7.1(a). Diese Knoten werden mit den Ergebnissen der Vorverarbeitung wie Aktivierungsgrad, Orientierung und Schritt instantiiert. Die bedingten Wahrscheinlichkeiten der Knoten modellieren dabei die Zuverlässigkeit der Sensoren, die in unserem

Fall nur eingeschätzt wurde.

Der Knoten ACCStepSensor modelliert die Zuverlässigkeit des Schrittsensors, der die Anzahl der Schritte nach 'keine', 'wenige', 'mittel' und 'viele' einschätzt. Die Tabelle bedingter Wahrscheinlichkeiten des Knoten ACCStepSensor sieht wie folgt aus:

$$\mathrm{CPT}(\mathsf{ACCStepSensor}) = \left[\begin{array}{r|cccc} \mathsf{Steps} & \text{keine} & \text{wenige} & \text{mittel} & \text{viele} \\ \hline \text{keine} & 0.988 & 0.040 & 0.010 & 0.001 \\ \text{wenige} & 0.010 & 0.900 & 0.040 & 0.001 \\ \text{mittel} & 0.001 & 0.050 & 0.900 & 0.010 \\ \text{viele} & 0.001 & 0.010 & 0.050 & 0.988 \end{array}\right]$$

Der ACCOrientationSensor hat die Tabelle

$$\mathrm{CPT}(\mathsf{ACCOrientationSensor}) = \left[\begin{array}{r|cc} \mathsf{Orientation} & \text{waagrecht} & \text{senkrecht} \\ \hline \text{waagrecht} & 0.998 & 0.002 \\ \text{senkrecht} & 0.002 & 0.998 \end{array}\right]$$

als Tabelle bedingter Wahrscheinlichkeiten.

Für die beiden Knoten EMGSensor und EOGSensor haben wir jeweils die folgende Tabelle bedingter Wahrscheinlichkeiten:

$$\mathrm{CPT}(\{\mathsf{EMG/EOG}\}\mathsf{Sensor}) = \left[\begin{array}{r|ccc} \{\mathsf{EMG/EOG}\} & \text{gering} & \text{mittel} & \text{hoch} \\ \hline \text{gering} & 0.90 & 0.05 & 0.02 \\ \text{mittel} & 0.08 & 0.90 & 0.08 \\ \text{hoch} & 0.02 & 0.05 & 0.90 \end{array}\right].$$

Die Knoten Steps, Orientation, EMG und EOG schätzen die tatsächlichen Werte des jeweiligen Sensors ein. Die beiden Knoten Steps und Orientation werden vom Knoten Movement beeinflusst, der die Art der Fortbewegung beschreibt und die Zustände 'liegen', 'sitzen', 'stehen', 'trödeln', 'gehen', 'eilen' und 'laufen' besitzt.

Der Knoten Steps modelliert die Anzahl der Schreitte und hat die Zustände 'keine', 'wenige', 'mittel' und 'viele'. Seine Tabelle bedingter Wahrscheinlichkeiten ist dementsprechend groß und sieht wie folgt aus:

$$\mathrm{CPT}(\mathsf{Steps}) = \left[\begin{array}{r|ccccccc} \mathsf{Movement} & \text{liegen} & \text{sitzen} & \text{stehen} & \text{trödeln} & \text{gehen} & \text{eilen} & \text{laufen} \\ \hline \text{keine} & 0.99 & 0.95 & 0.90 & 0.02 & 0.01 & 0.01 & 0.03 \\ \text{wenige} & 0.01 & 0.04 & 0.06 & 0.80 & 0.20 & 0.09 & 0.07 \\ \text{mittel} & 0.00 & 0.01 & 0.03 & 0.16 & 0.70 & 0.60 & 0.10 \\ \text{viele} & 0.00 & 0.00 & 0.01 & 0.02 & 0.09 & 0.30 & 0.80 \end{array}\right].$$

Die Tabelle bedingter Wahrscheinlichkeiten des Knotens Orientation mit seinen Zuständen 'waagrecht' und 'senkrecht' sieht wie folgt aus:

$$\mathrm{CPT}(\mathsf{Orientation}) = \left[\begin{array}{r|ccccccc} \mathsf{Movement} & \text{liegen} & \text{sitzen} & \text{stehen} & \text{trödeln} & \text{gehen} & \text{eilen} & \text{laufen} \\ \hline \text{waagr.} & 0.99 & 0.85 & 0.05 & 0.10 & 0.15 & 0.20 & 0.25 \\ \text{senkr.} & 0.01 & 0.15 & 0.95 & 0.90 & 0.85 & 0.80 & 0.75 \end{array}\right].$$

Mit dieser Modellierung erreichen wir, dass im Knoten Movement nur in der horizontalen Ausrichtung (Knoten Orientation) des ACC-Sensors Schwellwertüberschreitungen (Knoten Steps) als Schritte erkannt werden.

Der Knoten Perception schätzt ein, wie stark das Signal vom Anwender wahrgenommen wurde. Er besitzt die Zustände 'nicht', 'mittel', 'normal' und 'stark'. Seine bedingten Wahrscheinlichkeiten und die seiner direkten Nachfolgerknoten Movement, EMG und EOG sind so modelliert, dass der Zustand 'stark' des Knotens Perception erst dann eingeschätzt wird, wenn über längere Zeit permanent beim Knoten Movement einer der beiden Zustände 'eilen' oder 'laufen' und bei den beiden Knoten EMG und EOG jeweils der Zustand 'hoch' eingeschätzt wird. Die Tabelle bedingter Wahrscheinlichkeiten des Knotens Movement sieht wie folgt aus:

$$\mathrm{CPT}(\mathsf{Movement}) = \left[\begin{array}{r|cccc} \mathsf{Perception} & \text{keine} & \text{mittel} & \text{normal} & \text{stark} \\ \hline \text{liegen} & 0.1754390 & 0.0951477 & 0.0223464 & 0.0000000 \\ \text{sitzen} & 0.1754390 & 0.1308280 & 0.0849162 & 0.0115929 \\ \text{stehen} & 0.1754390 & 0.1546150 & 0.1061450 & 0.0231858 \\ \text{trödeln} & 0.1754390 & 0.1784010 & 0.1273740 & 0.0579643 \\ \text{gehen} & 0.1754390 & 0.1784010 & 0.2234640 & 0.1119870 \\ \text{eilen} & 0.0614035 & 0.1531870 & 0.2234640 & 0.3408300 \\ \text{laufen} & 0.0614035 & 0.1094200 & 0.2122910 & 0.4544400 \end{array}\right].$$

Die Tabelle bedingter Wahrscheinlichkeiten der beiden Knoten EMG und EOG ist identisch und sieht wie folgt aus:

$$\mathrm{CPT}(\{\mathsf{EMG/EOG}\}) = \left[\begin{array}{r|cccc} \mathsf{Perception} & \text{keine} & \text{mittel} & \text{normal} & \text{stark} \\ \hline \text{gering} & 0.499 & 0.39 & 0.10 & 0.01 \\ \text{mittel} & 0.500 & 0.60 & 0.60 & 0.50 \\ \text{hoch} & 0.001 & 0.01 & 0.30 & 0.49 \end{array}\right].$$

Die Tabelle bedingter Wahrscheinlichkeiten des (dynamischen) Knotens Perception sieht wie folgt aus:

$$\mathrm{CPT}(\mathsf{Perception}) = \left[\begin{array}{r|cccc} \mathsf{Perception\_1} & \text{keine} & \text{mittel} & \text{normal} & \text{stark} \\ \hline \text{keine} & 0.40 & 0.30 & 0.29 & 0.001 \\ \text{mittel} & 0.30 & 0.40 & 0.30 & 0.009 \\ \text{normal} & 0.29 & 0.29 & 0.40 & 0.090 \\ \text{stark} & 0.01 & 0.01 & 0.01 & 0.900 \end{array}\right].$$

Sie ist so modelliert, dass sich die Einschätzung des Knotens nach einer gewissen Zeit, in der keine gestiegenen Werte an den Sensorknoten anliegen, wieder auf Normalmaß 'normal' einpendelt. Erst wenn der Zustand 'stark' über einen kurzen Zeitraum anliegt, wird wie geschildert angenommen, dass der Alarm wahrgenommen wurde.

**Systemeinbindung** JavaDBN bestimmt nun aus diesen beiden Zeitscheibenschemata ein Polynom, das dem dynamischen Bayesschen Netz entspricht und sowohl Inferenzen als auch das Aufrollen vorhergehender Zeitscheiben erlaubt. Für dieses Polynom erzeugt JavaDBN Quellcode (Java oder C++), der die Berechnungen für die Auswertung des Polynoms (die Inferenz im dynamischen Bayesschen Netz) und den Rollup (das Abschneiden der vorhergehenden Zeitscheibe) durchführt. Dieses Vorgehen ist in Kapitel 6 genauer beschrieben.

Der Quellcode des dynamischen Bayesschen Netzes lässt sich nun auf einfache Weise in vorhandene Anwendungen einbetten, wie es im folgenden gezeigt wird:

```
import JavaDBN.EmbeddedJavaDBN;

public class Example {
    [...]
    public static void main(String args[]) {
        [...]
        EmbeddedJavaDBN jDBN =
            new EmbeddedJavaDBN();
        [...]
        while([...]) {
            [...]
            jDBN.setEvidence_EMGSensor1_State2();
            [...]
            jDBN.initialize();
            SendData.sendBiodata(jDBN);
            jDBN.rollup();
            [...]
        }
        [...]
        return;
    }
}
```

Nach dem Rollup kann neue Evidenz eingetragen werden, so dass eine neue Einschätzung berechnet werden kann, die auch die vorhergehenden (aufgerollten) Zeitscheiben mit ihrer Evidenz ohne Approximation berücksichtigt.

## 7.3 Multisensorielles Positionierungssystem

Das System LORIOT ist ein Beispiel für die zeitgesteuerte Instantiierung von Zeitscheiben (siehe Abschnitt 5.2.1). In genau definierten Abständen wird eine neue Zeitscheibe angehängt, ein Rollup durchgeführt und neue Evidenz eingetragen. Wie wir sehen werden, werden bei diesem Vorgang bestimmte dynamische Bayessche Netze gelöscht und neue dynamische Bayessche Netz erzeugt.

Im folgenden Abschnitt stellen wir das System LORIOT[2] und das neu eingeführte Verfahren zur Positionsbestimmung mit den sogenannten georeferenzierten dynamischen Bayesschen Netzen vor (siehe [Brandherm & Schwartz 05]).

Für ein weites Anwendungsspektrum im Bereich des *Pervasive Computing* ist es wichtig die Position des Anwenders zu kennen. Wir präsentieren für die In- und Outdoor-Positionierung im folgenden eine Methode, die sowohl im Rechenzeit- als auch im Speicherplatzbedarf effizient und darüber hinaus hoch-skalierbar ist. Es wird dabei eine Technik angewendet, die das Problem von Unsicherheit, verrauschten Sensordaten und der Sensorfusion bewältigt. Die sogenannten georeferenzierten dynamischen Bayesschen Netze ermöglichen die Berechnung der Benutzerposition auf dem eigenen kleinen Hand-held Gerät (z. B. Pocket PC), ohne dass eine externe Verbindung zu einem Server besteht. Somit wird die Privatsphäre des Anwenders gewahrt, und es verbleibt in seinen Händen, an wen er die Positionsdaten übermitteln möchte.

### 7.3.1 Einleitung

Um die Position des Anwenders bestimmen zu können, müssen Sensordaten ausgewertet werden. Autonavigationssysteme oder Out-door Positionierungssysteme verwenden typischerweise GPS (Global Positioning System), um die Koordinaten des Anwenders zu berechnen. Allerdings arbeitet GPS aufgrund seiner physikalischen Signaleigenschaften in Indoor-Umgebungen nicht zuverlässig, so dass andere Technologien eingesetzt werden müssen. Beispiele wären Infrarot-Sender, Radiofrequenz-Emitter, Laser Range Scanner oder Videokameras. Diese Sender (und ihren entsprechenden Empfänger) unterscheiden sich beispielsweise bezüglich ihrer Sender-Eigenschaften, Sender-Reichweite, Genauigkeit, Preis, Baugröße und Stromverbrauch. Natürlich gibt es auch hier den üblichen Kompromiss zwischen den offensichtlich wichtigen technischen Faktoren (Genauigkeit, Stromverbrauch) und den einschränkenden Faktoren (Kosten, Baugröße, Sender-Eigenschaften), so dass die Entscheidung, welcher Sender verwendet werden soll, stark von der Umgebung und der geplanten Anwendung als auch von den finanziellen Gegebenheiten abhängt. Ebenso kann es der Fall sein, dass bereits installierte

[2] LORIOT ist ein Akronym für **L**ocalization and **Or**ientation in **I**n- and **O**ut-door Environmen**t**s.

Positionierungs-Sender und/oder -Sensoren durch zusätzliche Sender mit höherer Genauigkeit verbessert werden sollen. Ein Positionierungssystem sollte deswegen nicht nur verschiedene Arten von Sensordaten verarbeiten, sondern sie auch zusammenbringen können, um möglichst die genaueste Position bestimmen zu können.

Ein anderes Problem stellen die verrauschten Daten der Sensoren dar; weshalb eine probabilistische Herangehensweise wünschenswert wäre, um die Benutzerposition zu berechnen. Üblicherweise werden Bayessche Filter verwendet, um die wahre Position des Benutzers aufgrund der empfangenen Sensordaten zu bestimmen. Es sind verschiedene Verfahren bekannt, die jeweils solche Bayesschen Filter realisieren und sich in der Komplexität und gewissen Einschränkungen unterscheiden (siehe [Fox et al. 02] für eine detaillierte Beschreibung der verschiedenen Bayesschen Filtertechniken). Partikelfilter sind ein solches Verfahren, die oft in Lokalisierungssystemen eingesetzt werden. Obwohl sie die meisten Einschränkungen anderer Filter überwinden, müssen sie immer noch für ihre Umgebung trainiert bzw. kalibriert werden.

Unsere Idee der *georeferenzierten dynamischen Bayesschen Netze* erlaubt die Fusion von Sensordaten und bewältigt das Problem ungenauer Daten. Dieser neue Ansatz ist so sparsam im Speicherplatz- und Rechenzeitbedarf, dass er auf einem eingebetteten System (z. B. Hewlett-Packard iPAQ) implementiert und ausgeführt werden kann.

In Abschnitt 7.3.2 beschreiben wir die technischen Details und die Design-Entscheidungen unseres Positionierungssystems. In Abschnitt 7.3.3 erklären wir die Idee der georeferenzierten dynamischen Bayesschen Netze, beschreiben den Algorithmus im Detail und illustrieren ihn mit einem Beispiel. Die Realisierung des Systems an unserem Lehrstuhl als Beweis der Funktionstüchtigkeit stellen wir in Abschnitt 7.3.4 vor.

## 7.3.2 Technische Voraussetzungen/Fragestellungen

Im folgenden werden einige technische Details des Systems erklärt, die für das Verständnis des Konzeptes georeferenzierter dynamischer Bayesscher Netze hilfreich sein könnten.

### 7.3.2.1 Tracking

In Systemen wie Active Badge, Active Bat oder Ubisense (siehe [Want et al. 92, Harter et al. 02, Ubisense 05]) trägt der Benutzer eine Art von Sender (Infrarot, Ultrasound, Ultraweitband Radio) und die entsprechenden Sensoren sind in der Umgebung installiert. Wir nennen diese Systeme *Trackingsysteme*, weil der Sender aktiv die Benutzerposition seiner Umgebung offenbart. Ein Benutzer eines

solchen Systems weiß nicht, was mit den gesammelten Daten passiert und wer Zugriff auf sie hat. Sicherlich können Funktionen implementiert werden, die es dem Benutzer ermöglichen festzulegen, wer Zugriff auf seine Daten haben darf. Dies bedeutet aber, dass der Benutzer dem System vertraut. Wenn er dem System nicht vertraut, dann ist seine einzige Alternative, dass er den Sender nicht trägt (oder ihn ausschaltet). So verhindert er allerdings nicht nur, dass andere seine aktuelle Position erfahren, sondern auch, dass er das System für seine eigenen Bedürfnisse nicht benutzen kann (z. B. für Navigation).

#### 7.3.2.2 Positionierung

Bei einem Positionierungssystem wird die Position des Benutzers auf dem persönlichen mobilen Gerät des Benutzers berechnet. Dies kann man bewerkstelligen, indem die *Sender* in der Umgebung installiert werden und das mobile Gerät des Benutzers mit den entsprechenden Sensoren ausgestattet wird. Somit liegt die Berechnung der Position und die Benutzerposition buchstäblich in der Hand des Benutzers, und er kann entscheiden, ob er diese Information über eine WiFi-Verbindung mit anderen Leuten oder Anwendungen teilen möchte. Wenn er der Datenschutzkontrolle des Systems misstraut, dann kann er seine WiFi-Verbindung ausschalten, und er wird trotzdem in der Lage sein, seine eigene Position für die Navigation zu bestimmen.

Wir bevorzugen diese Herangehensweise und haben deswegen unser System in dieser Art entworfen. (Als Sender sind Infrarot-Baken und aktive RFID-Tags in der Umgebung installiert und werden *nicht* vom Benutzer getragen.)

#### 7.3.2.3 Infrarot-Baken

Wir verwenden Infrarot-Baken der Eyeled GmbH (siehe [Eyeled 06]). Diese Baken haben Batterien als Stromversorgung und senden über Infrarotlicht einen 16 Bit breiten Identifizierungscode aus, der für jede Bake individuell eingestellt werden kann. Der Infrarot-Strahl ist aufgrund der physikalischen Eigenschaften von Licht konisch. Der benötigte Infrarot-Sensor ist oftmals schon im PDA eingebaut, um Daten austauschen zu können.

#### 7.3.2.4 Aktive RFID-Tags

RFID-Tags (Radio Frequency IDentification) sind als passive oder als aktive Bauteile erhältlich. In beiden Arten speichern die Tags einen Identifizierungscode, der durch ein spezielles RFID-Lesegerät ausgelesen werden kann. Die passiven Tags erhalten ihre Energie durch das Radiosignal des Lesegerätes. Aktive RFID-Tags

haben ihre eigene Stromversorgung (wie beispielsweise durch Batterien). Wir verwenden RFID-Tags der Identec Solutions AG (siehe [Identec 05]), die im Freien eine Reichweite von bis zu 100 m aufweisen. Aufgrund der physikalischen Eigenschaften von Radiowellen wird radial ausgestrahlt. Die Lesegeräte für aktive RFID-Tags haben verschiedene Baugrößen. In Verbindung mit dem PDA verwenden wir eine PCMCIA-Lesekarte.

In [Ni et al. 04] wird ein *Trackingsystem* (siehe Abschnitt 7.3.2.1) beschrieben, das aktive RFID-Tags verwendet. Es werden Lesegeräte in der Umgebung installiert und der Anwender mit einem Tag ausgestattet. Dabei ist der Empfang der aktiven RFID-Tags stark von unterschiedlichen Faktoren abhängig wie statische Hindernisse oder die Benutzer-Bewegung. Um die Genauigkeit des Systems zu erhöhen, haben sie Referenz Tags in der Umgebung installiert.

Ein *Positionierungssystem* wird in [Amemiya et al. 04] beschrieben. Ihre Herangehensweise ist unserer ähnlich, da die Tags in der Umgebung installiert werden (auf dem Boden, wir haben unsere Tags an der Decke befestigt.), und die Anwender einen oder mehrere Leser tragen lassen. Sie berechnen die Benutzerposition zum Zeitpunkt $t$, indem die Summe der bekannten Positionen der empfangenen RFID-Tags und die vorhergehende Position berechnet und durch die Anzahl der empfangenen Sensoren plus 1 (für die vorhergehende Position) dividiert wird:

$$\text{UserPos}(t) = \frac{1}{n+1}\left(\text{UserPos}(t-1) + \sum_{i=1}^{n} \text{Coord}\left(\text{receivedTag}[i]\right)\right) \tag{7.1}$$

(Die Gleichung wurde vereinfacht und unserer Schreibweise angepasst. Für die originale Version verweisen wir auf [Amemiya et al. 04].) Unsere Erfahrungen mit aktiven RFID-Tags zeigen, dass das Lesegerät oftmals auch Tags empfängt, die sich sehr weit entfernt vom Benutzer befinden. Deswegen versuchen wir mittels dynamischen Bayesschen Netzen diese Messfehler herauszufiltern.

### 7.3.3 Georeferenzierte dynamische Bayessche Netze

Im folgenden beschreiben wir die Idee der georeferenzierten Dynamischen Bayesschen Netze (geoDBN) und wie man sie zur Sensorfusion und zum Herausfiltern von Messfehlern verwenden kann.

#### 7.3.3.1 Idee

Unsere Idee ist es nun, die Eigenschaften der verwendeten Sender (in unserem Fall IR-Baken und RFID-Tags) in einem dynamischen Bayesschen Netz zu modellieren. Man beachte, dass wir mit dem dynamischen Bayesschen Netz nicht alle Sender gleichzeitig modellieren, die in der Umgebung installiert sind. Wir

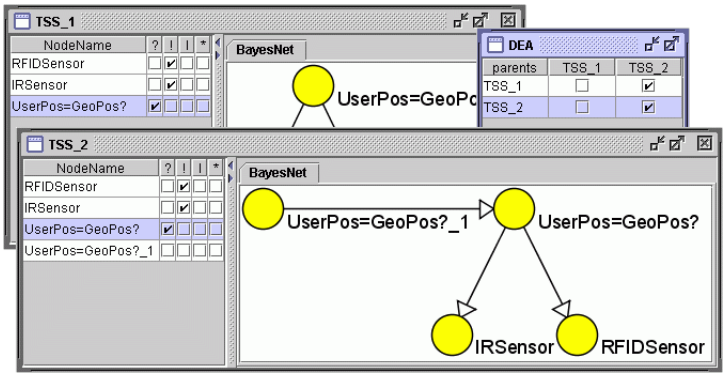



Abbildung 7.10: Bildschirmabzug von JavaDBN mit gerichtetem Graphen (DEA) und den Zeitscheibenschemata TSS_1 und TSS_2.

verwenden ein kleines dynamisches Bayessches Netz, das prototypisch die Verlässlichkeit der Sendertypen beschreibt. (Wir nehmen dabei an, dass alle Sender des gleichen Typs dieselben Eigenschaften besitzen.) Dieses prototypische dynamische Bayessche Netz wird mehrmals während der Laufzeit instantiiert und jede Instanz dieses dynamischen Bayesschen Netzes wird mit einer Geokoordinate *GeoPos* assoziiert.

In Abbildung 7.10 ist das dynamische Bayessche Netz zu sehen, das wir in unserer Beispielanwendung benutzen. Der oberste rechte Knoten (beschriftet mit UserPos=GeoPos?) repräsentiert die Wahrscheinlichkeit, dass der Anwender an der assoziierten Geokoordinate *GeoPos* steht. Der Knoten UserPos=GeoPos?_1, der sich links davon befindet, gibt die Wahrscheinlichkeit wieder, die in der vorhergehenden Zeitscheibe berechnet wurde. Die beiden unteren Knoten IRSensor und RFIDSensor repräsentieren die Wahrscheinlichkeit, dass eine IR-Bake und/-oder ein RFID-Tag an der Geokoordinate *GeoPos* empfangen werden kann, unter der Voraussetzung, dass der Anwender an der Geokoordinate *GeoPos* steht.

Wenn ein Infrarot-Signal empfangen wird, ist dies eine hohe Evidenz dafür, dass sich der Anwender in der Nähe der entsprechenden Bake aufhält (die Infrarot-Sensordaten sind sehr rauscharm). Wird hingegen ein RFID-Tag empfangen, so ist dies eine wesentlich geringere Evidenz dafür, dass sich der Anwender in der Nähe des Senders aufhält (der Sensor ist sehr verrauscht). Diese Eigenschaften sind in den bedingten Wahrscheinlichkeitentabellen (CPTs) der IR-Sensor- und RFID-Sensor-Knoten modelliert. Das von uns in unserer Beispielanwendung aktuell verwendete dynamische Bayessche Netz wird in Abschnitt 7.3.3.3 detaillierter beschrieben.

Diese Netze mit ihren assoziierten Koordinaten sind die sogenannten georeferenzierten dynamischen Bayesschen Netze (geoDBNs). Jedes geoDBN repräsentiert dabei die Einschätzung, dass sich der Anwender an der assoziierten Koordi-

nate befindet.

Die aktiven RFID-Tags haben einen kleinen internen Speicher, den man frei zum Lesen und Beschreiben von Daten verwenden kann. Wir verwenden diesen Speicher, um die Koordinaten des Tags darin zu speichern. IR-Baken sind immer mit einem RFID-Tag kombiniert, das einmal seine eigene Koordinaten und die der nahe gelegenen Bake zur Verfügung stellt. Wird nun eine Bake oder ein Tag empfangen, werden geoDBNs instantiiert und mit der induzierten Koordinate assoziiert. Diese induzierten Koordinaten sind dabei von den gespeicherten Koordinaten und der Sendertypen abhängig.

Die Berechnung der Benutzerposition ist ziemlich ähnlich zu der in 7.1 vorgestellten Berechnung; wir gewichten hingegen die Koordinaten mit den berechneten Wahrscheinlichkeiten der existierenden geoDBNs:

$$\text{UserPos}(t) = \sum_{i=1}^{n} \alpha \; w(\text{GeoDBN}[i]) \; \text{GeoPos}(\text{GeoDBN}[i]). \tag{7.2}$$

In der obigen Formel ist $n$ die Anzahl der existierenden geoDBNs zum Zeitpunkt $t$ ($n \geq \#ReceivedSenders_t$), GeoPos(GeoDBN[$i$]) ist die Koordinate und $w$(GeoDBN[$i$]) das Gewicht des $i$-ten geoDBN. $\alpha$ ist ein Normalisierungsfaktor, so dass sich alle Gewichte multipliziert mit $\alpha$ zu 1 summieren.

Um die Berechnungskosten und den Speicherverbrauch gering zu halten, muss die Anzahl der instantiierten geoDBNs so gering wie möglich gehalten werden. Um dieses Ziel zu erreichen, werden geoDBNs mit einem Gewicht, das unter $threshold_{use}$ liegt, als ungenutzt markiert (diese geoDBNs bieten nur eine sehr geringe Evidenz, dass sich der Anwender in ihrer Umgebung aufhält). Dieser Schwellwert sollte der A-priori-Wahrscheinlichkeit des geoDBNs bei seiner ersten Instantiierung entsprechen. Um im Vorfeld Ressourcenbeschränkungen gerecht zu werden, kann eine maximale Anzahl an geoDBNs angegeben werden. Wird diese Anzahl überschritten, so werden geoDBNs gelöscht, die die geringste Einschätzung liefern. Um die Unkosten für die Speicherverwaltung gering zu halten (oder um eine automatische Speicherbereinigung zu vermeiden, wenn das System in Sprachen wie Java oder C# implementiert wurde), können als ungenutzt markierte geoDBNs wiederverwendet werden, indem ihre Werte zu Anfangswerten zurückgesetzt werden und sie neue Koordinaten erhalten.

Der folgende Abschnitt beschreibt den Algorithmus und erklärt ihn an einem Beispiel.

#### 7.3.3.2 Einschätzung der Benutzerposition

Eine neue Einschätzung der aktuellen Position des Anwenders wird nach jeder neuen Messung berechnet und basiert dabei sowohl auf den aktuellen Sensordaten

als auch auf den vorhergehenden Daten. Eine gewichtete Kombination der alten und neuen Daten wird durch die Inferenz im entsprechenden geoDBN erzielt. Die Herangehensweise sieht schematisch wie folgt aus:

1. Führe eine neue Messung durch und speichere die empfangenen Koordinaten

2. Hänge jedem existierenden geoDBN eine neue Zeitscheibe an und führe einen Rollup durch.

3. Trage die neue Evidenz der Sensoren ein:

   (a) Gibt es noch kein geoDBN an der empfangenen Koordinate, dann erzeuge ein neues geoDBN und trage die Evidenz ein.

   (b) Gibt es schon ein geoDBN an der empfangenen Koordinate, dann trage die Evidenz in der aktuellen Zeitscheibe ein.

4. Gehe alle geoDBNs durch und berechne die Einschätzung, dass sich der Anwender an der assoziierten Koordinate befindet.

5. Sortiere die geoDBNs in abfallender Reihenfolge bezüglich ihres BEL-Wertes.

6. Markiere geoDBNs als ungenutzt, wenn ihr BEL-Wert unter dem Schwellwert $threshold_{use}$ liegt.

7. Berechne die Benutzerposition, indem nur geoDBNs berücksichtigt werden, deren Einschätzung über dem Schwellwert $threshold_{consider}$ liegen.

Ein Beispieldurchlauf verbildlicht die Herangehensweise und den Algorithmus (die hier verwendeten Werte dienen nur zur Veranschaulichung und sind keine echten Daten):

Die nachfolgende Tabelle soll die aktuelle Situation beschreiben. Das Array GeoDBN[.] enthält die existierenden geoDBNs $\text{geoDBN}_a$ bis $\text{geoDBN}_d$ mit ihren entsprechenden Koordinaten nach ihren Einschätzungen (BEL(.)) absteigend sortiert.

| $i$ | GeoDBN[$i$] | GeoPos(.) | BEL(.) | $w$(.) |
|---|---|---|---|---|
| 1 | $\text{geoDBN}_a$ | $(10,5,0)$ | $(0.70,0.30)$ | 0.70 |
| 2 | $\text{geoDBN}_b$ | $(12,5,0)$ | $(0.70,0.30)$ | 0.70 |
| $n=3$ | $\text{geoDBN}_c$ | $(8,5,0)$ | $(0.50,0.50)$ | 0.50 |
| 4 | $\text{geoDBN}_d$ | $(14,5,0)$ | $(0.10,0.90)$ | 0.10 |

Mit Gleichung 7.2 berechnet sich die Benutzerposition zu $(10.21, 5.00, 0.00)$. Es ist dabei zu beachten, dass nur die ersten drei geoDBNs zur Berechnung beitragen, da der BEL-Wert des vierten geoDBN unter dem Schwellwert $threshold_{consider}$ liegt (Schritt 7 im Algorithmus).

Bei der nächsten Messung (Schritt 1) werden die Sender $\text{RFID}_b$, $\text{RFID}_d$, $\text{IR}_d$, $\text{RFID}_e$ und $\text{RFID}_f$ empfangen. Mit einem Blick in der vorhergehenden Tabelle erkennt man, dass für die ersten drei Sender schon geoDBNs existieren, wohingegen die letzten beiden Sender neu sind. Die neuen Daten werden in die geoDBNs eingetragen (für die ersten drei Sender, Schritt 3b) oder neue geoDBNs werden entsprechend erzeugt (die letzten drei Sender, Schritt 3a). Nach dem Update aller geoDBNs (Schritt 4) und der anschließenden Sortierung (Schritt 5) erhalten wir die folgende Situation:

| $i$ | GeoDBN[$i$] | GeoPos(.) | BEL(.) | $w$(.) |
|---|---|---|---|---|
| 1 | $\text{geoDBN}_b$ | $(12,5,0)$ | $(0.85, 0.15)$ | 0.85 |
| 2 | $\text{geoDBN}_d$ | $(14,5,0)$ | $(0.75, 0.25)$ | 0.75 |
| 3 | $\text{geoDBN}_a$ | $(10,5,0)$ | $(0.50, 0.50)$ | 0.50 |
| $n=4$ | $\text{geoDBN}_c$ | $(8,5,0)$ | $(0.25, 0.75)$ | 0.25 |
| 5 | $\text{geoDBN}_e$ | $(20,5,0)$ | $(0.06, 0.94)$ | 0.06 |
| 6 | $\text{geoDBN}_f$ | $(16,5,0)$ | $(0.06, 0.94)$ | 0.06 |

Das fünfte und sechste geoDBN sind unter dem Schwellwert $threshold_{consider}$, so dass im Schritt 7 nur die ersten vier geoDBNs zur Berechnung herangezogen werden. Mit Gleichung (7.2) kommt man zum Ergebnis $(11.78, 5.00, 0.00)$. Eine weitere Messung empfängt die Sender $\text{RFID}_b$, $\text{RFID}_d$ und $\text{RFID}_f$. Das Auffrischen und Sortieren der Daten liefert das folgende:

| $i$ | GeoDBN[$i$] | GeoPos(.) | BEL(.) | $w$(.) |
|---|---|---|---|---|
| 1 | $\text{geoDBN}_d$ | $(14,5,0)$ | $(0.89, 0.11)$ | 0.89 |
| 2 | $\text{geoDBN}_b$ | $(12,5,0)$ | $(0.85, 0.15)$ | 0.85 |
| 3 | $\text{geoDBN}_a$ | $(10,5,0)$ | $(0.25, 0.75)$ | 0.25 |
| $n=4$ | $\text{geoDBN}_f$ | $(16,5,0)$ | $(0.25, 0.75)$ | 0.25 |
| 5 | $\text{geoDBN}_c$ | $(8,5,0)$ | $(0.05, 0.95)$ | 0.05 |
| 6 | $\text{geoDBN}_e$ | $(20,5,0)$ | $(0.04, 0.96)$ | 0.04 |

Die Einschätzungen von $\text{geoDBN}_c$ und $\text{geoDBN}_e$ sind nun unter dem Schwellwert $threshold_{use}$, so dass beide als ungenutzt markiert werden (Schritt 6). Nur $\text{geoDBN}_d$, $\text{geoDBN}_b$, $\text{geoDBN}_a$, und $\text{geoDBN}_f$ werden in der Berechnung berücksichtigt, so dass sich die Benutzerposition zu $(13.02, 5.00, 0.00)$ bestimmt.

Der folgende Abschnitt ist eine detaillierte Darstellung des geoDBNs, das wir in unserer aktuellen Realisierung des Systems verwenden.

#### 7.3.3.3 Das geoDBN der Beispielanwendung

Um die Zeitscheibenschemata des dynamischen Bayesschen Netzes zu modellieren, verwenden wir die Java-Anwendung *JavaDBN* (siehe Abbildung 7.10). Ein gerichteter Graph (DEA) bestimmt, in welcher Reihenfolge die Zeitscheibenschemata instantiiert werden können, weiterhin kann der Anwender die Anfrage- und Evidenz-Knoten jedes Zeitscheibenschemas angeben (siehe die Tabelle in jedem Zeitscheibenschema-Fenster in Abbildung 7.10). Dann erzeugt JavaDBN Quellcode (bis jetzt Java und C++), der das dynamische Bayessche Netz repräsentiert und der die Routinen für die Inferenz enthält. Dieser Code erlaubt die Auswertung eines dynamischen Bayesschen Netzes mit konstantem Speicherplatzverbrauch und enthält ein nicht-approximatives Rollup-Verfahren, welches ältere Zeitscheiben abschneidet und dabei ihren Einfluss auf die verbleibenden Zeitscheiben des dynamischen Bayesschen Netzes ohne Informationsverlust berücksichtigt (siehe [Darwiche 03, Brandherm & Jameson 04] und Abschnitt 3.3 für den theoretischen Hintergrund).

Der Quellcode hat die weiteren zusätzlichen Fähigkeiten: Alle für die Berechnung benötigten Variablen werden während der Initialisierung des dynamischen Bayesschen Netzes erzeugt. Zur Laufzeit werden keine weiteren Variablen erzeugt oder weggeworfen, um die Unkosten für die Speicherverwaltung (C++) zu minimieren oder sogar die automatische Speicherbereinigung (Java) zu verhindern. Die Inferenz im dynamischen Bayesschen Netz wurde bezüglich der ausgewählten Anfrage- und Evidenz-Knoten optimiert. Es werden spezielle Funktionen zur Verfügung gestellt, um Evidenz einzutragen. Da aber auf die Variablen frei von außen zugegriffen werden kann, kann Evidenz in den Knoten auch direkt eingetragen werden.

Das georeferenzierte dynamische Bayessche Netz besteht aus einem Zeitscheibenschema zur Instantiierung der initialen Zeitscheibe (TSS_1) und einem Zeitscheibenschema zur Instantiierung der nachfolgenden Zeitscheiben (TSS_2). Die beiden Zeitscheibenschemata TSS_1 und TSS_2 sind in Abbildung 7.10 graphisch dargestellt. TSS_1 unterscheidet sich von TSS_2 dadurch, das es keinen Knoten UserPos=GeoPos?_1enthält, der den Einfluss der vorhergehenden Zeitscheibe auf die aktuelle modelliert. Der Knoten UserPos=GeoPos? in TSS_2 repräsentiert das Bewegungsmodell und enthält die beiden Zustände $a_1 =$ “ja” und $a_2 =$ “nein”. Der BEL-Wert dieses Knotens repräsentiert die Einschätzung des Netzes, ob sich der Anwender an der assoziierten Geokoordinate befindet. Der Zustand “ja” steht für “*nehme an, dass sich der Anwender hier aufhält*” und der Zustand “nein” steht für “*nehme an, dass sich der Anwender hier nicht aufhält*”. Die bedingten Wahrscheinlichkeiten des Knotens sind an die durchschnittliche Gehgeschwindigkeit eines Fußgängers angepasst, so dass die Einschätzung des Netzes entsprechend schnell abfällt, wenn der entsprechende Sender für ein paar aufeinanderfolgende

Messungen nicht empfangen wird (z. B. wenn sich der Benutzer nicht im Sendebereich des Senders aufhält). Die Einschätzung nimmt entsprechend zu, wenn der Anwender in den Sendebereich des Senders eindringt (oder wieder eindringt). Konkret sieht die Tabelle bedingter Wahrscheinlichkeiten des Knotens wie folgt aus:

$$\mathrm{CPT}(\mathsf{UserPos{=}GeoPos?}) = \left[\begin{array}{r|cc} \mathsf{UserPos{=}GeoPos?\_1} & a_1 & a_2 \\ \hline a_1 & 0.7 & 0.001 \\ a_2 & 0.3 & 0.999 \end{array}\right].$$

Der Knoten UserPos=GeoPos? in TSS_1 enthält ebenfalls die beiden Zustände $a_1 =$ “ja” und $a_2 =$ “nein”, aber es fehlt der vorhergehende Knoten, so dass sich die CPT zu den A-priori-Wahrscheinlichkeiten

$$\mathrm{CPT}(\mathsf{UserPos{=}GeoPos?}) = \left[\begin{array}{c|c} a_1 & 0.05 \\ a_2 & 0.95 \end{array}\right]$$

reduziert.

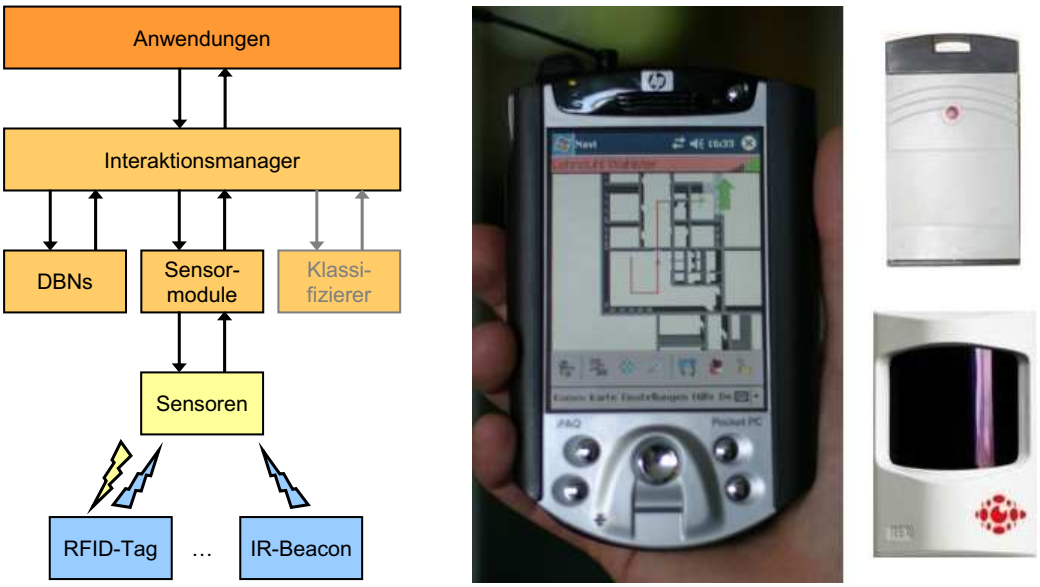
Die Knoten IRSensor und RFIDSensor entsprechen den echten Sensoren in der realen Welt. Diese Knoten werden mit den Messresultaten instantiiert. Die bedingten Wahrscheinlichkeiten der Knoten modellieren dabei die Verlässlichkeit der Sensoren (Wahrnehmungsmodell) entsprechend der realen Weltsituation. Der Knoten IRSensor hat die Zustände $b_1 =$ “ja”, und $b_2 =$ “nein”. Die nachfolgende CPT ist mit diesem Knoten assoziiert:

$$\mathrm{CPT}(\mathsf{IRSensor}) = \left[\begin{array}{r|cc} \mathsf{UserPos{=}GeoPos?} & a_1 & a_2 \\ \hline b_1 & 0.9 & 0.05 \\ b_2 & 0.1 & 0.95 \end{array}\right].$$

Die CPT lässt sich wie folgt interpretieren: Angenommen, der Anwender befindet sich im Sichtbereich der IR-Bake (bis zu 2 m Abstand), dann wird der Sensor des PDAs den Sender in 90% aller Fälle entdecken und in 10% aller Fälle nicht entdecken. Umgekehrt, befindet sich der Anwender weiter vom Sender entfernt, dann wird der Sensor in 5% aller Fälle doch noch das IR-Signal empfangen aber in 95% aller Fälle doch nicht empfangen. Es sei darauf hingewiesen, dass die IR-Bake alle 500 ms ein Signal sendet, wohingegen das RFID-Tag erst nach einem Ping-Signal vom PDA des Anwenders ein Signal sendet. Der Knoten RFIDSensor wie auch der Knoten IRSensor enthält die Zustände $c_1 =$ “ja” und $c_2 =$ “nein”. Die geringere Präzision des RFID-Tags im Vergleich zur IR-Bake spiegelt sich in der assoziierten CPT wider:

$$\mathrm{CPT}(\mathsf{RFIDSensor}) = \left[\begin{array}{r|cc} \mathsf{UserPos{=}GeoPos?} & a_1 & a_2 \\ \hline c_1 & 0.6 & 0.3 \\ c_2 & 0.4 & 0.7 \end{array}\right].$$

Die Ungenauigkeit des RFID-Signals beruht u. a. auf der hohen Sendereichweite. Die Genauigkeit kann erhöht werden, indem die Sendestärke des RFID-Lesegerätes verringert wird, der das oben erwähnte Ping-Signal aussendet.




(a) Systemarchitektur

(b) iPAQ mit PCMCIA RFID-Lesegerät, eingebautem WLAN und IR-Sensor (links) und RFID-Tag und IR-Bake (rechts)

Abbildung 7.11: Systemübersicht

Wenn neue Daten eintreffen, wird eine neue Zeitscheibe angehängt, die Einschätzung berechnet und danach ein Rollup durchgeführt.

### 7.3.4 Beispielanwendung

Um unsere Herangehensweise zu testen, haben wir ein Lokalisierungssystem implementiert und Teile unseres Lehrstuhl mit IR-Baken und RFID-Tags ausgestattet. Die Software läuft auf einem HP iPAQ h5550 mit 128 MB RAM und dem Betriebssystem Windows CE 4.2. Der iPAQ steckt in einem Erweiterungspack, das eine zusätzliche Batterie und einen PCMCIA-Slot, in dem das aktive RFID-Tag-Lesegerät steckt, zur Verfügung stellt. Die IR-Signale werden durch den internen IR-Port des iPAQ empfangen. Das System selbst besteht aus zwei Anwendungen, die simultan auf dem iPAQ laufen: Die erste Anwendung heißt PositionSensor, die alle 500 ms eine Messung beider Sensoren vornimmt und dann die gespeicherten Koordinaten jedes empfangenen RFID-Tags ausliest. Die gesammelten Daten werden wie vor beschrieben weiterverarbeitet und die eingeschätzte Koordinate des Anwenders wird an die zweite Anwendung geschickt, nämlich den BMW Personal Navigator (siehe [Krüger et al. 04]), der die Position des Anwenders visualisiert.

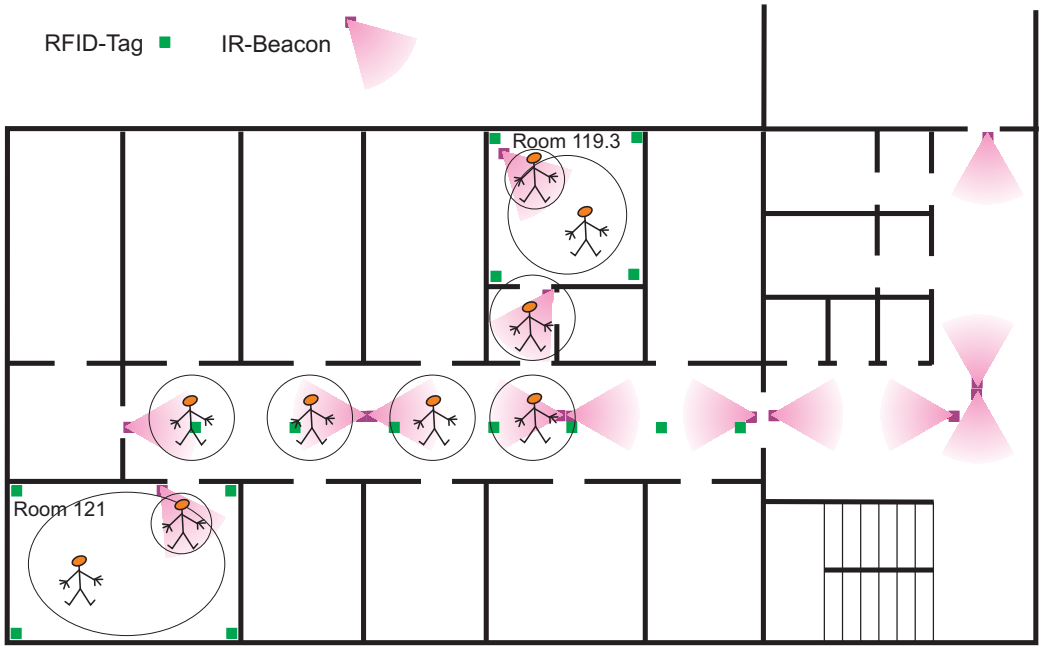



Abbildung 7.12: Ein Testgang durch den Lehrstuhl Wahlster. Das Strichmännchen zeigt die eingeschätzte Position und der Kreis zeigt die Fläche der realen Positionen an.

Abbildung 7.11 zeigt, wie der PositionSensor in das System integriert ist, so dass weitere Anwendungen nur über den Interaction Manager kommunizieren. Dieser Interaction Manager ist dafür zuständig, dass die rohen Sensordaten zu ihren entsprechenden Klassifizierern oder dynamischen Bayesschen Netzen weitergeleitet werden und dass die Rollups der dynamischen Bayesschen Netze und die Verarbeitung und das Lesen der Sensoren synchronisiert werden. Ebenso werden neue Klassifizierer und Sensoren registriert und ihr Zugriff verwaltet. Auf diese Art und Weise wird die Entwicklung von Anwendungen getrennt von der Erweiterung um Sensoren, Klassifizierern und dynamischen Bayesschen Netzen, welche weiterhin unabhängig von allen anderen Modulen und dem System selbst genutzt werden können.

In Abbildung 7.12 ist ein Testgang eines Anwenders von Raum 121 (untere linke Ecke in Abbildung 7.12) zu Raum 119.3 (obere rechte Ecke in Abbildung 7.12) unseres Lehrstuhls dargestellt. Das Strichmännlein zeigt die Position des Anwenders an, wie sie vom System eingeschätzt wird. Der dazugehörige Kreis um jedes Strichmännlein gibt die Fläche der möglichen Positionen wieder. RFID-Tags sind nur als Quadrate und IR-Baken als Quadrate mit Kegeln, die die Senderichtung anzeigen, eingezeichnet. Beide Räume sind mit RFID-Tags in jeder der vier Ecken und zusätzlichen IR-Baken am Eingang ausgestattet. In

Raum 119.3 steht ein Bücherregal, das mit einer IR-Bake ausgestattet ist. Der Flur selbst wurde alle zwei Meter mit einem RFID-Tag und alle vier Meter mit zwei IR-Baken (eine in jede Richtung bzgl. des Flures) versehen. Der Anwender hat sich mit langsamer bis mittlerer Gehgeschwindigkeit fortbewegt. Auf Raumebene wird die Benutzerposition sehr genau eingeschätzt; wenn sich z. B. der Anwender in einem bestimmten Raum befindet, wird das System die Position irgendwo in diesem Raum bestimmen. Die Position im Raum ist somit ziemlich grobkörnig, aber die Genauigkeit kann enorm erhöht werden, indem man IR-Baken an interessanten Stellen platziert (wie z. B. das Bücherregal in Raum 119.3). Die Positionseinschätzung im Flur variiert in etwa um einen Meter um die tatsächliche Position.

### 7.3.5 Zusammenfassung und Ausblick

Wir haben eine neue Herangehensweise zur Einschätzung der Benutzerposition durch eine probabilistische Sensorfusion vorgestellt. Unsere Methode ist in der Speicher- und Rechenkomplexität so gering, dass sie vollständig auf einem PDA lauffähig ist. Die Berechnung der Benutzerposition benötigt kein Modell der Umgebung und ebenso ist eine zeitaufwändige Kalibrierung überflüssig. Eine beliebige Umgebung kann instrumentiert werden, indem die Tags und Baken einfach aufgehängt und die entsprechenden Koordinaten in den Speicher des Gerätes geschrieben werden. Es können Regionen mit gröberer oder feinerer Granularität aufgebaut werden, indem Sender mit der entsprechenden Genauigkeit verwendet werden (z. B. IR-Baken für eine höhere und RFID-Tags für eine geringere Präzision).

In der Zukunft werden andere Sensortypen in das System integriert werden (z. B. Videokameras, Mikrophon-Arrays). Zudem muss ein System entwickelt werden, das die optimale Platzierung der Sender und Sensoren bestimmt.

Basierend auf dem in diesem Abschnitt beschriebenen Verfahren werden in [Monstadt 06] Beschleunigungssensoren verwendet, um das Bewegungsmodell der geoDBNs an die aktuelle Gehgeschwindigkeit des Benutzers anzupassen (z. B. Sitzen, schnelles oder langsames Gehen), indem die CPT des Bewegungsmodells

$$\mathrm{CPT}(\mathsf{UserPos{=}GeoPos?}) = \left[\begin{array}{r|cc} \mathsf{UserPos{=}GeoPos?\_1} & \underline{a_1} & \underline{a_2} \\ \hline a_1 & 0.7 & 0.001 \\ a_2 & 0.3 & 0.999 \end{array}\right]$$

mit der Bewegungsgeschwindigkeit des Benutzers $v_{\mathscr{B}}$ wie folgt parametrisiert wird:

$$\mathrm{CPT}(\mathsf{UserPos{=}GeoPos?}) = \left[\begin{array}{r|cc} \mathsf{UserPos{=}GeoPos?\_1} & \underline{a_1} & \underline{a_2} \\ \hline a_1 & f(v_{\mathscr{B}}) & 0.001 \\ a_2 & 1-f(v_{\mathscr{B}}) & 0.999 \end{array}\right].$$

Die bedingten Wahrscheinlichkeiten werden dabei durch die Funktion $f(.)$ in Abhängigkeit von $v_{\mathscr{B}}$ berechnet. Mit dieser Maßnahme sollte erreicht werden, dass sich die einzelnen geoDBNs umso schneller abbauen, je schneller sich der Benutzer fortbewegt, so dass der Schweif der geoDBNs, der den Benutzer verfolgt, nicht zu lang wird. Wie aber in [Monstadt 06] festgestellt wurde, wird die Berechnung der Position nicht genauer, da sich alle geoDBNs gleich schnell abbauen; insbesondere auch die geoDBNs, die neu instantiiert wurden und sich in der Nähe des Benutzers befinden. Eine mögliche Verbesserung könnte sein, dass neu instantierte geoDBNs oder solche geoDBNs, die sich in der Nähe der aktuell berechneten Benutzerposition befinden, sich nicht so schnell abbauen wie alte geoDBNs bzw. solche geoDBNs, die sich weiter entfernt von der aktuell berechneten Benutzerposition befinden.

In einem weiteren Schritt ließen sich die Sensorknoten parametrisieren, so dass jeder Sensor seine eigene Zuverlässigkeit dem Netz mitteilen kann und die Tabelle bedingter Wahrscheinlichkeiten mit den entsprechenden Werten versorgt. So wäre auch nur noch ein (parametrisierter) Sensorknoten notwendig. Dieses entspricht in der Praxis auch dem Normalfall, da sich an einer Geokoordinate meistens auch nur ein Sender befindet (in unserem Fall entweder ein Infrarot-Sender oder ein RFID-Sender). Aber auch der Fall mit mehreren Sendern an einer Geokoordinate lässt sich leicht realisieren. Werden beispielsweise ein RFID-Sender und ein IR-Sender gleichzeitig empfangen, so wird zuerst eine Zeitscheibe für den RFID-Sender mit den Zuverlässigkeitswerten des RFID-Senders angehängt und dann die neu anzuhängende Zeitscheibe mit den Eigenschaften des IR-Senders. Die Übergangswahrscheinlichkeit des Knotens UserPos=GeoPos? muss dann allerdings als 1:1-Übergang realisiert sein, so dass in diesem Moment der Knoten UserPos=GeoPos? als statischer Knoten dient und beide Sender bzgl. ihrer Sendereigenschaften gleich gewertet werden.

## 7.4 AGENDER – Alters- und Geschlechtserkennung

Das System AGENDER[3] (siehe [Müller 06]) ist ein Beispiel für die ereignisgesteuerte Instantiierung von Zeitscheiben (siehe Abschnitt 5.2.1), da der Auslöser für das Anhängen einer neuen Zeitscheibe eine sprachliche Äußerung des Benutzers ist.

AGENDER schätzt das Alter und das Geschlecht eines Sprechers aufgrund seiner sprachlichen Äußerungen ein. Das System nutzt dabei die in der Sprache enthaltenen paralinguistischen Informationen wie Charakteristika der Stimme (beispielsweise mittlere Grundfrequenz, Jitter und Shimmer) und Charakteristika des Sprechverhaltens (z. B. Artikulationsgeschwindigkeit und Anzahl und Dauer der Sprechpausen), die bzgl. Alter und Geschlecht differieren. Die Einschätzung des Sprecheralters erfolgt dabei in den Altersklassen KINDER (Alter $\leq$ 12 Jahre), JUGENDLICHE (13 Jahre $\leq$ Alter $\leq$ 19 Jahre), ERWACHSENE (20 Jahre $\leq$ Alter $\leq$ 64 Jahre) und SENIOREN (65 Jahre $\leq$ Alter).

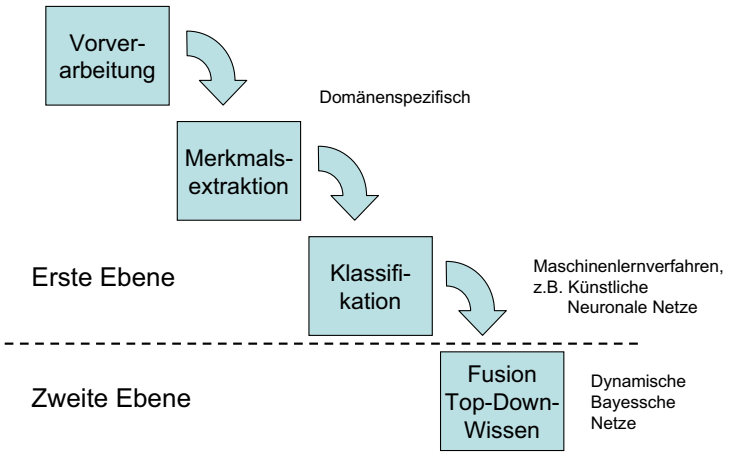



Abbildung 7.13: Die Ebenen der AGENDER-Sprecherklassifikation.

Wie wir in Abbildung 7.13 sehen können, ist das System zweistufig. Die verschiedenen Phasen der Merkmalsextraktion und der Klassifikation werden in AGENDER als *Erste Ebene* bezeichnet. Bezüglich der Klassifikation wurden die folgenden bekannten Methoden des maschinellen Lernens untersucht: 1. Naive Bayes, 2. Gaussian-Mixture-Models, 3. k-Nearest-Neighbor, 4. C 4.5 Entscheidungsbäume, 5. Support-Vector-Machines und 6. Künstliche Neuronale Netze.

In der sogenannten *Zweiten Ebene* werden die Ergebnisse der verschiedenen Klassifizierer miteinander kombiniert. Das Problem der Fusion multipler Klassi-

[3]Die Bezeichnung AGENDER wurde von den englischen Begriffen *age* für Alter und *gender* für Geschlecht abgeleitet.

fikationsergebnisse bezüglich einer Äußerung (statischer Aspekt) oder mehrerer aufeinander folgender Äußerungen (dynamischer Aspekt) wird dabei durch ein dynamisches Bayessches Netz (siehe Abbildung 7.14 für die erste Zeitscheibe des Systems) gelöst. Mit diesem dynamischen Bayesschen Netz wurde sowohl die klassifikationsinhärente Unsicherheit explizit modelliert als auch Top-Down-Wissen in den Entscheidungsprozess eingebunden. Beispielsweise sind je nach Kontext (siehe Knoten (9) tatsächlicher Kontext mit den Zuständen 'quiet', 'noisy' und 'voicy') die Resultate bestimmter Klassifizierer zuverlässiger als die anderer. Es wird zwischen stimmhaften ('voicy') und stimmlosen ('noisy') Lärm unterschieden. Wie man beispielsweise an den bedingten Wahrscheinlichkeiten des Knotens (2) Stimmmerkmalsmodell Gruppierung 5 erkennt, ist sein Klassifikationsergebnis am besten, wenn es leise ('quiet') ist, oder stimmhafter Lärm ('voicy') vorherrscht. Wenn der Lärm stimmlos ist, dann ist sein Klassifikationsergebnis unbrauchbar. Umgekehrt liefert der Knoten (3) Sprecherverhaltensmodell Gruppierung 5 ein gutes Klassifikationsergebnis bei stimmlosen Lärm ('noisy') und liefert in den anderen Fällen kein brauchbares Ergebnis.

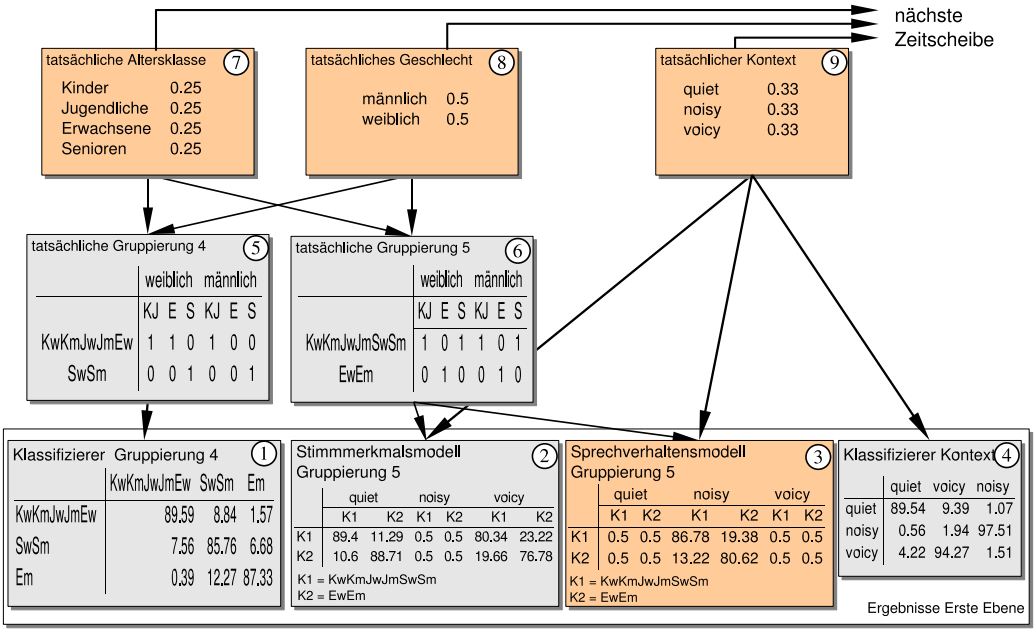



Abbildung 7.14: Erste Zeitscheibe des dynamischen Bayesschen Netzes von AGENDER.

Schauen wir uns Abbildung 7.14 an, in der die erste Zeitscheibe des in AGENDER verwendeten dynamischen Bayesschen Netzes dargestellt ist. In den unteren Knoten (1) Klassifizierer Gruppierung 4, (2) Stimmmerkmalsmodell Gruppierung 5, (3) Sprecherverhaltensmodell Gruppierung 5 und (4) Klassifizierer Kontext werden

die Ergebnisse der ersten Ebene als Evidenz eingetragen. Die den Knoten zugeordneten Tabellen bedingter Wahrscheinlichkeiten modellieren dabei die oben erwähnte klassifikationsinhärente Unsicherheit.

Die Zwischenebene mit den beiden Knoten (5) tatsächliche Gruppierung 4 und (6) tatsächliche Gruppierung 5 ermöglicht es, dass die Werte der Tabelle bedingter Wahrscheinlichkeiten der Knoten (1) Klassifizierer Gruppierung 4, (2) Stimmmerkmalsmodell Gruppierung 5 und (3) Sprecherverhaltensmodell Gruppierung 5 direkt durch das Validierungsergebnis des jeweiligen Klassifizierers bestimmt werden können. Beispielsweise fasst dazu der Knoten (6) tatsächliche Gruppierung 5 die möglichen Kombinationen der Knotenzustände der Knoten (7) tatsächliche Altersklasse und (8) tatsächliches Geschlecht zu neuen Gruppierungen zusammen, indem an der entsprechende Stelle in der Tabelle bedingter Wahrscheinlichkeiten eine Eins oder Null eingetragen ist. Wenn der Klassifizierer mit weiteren Daten trainiert wird und danach bessere Ergebnisse liefert, so können die bedingten Wahrscheinlichkeiten für die entsprechenden Gruppierungen direkt eingetragen werden.

Die Knoten (7) tatsächliche Altersklasse und (8) tatsächliches Geschlecht wurden als statische Knoten implementiert, da sie Benutzereigenschaften (nämlich Alter und Geschlecht) modellieren, die sich während einer Sitzung mit dem System nicht ändern. Ihre Tabellen bedingter Wahrscheinlichkeiten wurden als Übergänge so ausgeführt, dass die einzelnen Messungen in den jeweiligen Knoten gleich gewichtet werden. (Es wurde also die Umwandlung der statischen in dynamische Knoten vorgenommen, wie sie in Abschnitt 5.1.3 beschrieben wird.) Diese statischen Knoten werden mit jeder Äußerung besser und besser eingeschätzt. Der Knoten (9) tatsächlicher Kontext wurde als dynamischer Knoten modelliert, da sich die Eigenschaft, die er modelliert (nämlich, ob es in der Umgebung ruhig ’quiet’ oder laut ’noisy’ ist oder ob in der Umgebung gesprochen (’voicy’) wird), über die Zeit schnell ändern kann.

Für eine detailliertere Beschreibung des dynamischen Bayesschen Netzes wollen wir an dieser Stelle auf [Müller 06] verweisen.

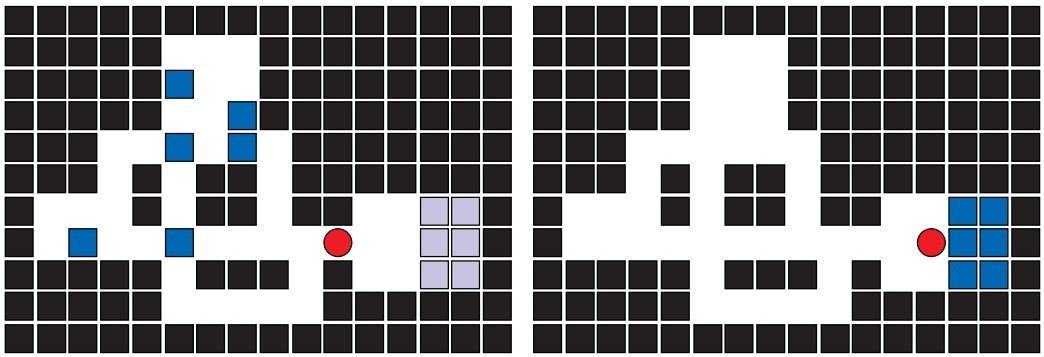
Mit JavaDBN wurde das dynamische Bayessche Netz in Quellcode umgewandelt. Bei einer neuen Äußerung des Sprechers wird eine neue Zeitscheibe angehängt und die alte Zeitscheibe aufgerollt. Dieser Rollup ist ohne Informationsverlust und wird vom Quellcode zur Verfügung gestellt, in den JavaDBN das dynamische Bayessche Netz umgewandelt hat.

Neben den Anpassungen der Klassifizierer an die Ressourcen eines eingebetteten Systems (siehe hierzu [Feld 06]) ermöglicht JavaDBN, dass das System AGENDER auf eingebetteten Systemen lauffähig ist.

## 7.5 Bewusste Steuerung über Muskelanspannung

Wir haben einen Prototypen entworfen, um zu zeigen, wie sich mit nur zwei EMG-Sensoren und Sensorfusion durch ein dynamisches Bayessches Netz eine Steuerung in zwei Dimensionen erzielen lässt. Im folgenden stellen wir den Prototypen anhand des Spieles Sokoban vor.

**Spielesteuerung Sokoban** Beim Spiel *Sokoban* müssen mit der Kugel die dunkelgrauen Steine in die hellgrauen Felder geschoben werden (siehe Abb. 7.15). Die schwarzen Steine sind Wände und können nicht verschoben werden. Die Schwierigkeit des Spiels entsteht dadurch, dass sich mit der Kugel immer nur ein Stein und nie zwei oder mehr Steine schieben lassen, und dass sich die Steine nicht mit der Kugel ziehen lassen. Man muss also aufpassen, dass man (a) die Steine nicht so zusammenschiebt, dass sie sich gegenseitig blockieren, und dass man (b) die Steine nicht in eine ungewollte Ecke schiebt, aus der man sie dann nicht mehr herausziehen kann. In beiden Fällen müsste man das Spiel von neuem starten. In Abbildung 7.15(a) ist ein Beispiel-Spielfeld in der Ausgangsstellung und in Abbildung 7.15(b) in der Endstellung dargestellt.

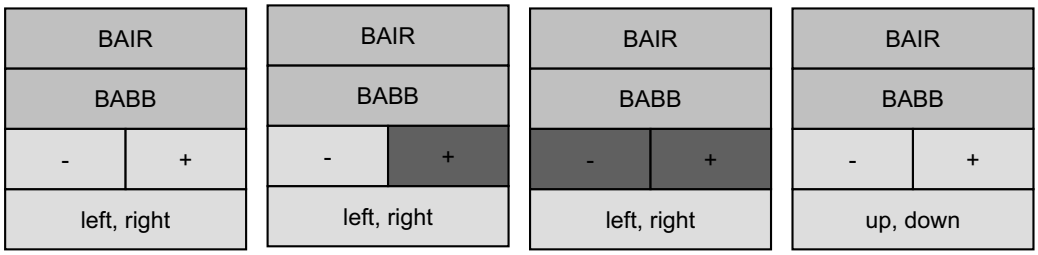


(a) Ausgangsstellung (b) Endstellung

Abbildung 7.15: Sokoban Spielfeld. Mit der Kugel kann immer nur ein dunkelgrauer Stein geschoben werden.

Das besondere an diesem Spiel ist, dass die Kugel über die Muskelanspannung des linken bzw. rechten Unterarmes gesteuert wird. Dazu müssen sowohl am linken als auch am rechten Unterarm jeweils ein EMG-Sensor appliziert werden, um die Intensität der Muskelanspannung messen zu können. Diese Messungen werden durch ein dynamisches Bayessches Netz (siehe Abbildung 7.17) interpretiert und dann umgesetzt. Dieses Bayessche Netz wird später näher erläutert werden, wir wollen jetzt aber erst darauf eingehen, wie sich eine Kugel mit *zwei* Biosen-

soren in *vier* Richtungen steuern lässt. Die Lösung liegt im sogenannten Modus-Wechsel, wie wir ihn im folgenden beschreiben.




(a) links/rechts (b) Schritt nach rechts (c) Moduswechsel (d) hoch/runter

Abbildung 7.16: Steuerung

Wenn keine Muskelanspannung gemessen wird, so bleibt die Kugel ruhig liegen (siehe Abbildung 7.16(a)). Wenn man den linken oder den rechten Unterarm anspannt, indem man beispielsweise eine Faust ballt, so bewegt sich die Kugel nach links bzw. rechts, wenn sich das System im Modus (left, right) befindet (siehe Abbildung 7.16(b), die anzeigt, dass sich die Kugel nach rechts bewegt), oder hoch bzw. runter, wenn sich das System im Modus (up, down) befindet. Man erreicht den Modus-Wechsel zwischen links/rechts und hoch/runter, indem man beide Unterarmmuskeln gleichzeitig anspannt (siehe Abbildung 7.16(c)).

Das dynamische Bayessche Netz (siehe Abbildung 7.17) sieht nun wie folgt aus. Die beiden Knoten linkerArmsensor und rechterArmsensor erhalten als Eingabe die Ergebnisse aus den Sensormessungen der beiden EMG-Sensoren am linken bzw. rechten Unterarm. Die Tabelle bedingter Wahrscheinlichkeiten für den Knoten linkerArmsensor sieht wie folgt aus:

$$\mathrm{CPT}(\mathsf{linkerArmsensor}) = \left[\begin{array}{r|lll} \mathsf{linkerArm} & \text{nicht} & \text{mittel} & \text{stark} \\ \hline \text{nicht} & 0.99 & 0.01 & 0.00 \\ \text{mittel} & 0.01 & 0.98 & 0.01 \\ \text{stark} & 0.00 & 0.01 & 0.99 \end{array}\right].$$

Sie modelliert nicht nur die Zuverlässigkeit der Sensoren sondern auch wie genau man die Sensoren mit der Muskelanspannung ansteuern kann. Mit ein wenig Übung erreicht man sehr schnell ein hohes Maß an Genauigkeit, was die unterschiedlichen Stärken der Muskelanspannung angeht. Die Tabelle bedingter Wahrscheinlichkeiten für den Knoten rechterArmsensor wurde analog modelliert.

In den Knoten linkerArm und rechterArm wird die tatsächliche Muskelanspannung eingeschätzt, aufgrund der in den Knoten linkerArmsensor und rechterArmsensor eingetragenen Evidenzen. Da beide Knoten Wurzelknoten sind, degeneriert die Tabelle bedingter Wahrscheinlichkeiten zu einer Tabelle mit A-priori-

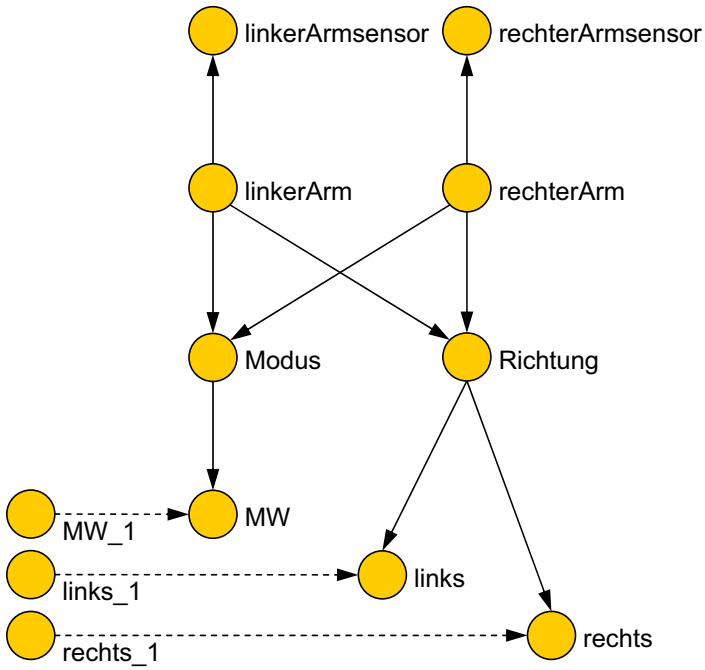



Abbildung 7.17: Zeitscheibenschema TSS_2 des dynamischen Bayesschen Netzes zur Interpretation der Biosignale.

Wahrscheinlichkeiten:

$$\mathrm{CPT}(\{\text{linker/rechter}\}\text{Arm}) = \left[\begin{array}{ccc} \text{nicht} & \text{mittel} & \text{stark} \\ \hline 0.33 & 0.33 & 0.33 \end{array}\right].$$

Diese A-priori-Wahrscheinlichkeiten modellieren eine Gleichverteilung, da jede Muskelanspannung in dieser Anwendung gleich möglich ist.

In Abhängigkeit dieser beiden Wurzelknoten werden die Knoten Modus und Richtung wie folgt eingeschätzt. Schauen wir uns zuerst den Knoten Modus an. Mit ihm wird eingeschätzt, ob beide Arme gleichzeitig angespannt wurden und ein Moduswechsel erwünscht wird. Seine ihm zugeordnete Tabelle bedingter Wahrscheinlichkeiten sieht wie folgt aus:

$$\mathrm{CPT}(\text{Modus}) = \left[\begin{array}{r|ccc|ccc|ccc} \text{rechterArm} & \multicolumn{3}{c|}{\text{nicht}} & \multicolumn{3}{c|}{\text{mittel}} & \multicolumn{3}{c}{\text{stark}} \\ \text{linkerArm} & \text{n} & \text{m} & \text{s} & \text{n} & \text{m} & \text{s} & \text{n} & \text{m} & \text{s} \\ \hline 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ \text{m1} & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \text{m2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

Wenn die Unterarmmuskulatur einer der beiden Arme nicht angespannt ist, so wird die Hypothese “0” eingeschätzt. Wenn beide Arme angespannt (“mittel” oder

"stark") sind, so wird wenigstens die Hypothese "m1" eingeschätzt. Die Hypothese "m2" wird eingeschätzt, wenn beide Arme "stark" angespannt sind.

Die Tabelle bedingter Wahrscheinlichkeiten des Knotens Richtung ist ein wenig komplexer, da der Knoten Richtung im Gegensatz zum Knoten Modus nicht nur drei sondern fünf Hypothesen besitzt:

$$\mathrm{CPT}(\mathsf{Richtung}) = \left[\begin{array}{r|ccc|ccc|ccc} \mathsf{rechterArm} & \multicolumn{3}{c|}{\text{nicht}} & \multicolumn{3}{c|}{\text{mittel}} & \multicolumn{3}{c}{\text{stark}} \\ \mathsf{linkerArm} & \text{n} & \text{m} & \text{s} & \text{n} & \text{m} & \text{s} & \text{n} & \text{m} & \text{s} \\ \hline 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \text{l1} & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \text{l2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{r1} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \text{r2} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right].$$

Es wird eine der Hypothesen "l1", "l2", "r1" oder "r2" eingeschätzt, wenn der entsprechende Arm stärker als der andere Arm angespannt ist. Die Stufe 2 ("l2" oder "r2") wird jeweils nur erreicht, wenn der Arm alleine und "stark" angespannt wird. Werden beide Arme gar nicht oder gleich stark angespannt, dann wird die Hypothese "0" eingeschätzt.

Die Tabelle bedingter Wahrscheinlichkeiten des Knotens MW ist so modelliert, dass sie die direkten Änderungen der Muskelanspannung einschätzt, ohne vorhergehende Muskelanspannungen zu berücksichtigen. Dieses lässt sich direkt aus den Tabelleneinträgen ablesen:

$$\mathrm{CPT}(\mathsf{MW}) = \left[\begin{array}{r|ccc|ccc|ccc} \mathsf{Modus} & \multicolumn{3}{c|}{0} & \multicolumn{3}{c|}{\text{m1}} & \multicolumn{3}{c}{\text{m2}} \\ \mathsf{MW\_1} & 0 & \text{m1} & \text{m2} & 0 & \text{m1} & \text{m2} & 0 & \text{m1} & \text{m2} \\ \hline 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{m1} & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \text{m2} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right].$$

Wie man erkennen kann, ist die Einschätzung einer der Hypothesen ("0", "m1" oder "m2") nur vom Knoten Modus abhängig.

Die folgenden Knoten links und rechts berücksichtigen dagegen die Zustände ihrer Elternknoten der vorhergehenden Zeitscheibe (nämlich den Knoten links_1 bzw. rechts_1). Schauen wir uns dazu die Tabelle bedingter Wahrscheinlichkeiten des Knotens links an:

$$\mathrm{CPT}(\mathsf{links}) = \left[\begin{array}{r|ccccc|ccccc|ccccc} \mathsf{links\_1} & \multicolumn{5}{c|}{0} & \multicolumn{5}{c|}{\text{l1}} & \multicolumn{5}{c}{\text{l2}} \\ \mathsf{Richtung} & 0 & \text{l1} & \text{l2} & \text{r1} & \text{r2} & 0 & \text{l1} & \text{l2} & \text{r1} & \text{r2} & 0 & \text{l1} & \text{l2} & \text{r1} & \text{r2} \\ \hline 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ \text{l1} & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ \text{l2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right].$$

Die Tabelle wurde so modelliert, dass man nicht vom Zustand "l2" direkt zum Zustand "0" bzw. umgekehrt kommt. Analog dazu wurde die Tabelle bedingter

Wahrscheinlichkeiten des Knotens rechts entworfen:

$$\mathrm{CPT}(\mathsf{rechts}) = \left[\begin{array}{r|ccccc|ccccc|ccccc}
\text{rechts_1} & & & 0 & & & & & r1 & & & & & r2 & & \\
\text{Richtung} & 0 & l1 & l2 & r1 & r2 & 0 & l1 & l2 & r1 & r2 & 0 & l1 & l2 & r1 & r2 \\
\hline
0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
r1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
r2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{array}\right].$$

Vom System werden etwa vier mal pro Sekunde die Anspannung der Muskulatur der beiden Unterarme gemessen und als Evidenz in den entsprechenden Sensorknoten eingetragen. Danach wird das dynamische Bayessche Netz ausgewertet, die Einschätzungen ausgelesen und der Rollup ausgeführt. Die ausgelesenen Einschätzungen werden an die Steuerung (siehe Abbildung 7.16) weitergeleitet und dort entsprechend umgesetzt.

Durch den Modus-Wechsel ermöglichen wir dem Benutzer, dass er die Kugel in zwei Dimensionen steuern kann, indem er vom Modus links/rechts zum Modus hoch/runter wechselt. Prinzipiell ließen sich beliebig viele verschiedene Modi realisieren

# Kapitel 8

# Zusammenfassung und Ausblick

## 8.1 Zusammenfassung

In der vorliegenden Arbeit wurde der differentielle Ansatz von Darwiche zur Lösung Bayesscher Netze so erweitert, dass nun auch die speziellen Bedürfnisse dynamischer Bayesscher Netze wie das Anhängen neuer Zeitscheiben oder das Aufrollen alter Zeitscheiben ohne Informationsverlust berücksichtigt werden. Es wurde eine auf dieser theoretischen Ausarbeitung aufbauende Anwendung (*JavaDBN*) entwickelt, die dynamische Bayessche Netze in Polynome umwandelt und für diese dann Quellcode generiert, der die Berechnungen für die Auswertung des Polynoms und den Rollup durchführt, wobei der Quellcode auf die speziellen Bedürfnisse eingebetteter Systeme ausgerichtet ist. Während der Laufzeit wird kein Speicher allokiert oder deallokiert, so dass keine Speicherbereinigung notwenig wird. Anhand des Quellcodes oder auch des Polynoms lässt sich feststellen, wie viele Additionen, Multiplikationen und Divisionen zur Auswertung des Polynoms durchgeführt werden müssen, so dass sich eine obere Laufzeitschranke und der Speicherplatzbedarf für den Quellcode bestimmen lassen. Dieser Quellcode ist auf eingebetteten Systemen lauffähig und kann somit als *integrierte Lösung dynamischer Bayesscher Netze für eingebettete Systeme* angesehen werden.

In der Einleitung wurden insgesamt acht Forschungsfragen formuliert, die wir nun in der Reihenfolge der Kapitel noch einmal kurz beantworten.

In **Kapitel 2** führen wir die grundlegenden Begriffe für Bayessche Netze ein und stellen drei Inferenzverfahren vor, nämlich (a) die Aalborg-Architektur und (b) die Shafer-Shenoy-Architektur, die auf einem sogenannten Junction Tree arbeiten, und (c) den differentiellen Ansatz von Darwiche, der aus solch einem Junction Tree ein faktorisiertes multivariates Polynom bestimmt, das das Bayessche Netz repräsentiert.

In **Kapitel 3** werden die dynamischen Bayesschen Netze motiviert und vor-

gestellt. Sie bauen sich aus sogenannten Zeitscheiben auf, die die Markov-Eigenschaft erfüllen müssen, d. h. es dürfen keine zeitscheibenüberspringende Kanten auftreten. Für das Lösen von dynamischen Bayesschen Netzen vergleichen wir zwei verschiedene Strategien. Die eine Strategie kann als global angesehen werden, bei der das gesamte dynamische Bayessche Netz beim Lösen als ein Bayessches Netz betrachtet wird, das sich nicht aus Zeitscheiben aufbaut. Bei der zweiten Strategie hingegen soll sich der Aufbau des Bayesschen Netzes aus Zeitscheiben auch in der Struktur des Junction Tree widerspiegeln. Wie sich herausstellte, lassen sich mit der zweiten Strategie Zeitscheiben effizient anhängen und Informationsverlust abschneiden. In diesem Zusammenhang wird der Begriff der eingeschränkten Eliminationsreihenfolge eingeführt. Abschließend werden in diesem Kapitel exakte und approximative Rollup-Verfahren vorgestellt und miteinander verglichen.

In **Kapitel 4** zeigen wir, wie Darwiches differentieller Ansatz zur Auswertung Bayesscher Netze für dynamische Bayessche Netze erweitert werden kann und beantworten damit die Frage *"Wie lässt sich der differentielle Ansatz zur Lösung Bayesscher Netze für dynamische Bayessche Netze erweitern?"*. Wir beschreiben die Prozeduren näher, die man benötigt, um die Polynome für dynamische Bayessche Netze bestimmen zu können. Wir können im dynamischen Bayesschen Netz eine Vorwärts- und Rückwärtspropagierung durchführen (und auch eine Kombination der beiden Verfahren). Alte Zeitscheiben und andere überflüssige Netzstrukturen können aufgerollt werden, wobei das Polynom bei konstantem Speicherverbrauch ausgewertet werden kann. Weiterhin haben wir beschrieben, wie das Verfahren an die Ressourcen eines eingebetteten Systems angepasst und auch eine approximierte Inferenz realisiert werden kann, indem die Tabelle, die den Systemzustand widerspiegelt, in kleinere Tabellen aufgespaltet wird.

In **Kapitel 5** haben wir neue Modellierungsstrukturen eingeführt, die den Modellierungsprozess der dynamischen Bayesschen Netze einfacher und aussagekräftiger und ihre Darstellung übersichtlicher gestalten. Es sind dies *statische Knoten*, *zeitscheibenüberspringende Kanten* und *Doppelschemata*. Diese neuen Modellierungsstrukturen erlauben nun Markov-Prozesse *n*-ter Ordnung und erweitern so die dynamischen Bayesschen Netze, deren Zeitscheiben die Markov-Eigenschaft aus Kapitel 3 erfüllen müssen, zu dynamischen Bayesschen Netzen *n*-ter Ordnung. Wir haben gezeigt, dass diese neuen Modellierungsstrukturen die direkte Anwendung der Rollup-Verfahren aus Abschnitt 3.3 verhindern, aber dass im allgemeinen durch graphentheoretische Umformungen der Rollup wieder ermöglicht wird.

Im Fall der zeitscheibenüberspringenden Kanten werden solange Zwischenknoten in der betroffenen Kante eingefügt, bis die Kante in jeder der Zeitscheiben zu liegen kommt, die sie übersprungen hat. Die Probleme, die die Doppelschemata und die statischen Knoten verursachen, wurden gelöst, indem die beiden

Modellierungsstrukturen auf die zeitscheibenüberspringenden Kanten zurückgeführt wurden. Dies beantwortet die Frage, *wie sich der Modellierungsprozess der dynamischen Bayesschen Netze vereinfachen und ihre Darstellung übersichtlicher gestalten lässt*.

Desweiteren haben wir die *Latenz* bei Sensordaten und die Probleme betrachtet, die sich in diesem Zusammenhang mit der Instantiierung von Zeitscheiben und der Inferenz im dynamischen Bayesschen Netz ergeben. Auf die Frage *"Wie lassen sich zusammengehörende Signale, die zeitlich verzögert eintreffen, gemeinsam auswerten?"* schlagen wir unter Berücksichtigung der *ereignis*- und *zeitgesteuerten Instantiierung* verschiedene Lösungsansätze vor, die sich je nach Anwendungsgebiet des dynamischen Bayesschen Netzes unterschiedlich gut eignen. Auch hier treten zeitscheibenüberspringende Kanten auf, die mit den in diesem Kapitel vorgestellten Mitteln umgewandelt werden können, um wieder einen Rollup durchführen zu können.

In **Kapitel 6** wird gezeigt, wie ein dynamisches Bayessches Netz in Quellcode umgewandelt wird, der nicht nur das dynamische Bayessche Netz repräsentiert sondern auch die Inferenz mit dem Rollup alter Zeitscheiben im dynamischen Bayesschen Netz ermöglicht. Da in der Initialisierungsphase alle notwendigen Variablen eingeführt werden und während der Laufzeit durch eine intelligente Wiederverwendung alter Strukturen und Variablen kein Speicher mehr allokiert oder deallokiert wird, wird keine automatische Speicherbereinigung aufgerufen. Dadurch lässt sich der Ressourcenbedarf bezüglich Speicher und Rechenzeit eindeutig abschätzen bzw. bestimmen, was somit die aufgeworfene Frage *"Wie lassen sich dynamische Bayessche Netze effizient auf eingebetteten Systemen auswerten, die nur einen minimalen Speicherplatz oder eine geringe Rechenleistung aufweisen?"* beantwortet. Ein weiterer großer Vorteil des Quellcodes liegt darin, dass er leicht lesbar ist, so dass das Verfahren der Inferenz und des Rollups für den Anwender transparent bleibt. Dadurch lassen sich Veränderungen im Quellcode vornehmen, so dass sich Zusatzfunktionen wie Methodenaufrufe realisieren lassen, die nicht oder nur sehr schwer möglich wären, wenn der Inferenzalgorithmus vorkompiliert ist und das dynamische Bayessche Netz zur Verarbeitung als Eingabe erhält. Dazu haben wir gezeigt, wie sich konstante Tabelleneinträge zu variablen Tabelleneinträgen umwandeln lassen, indem Arrays durch Methodenaufrufe ersetzt werden. Dies ist z. B. dann sinnvoll, wenn Tabelleneinträge nicht im Voraus bekannt sind oder sich zur Laufzeit verändern können. Ein Methodenaufruf könnte beispielsweise auch einen Datenbankzugriff realisieren (um z. B. die Tabelleneinträge aus einem Benutzerprofil einzulesen) oder von der abgelaufenen Zeit seit der letzten Instantiierung abhängig sein. Wir haben damit die Frage *"Wie lassen sich dynamische Bayessche Netze parametrisieren, so dass die mit den Knoten des dynamischen Bayesschen Netzes behafteten Wahrscheinlichkeiten erst zur Laufzeit vorliegen müssen?"* beantwortet.

Weiterhin erzeugt JavaDBN Quellcode für die Sensitivitätsanalyse und unterstützt *MatLab*, was die Frage *"Wie lässt sich in dynamischen Bayesschen Netzen (a) der Einfluss einzelner Knoten und (b) der Einfluss mehrerer Knoten auf die anderen Knoten im Netz bestimmen?"* beantwortet.

Exemplarisch stellen wir in **Kapitel 7** einige Anwendungen vor, die mit *JavaDBN* erzeugt wurden und dabei sowohl die Möglichkeiten von JavaDBN als auch die verschiedenen Sensortypen und ihre Fusion demonstrieren.

In der ersten vorgestellten Anwendung wird das dynamische Bayessche Netz zur Behandlung von motorischen und sprachlichen Symptomen zur Einschätzung der aktuellen Benutzersituation bezüglich Zeitdruck und kognitiver Belastung vorgestellt. In einer ersten Version wurde der Rollup nur approximiert durchgeführt, indem die vorhergehende Zeitscheibe abgeschnitten und die berechneten BEL-Werte der Knoten als A-priori-Wahrscheinlichkeiten in die Knoten der aktuellen Zeitscheibe eingetragen wurden. Dieser Rollup war naturbedingt fehlerbehaftet. Ohne Rollup merkte man allerdings schon nach ca. drei Zeitscheiben eine deutliche Antwortlatenz des Systems. Durch das in dieser Arbeit vorgestellte Verfahren konnten wir einen Rollup ohne Approximation durchführen. Wir ließen den von JavaDBN erzeugten Quellcode sowohl auf einem Laptop als auch auf einem Zaurus SL-5500G (mit Intel StrongARM Prozessor mit einer Taktfrequenz von 206 MHz) laufen. Auf dem Laptop wurden dabei 25.000 Zeitscheiben und auf dem Zaurus 500 Zeitscheiben innerhalb einer Minute mit exaktem Rollup verarbeitet. Dabei wurde die simulierte multimodale Eingabe nicht verwendet und immer wieder dieselbe Evidenz eingetragen. War zuvor die Auswertung des dynamischen Bayesschen Netzes ohne Rollup der zeitbestimmende Faktor, so war es nun im Falle mit dem Rollup die multimodale Eingabe.

Die nächste Anwendung, der Alarm Manager, ist ein Beispiel dafür wie physiologische Daten eines Benutzers durch physiologische Sensoren erfasst und durch ein dynamisches Bayessches Netz in Echtzeit auf einem Zaurus interpretiert werden können, um dem Anwender an seinen aktuellen Zustand angepasste Nachrichten (in diesem Fall eine wichtige Benachrichtigung) zu präsentieren. Als Sensoren wurden ein Beschleunigungssensor, ein Muskeltätigkeitssensor und ein Augenbrauensensor eingesetzt. Mit diesen beiden Anwendungen haben wir gezeigt, dass sich mit dynamischen Bayesschen Netzen prinzipiell *physiologische Signale zur Einschätzung der Ressourcenlage eines Benutzers auf eingebetteten Systemen in Echtzeit auswerten lassen*.

In der Anwendung LORIOT führen wir als Innovation die sogenannten *georeferenzierten dynamischen Bayesschen Netze* (geoDBNs) ein, die es ermöglichen, dass die Benutzerposition auf dem eigenen kleinen Handheld-Gerät berechnet werden kann, ohne dass eine Verbindung zu einem externen Server besteht, auf dem die Positionsberechnung durchgeführt wird. Diese neue Methode benötigt wenig Rechenzeit- und Speicherplatz und ist zusätzlich hoch-skalierbar für die

In- und Outdoor-Positionierung. Momentan werden IR- und RFID-Sensoren verwendet, aber auch andere Sensoren lassen sich problemlos integrieren, indem das dynamische Bayessche Netz um den entsprechenden Sensorknoten erweitert wird.

Das System AGENDER ist ein weiteres Beispiel für die Verarbeitung von physiologischen Daten, die in diesem Fall über ein Mikrophon erfasst und durch Klassifizierer eingeschätzt werden. Das dynamische Bayessche Netz dient zur Fusion der Ergebnisse der verschiedenen Klassifizierer. Mit JavaDBN wurde das dynamische Bayessche Netz in Quellcode umgewandelt, so dass das dynamische Bayessche Netz auf einem eingebetteten System ausgeführt und der Rollup alter Zeitscheiben ohne Informationsverlust durchgeführt werden kann. Damit haben wir die Frage beantwortet, *wie dynamische Bayessche Netze genutzt werden können, um die Position eines Anwenders aufgrund von Sensordaten zu bestimmen, die der persönliche digitale Assistent empfängt*.

In der letzten vorgestellten Anwendung werden EMG-Sensoren zur bewussten Steuerung eingesetzt. Das dynamische Bayessche Netz hilft hier bei der Sensorinterpretation und der Sensorfusion.

In **Kapitel C** stellen wir als Datenrekorder exemplarisch das Varioport mit seinen Sensoren und die von uns entwickelte Software Javario zur Erfassung der Sensordaten auf einem eingebetteten System vor. Wir erklären, wie die Biosensoren an welchen Stellen angebracht werden müssen, so dass bei einer Messung möglichst wenig Artefakte auftreten.

## 8.2 Ausblick

Im folgenden stellen wir die aus der Arbeit entstandenen noch offenen Fragestellungen und mögliche weitere Arbeiten vor.

**Dynamische Bayessche Netze *n*-ter Ordnung** Im Zusammenhang mit Sensordaten und insbesondere biophysiologischen Daten ergeben sich für dynamische Bayessche Netze *n*-ter Ordnung u. a. in der Modellierung neue Fragestellungen.

Wie lässt sich die unterschiedliche Latenz der biophysiologischen Daten im dynamischen Bayesschen Netz *n*-ter Ordnung modellieren bzw. effizient verwalten? Sie verfügen schon über das Konzept der zeitscheibenüberspringenden Kanten, mit der sich rein theoretisch die Latenz modellieren ließe. Allerdings müsste dazu auch im Voraus bekannt sein, innerhalb welcher Zeitabstände ein Rollup durchgeführt bwz. eine neue Zeitscheibe angehängt wird. So erst würde man wissen, bis in welche nachfolgende Zeitscheibe die zeitscheibenüberspringende Kante reichen müsste. Bei der Modellierung mit zeitscheibenüberspringenden Kanten würde man dann aber auch schnell den Überblick verlieren. Außerdem ist durch

die zeitscheibenüberspringende Kante der Zeitabstand festgelegt, innerhalb dessen eine Zeitscheibe aufgerollt bzw. neu angehängt werden muss.
In Abschnitt 5.2.2 haben wir diesbezüglich schon einige Lösungen vorgeschlagen. Um sie umsetzen zu können, müssen im dynamischen Bayesschen Netz *n*-ter Ordnung die Latenz der biophysiologischen Signale und eventuell ihrer Vorverarbeitungsalgorithmen bzw. Klassifizierer und der Zeitabstand im Falle der zeitgesteuerten Instantiierung angegeben sein.

Bei der ereignis- und zeitgesteuerten Instantiierung stellt sich die Frage, wie die Einträge der Tabellen bedingter Wahrscheinlichkeiten angepasst werden müssen, wenn die Instantiierung beispielsweise mit der doppelten als der üblichen Frequenz durchgeführt werden soll, für die das dynamische Bayessche Netz ursprünglich konzipiert wurde.

In einem mobilen Szenario stellt sich insbesondere die Frage, wie sich neue Sensoren bzw. Klassifizierer in ein existierendes dynamisches Bayessches Netz *automatisch* mit ihren Eigenschaften und ihren Hypothesen einbinden lassen, so dass sie genutzt werden können. Beispielsweise könnte ein neuer Sensor verfügbar sein (der z. B. vom Raum zur Verfügung gestellt wird, den man gerade betritt), der bessere Eigenschaften besitzt, als der persönliche Sensor, den man mit sich trägt. Natürlich stellt sich dann auch die Frage, wie man den Sensor wieder aus dem dynamischen Bayesschen Netz entfernt, wenn man den Raum verlässt.

Welche Auswirkungen diese Fragestellungen auf den Quellcode haben, den JavaDBN zu generieren hat, muss ebenfalls noch untersucht werden. Hier müssen auf jeden Fall wohldefinierte Schnittstellen implementiert werden, die einen Datenaustausch zwischen altem ausführbaren Programmcode und neuen Programmcode ermöglichen.

**JavaDBN** Bei JavaDBN muss sowohl die graphische Oberfläche als auch die Verarbeitungskomponente weiterentwickelt werden.

Es gibt bislang noch kein adäquates System zur Modellierung dynamischer Bayesscher Netze *n*-ter Ordnung. Wir verwenden momentan die graphische Oberfläche von Hugin, um Zeitscheiben zu modellieren, die dann in JavaDBN eingelesen werden. Hier wird dann spezifiziert, in welcher Reihenfolge die Zeitscheiben nacheinander instantiiert werden können. Hier lassen sich dann eventuelle Modellierungsfehler in der Struktur des dynamischen Bayesschen Netzes wie fehlende Kanten von einer zur nächsten Zeitscheibe feststellen. Um die Netze korrigieren zu können, müssen sie wieder in Hugin eingeladen werden.

Es muss ein System entwickelt werden, mit dem sich dynamische Bayessche Netze bzw. Ready-Netze in einer gesamtheitlichen Sicht intuitiv modellieren lassen. Dazu müssen sich die Zeitscheiben einzeln modellieren lassen können und gleichzeitig sollen die strukturellen Auswirkungen dieser Modellierung auf das

gesamte dynamische Bayessche Netz angezeigt werden können, d. h. dass beispielsweise eventuell entstehende zeitscheibenüberspringende Kanten auch in den entsprechenden Zeitscheiben zeitgleich dargestellt bzw. angedeutet werden.

Neben den Konzepten, die für die READY-Netze eingeführt wurden wie beispielsweise zeitscheibenüberspringende Kanten oder statische Knoten, sollen auch die Latenz der Sensordaten, die ereignis- bzw. zeitgesteuerte Instantiierung und die in diesem Abschnitt erwähnten neuen Sensorknoten modelliert werden können. Ebenso soll dynamischen Knoten erlaubt sein, Elternknoten zu haben, die sich nicht mehr nur in der aktuellen oder der vorhergehenden Zeitscheibe befinden.

Während dem Entwurf eines dynamischen Bayesschen Netzes ist es ebenso wünschenswert, wenn man den Entwicklungsstand beispielsweise der einzelnen Knoten mit ihren bedingten Wahrscheinlichkeiten, ausgewählter Netzstrukturen oder des gesamten dynamischen Bayesschen Netzes annotieren könnte. Ebenso können Anmerkungen sinnvoll sein, die die Herkunft der Daten bzw. Struktur angeben.

Die bedingten Wahrscheinlichkeiten sollen einmal wie bisher ganz normal über Zahlen angegeben werden können. Zusätzlich soll die Eingabe aber auch intuitiv über die veränderte Intensität einer Farbe –also per malen– in Abhängigkeit der bedingten Wahrscheinlichkeiten erfolgen können. Zusätzlich zu den Zahlen muss es die Oberfläche ermöglichen, dass anstatt Zahlen auch Formeln oder Quellcode, der andere Operationen wie beispielsweise einen Datenbankzugriff ermöglicht, in den Tabellen bedingter Wahrscheinlichkeiten eingegeben werden können.

Die Verarbeitungskomponente ist dahingehend zu erweitern, dass die in der graphischen Oberfläche möglichen Modellierungen auch verarbeitet werden. Zusätzlich sollen durch die Verarbeitungskomponente schnellere Inferenzverfahren zur Verfügung gestellt werden, die dann jedoch nicht mehr so leicht verständlich sind und sich dementsprechend nicht mehr so einfach von Hand anpassen lassen. Dieses Manko wird aber durch die graphische Benutzeroberfläche ausgeglichen, wenn sich in ihr die bedingten Wahrscheinlichkeiten als Formel oder Quellcode parametrisiert eingeben lassen. So wird eine Anpassung von Hand des von JavaDBN generierten Quellcodes überflüssig.

**Positionierungssystem LORIOT** In der Zukunft werden andere Sensortypen in das System integriert werden (z. B. Videokameras, Mikrophon-Arrays). Zudem muss ein System entwickelt werden, das die optimale Platzierung der Sender und Sensoren bestimmt.

In [Monstadt 06] werden basierend auf dem im Abschnitt 7.3 beschriebenen Verfahren Beschleunigungssensoren verwendet, um das Bewegungsmodell der

geoDBNs an die aktuelle Gehgeschwindigkeit des Benutzers anzupassen (z. B. Sitzen, schnelles oder langsames Gehen). Die bedingten Wahrscheinlichkeiten werden dabei durch die Funktion $f(.)$ in Abhängigkeit von $v_{\mathscr{B}}$ berechnet. Mit dieser Maßnahme sollte erreicht werden, dass sich die einzelnen geoDBNs umso schneller abbauen, je schneller sich der Benutzer fortbewegt, so dass der Schweif nicht zu lang wird, der durch die geoDBNs entsteht, die den Benutzer verfolgen. Es wurde aber in [Monstadt 06] festgestellt, dass die Berechnung der Position nicht genauer wird, da sich alle geoDBNs gleich schnell abbauen; insbesondere auch die geoDBNs, die neu instantiiert wurden und sich –im Gegensatz zu den älteren geoDBNs– höchstwahrscheinlich in der Nähe des Benutzers befinden. Eine mögliche Verbesserung könnte sein, wenn sich neu instantiierte geoDBNs oder solche geoDBNs, die sich in der Nähe der aktuell berechneten Benutzerposition befinden, nicht mehr so schnell abbauen wie alte geoDBNs bzw. solche geoDBNs, die sich weiter entfernt von der aktuell berechneten Benutzerposition befinden. Dazu müsste das Bewegungsmodell eines jeden geoDBNs in Abhängigkeit von Zeit und Ort parametrisiert sein, d. h. umso älter das geoDBN ist und je weiter sich das geoDBN von der aktuellen berechneten Position des Benutzers befindet, desto stärker müsste sich die Einschätzung des geoDBNs, dass sich der Anwender an der Position des geoDBNs befindet, von Zeitscheibe zu Zeitscheibe abschwächen.

In einem weiteren Schritt ließen sich auch die Sensorknoten parametrisieren, so dass jeder Sensor seine eigene Zuverlässigkeit dem Netz mitteilen und die Tabelle bedingter Wahrscheinlichkeiten mit den entsprechenden Werten versorgen könnte. So wäre auch nur noch ein (parametrisierter) Sensorknoten notwendig, anstatt wie jetzt immer die beiden RFID- und IR-Sensorknoten. Dieses würde auch dem Normalfall in der Praxis entsprechen, da sich an einer Geokoordinate meistens nur ein Sender befindet. (In unserem Fall wäre dies entweder ein Infrarot- oder ein RFID-Sender.) Aber auch der Fall mit mehreren Sendern an einer Geokoordinate lässt sich leicht realisieren. Würden beispielsweise ein RFID-Sender und ein IR-Sender mit derselben Geokoordinate gleichzeitig empfangen werden, so würde zuerst eine Zeitscheibe für den RFID-Sender mit den Zuverlässigkeitswerten des RFID-Senders angehängt werden und dann die neu anzuhängende Zeitscheibe mit den Eigenschaften des IR-Senders. Die Übergangswahrscheinlichkeiten des Knotens UserPos=GeoPos? müssten dann allerdings als 1:1-Übergang realisiert sein, so dass in diesem Moment der Knoten UserPos=GeoPos? als statischer Knoten dienen würde und beide Sender bzgl. ihrer Sendereigenschaften gleich gewertet würden.

Zusammenfassend kann man sagen, dass LORIOT erweitert und verbessert werden soll. Dazu soll *ein* generierendes dynamisches Bayessches Netz entwickelt werden, dessen Tabellen bedingter Wahrscheinlichkeiten alle parametrisiert sind.

**Inferenzkomponente und Konfliktauflösung für UbisWorld** UbisWorld (siehe [Heckmann 06a]) ist ein Dienst, der Aussagen im allgemeinen und insbesondere über Anwender, Individuen oder Gruppen verwaltet und diese mit einem Mehrwert versieht. Beispielsweise können die Aussagen von verschiedenen Anwendungen untereinander ausgetauscht und verstanden werden. Zusätzlich schützt UbisWorld die *Privatsphäre* des Anwenders, indem Anwendungen bzw. Anwender Zugriff nur auf für sie autorisierte Aussagen erhalten. Weiterhin versieht UbisWorld jede Aussage mit einem Zeitstempel ihrer Eintragung und mit der Information, von welchem Anwender bzw. welcher Anwendung die Eintragung vorgenommen wurde, so dass sich jederzeit nachvollziehen lässt, wer wann die Eintragung vorgenommen hat, und somit *Transparenz* gegeben ist.

Durch die Vielzahl möglicher Anwendungen und Anwender, die Eintragungen vornehmen können, kann es auch immer zu Aussagen kommen, die sich gegenseitig widersprechen. Hier wird eine *Konfliktauflösungskomponente* notwendig, die die der Wahrscheinlichkeit nach zutreffendere Aussage auswählt bzw. eine neue Aussage erzeugt, deren Konfidenzwert sich aus den Konfidenzwerten der sich gegenseitig ausschließenden Aussagen neu berechnet. Ebenso besitzen manche Aussagen in UbisWorld nur eine gewisse Haltbarkeit, so dass sich die Konfidenz in die Aussage mit der Zeit abschwächen muss. Werden jedoch mit der Zeit Aussagen in das System eingetragen, die die Aussage unterstützen, so muss die Konfidenz in dieser Aussage wieder entsprechend anwachsen. Dazu wird in UbisWorld eine *Inferenzkomponente* benötigt.

Wie lassen sich in einem Ontologie-basierten System wie UbisWorld, in dem Anwender bzw. Anwendungen Aussagen im allgemeinen und insbesondere über Individuen oder Gruppen eintragen können, dynamische Bayessche Netze als Inferenzkomponente oder zur Konfliktauflösung einsetzen, um sich gegenseitig unterstützende oder auch sich (teils) widersprechende Aussagen zu einer neuen Aussage zu vereinen?

**Vorverarbeitungsverfahren, Sensordienste und Sensorontologie** Der instrumentierte Raum (siehe [Butz & Krüger 06]) soll sowohl um Biosensoren als auch um Varioport- und Procomp-Geräte erweitert werden. Abgesehen von der Hardware müssen die Konzepte der instrumentierten Umgebung entsprechend ergänzt werden. U. a. ist zu klären, wie eine verteilte Interpretation von Biosensoren im Raum und am Körper des Anwenders und ihre Sensorfusion realisiert werden kann. Dazu müssen Sensordienste und eine Sensorontologie entwickelt werden, in der die Eigenschaften von Sensoren, ihrer Vorverarbeitungsverfahren und Klassifizierer modelliert werden können, so dass sie miteinander vergleichbar werden und ein System sich automatisch die geeignetsten Kandidaten auswählen kann. Ein Teil der Sensordienste und der Sensorontologie umfasst dabei Vorverarbei-

tungsverfahren für Biosensordaten auf eingebetteten Systemen.

# Anhang A

# Mathematische Grundlagen: Wahrscheinlichkeitentabellen und multivariate Polynome

In diesem Anhang wird die *Algebra von Wahrscheinlichkeitentabellen* formal und tiefergehender wie bisher vorgestellt sowie die sogenannten *Multivariaten Polynome* formal eingeführt. Ebenso finden sich hier in einem eigenen Abschnitt wichtige Nebenrechnungen aus verschiedenen Kapiteln dieser Arbeit.

## A.1 Algebra von Wahrscheinlichkeitentabellen

Die A-priori- und bedingten Wahrscheinlichkeiten der Zufallsvariablen müssen dem Knoten zugeordnet werden, der mit der Zufallsvariablen assoziiert ist. In dieser Arbeit geschieht dies durch *Wahrscheinlichkeitentabellen*, die die A-priori- oder bedingten Wahrscheinlichkeiten jeweils einer Zufallsvariablen des Bayesschen Netzes zusammenfasst. Wahrscheinlichkeitentabellen sind bestimmte Tabellen über Knotenmengen. Alle notwendigen Berechnungen können als Berechnungen von Tabellen über Knotenmengen angegeben werden. Im folgenden wird beschrieben, wie mit Tabellen über Knotenmengen gerechnet wird, und welche besonderen Tabellen über Knotenmengen existieren.

***Definition A.1.1 (Tabelle über einer Knotenmenge)***
Seien $K_1, \ldots, K_n$ Knoten und bezeichne $k_{\ell,j_\ell}$ die $j_\ell$-te Hypothese des Knotens $K_\ell$. Weiter habe der Knoten $K_\ell$ genau $|K_\ell|$ Hypothesen. Ein mehrdimensionaler Vektor $A$ für den gilt:

1. Das Vektorelement $A[j_1, \ldots, j_n]$ steht für die Konfiguration $(k_{1,j_1}, \ldots, k_{n,j_n})$ der Knoten $K_1, \ldots, K_n$.

2. Für jede Konfiguration $(k_{1,j_1},\ldots,k_{n,j_n})$ der Knoten $K_1,\ldots,K_n$ existiert ein Vektorelement $A[j_1,\ldots,j_n]$ des mehrdimensionalen Vektors $A$.

wird als *Tabelle über der Knotenmenge* $\mathbb{V}=\{K_1,\ldots,K_n\}$ bezeichnet. Sie wird mit $T_{\mathbb{V}}$ bezeichnet. Die Menge aller möglichen Tabellen über der Knotenmenge $\mathbb{V}=\{K_1,\ldots,K_n\}$ wird mit $\mathbb{K}^{\mathbb{V}}$ bezeichnet, wobei $\mathbb{K}$ eine Menge ist, deren Elemente als Einträge in den Tabellen erlaubt sind. Ein Zugriff auf die Elemente der Tabelle ist durch Angabe eines Indizes möglich: $T(K_1,\ldots,K_n)_{j_1,\ldots,j_n}$ bezeichnet das Tabellenelement, das für die Konfiguration $(k_{1,j_1},\ldots,k_{n,j_n})$ der Knoten $K_1,\ldots,K_n$ steht. Man schreibt auch $T(k_{1,j_1},\ldots,k_{n,j_n})$ anstatt $T(K_1,\ldots,K_n)_{j_1,\ldots,j_n}$. Ist es eindeutig, für welchen Knoten ein Hypothesenname steht, so können die Hypothesennamen für den Zugriff auf die Tabelle auch in beliebiger Reihenfolge angegeben werden.

Ist $(k_{1,j_1},\ldots,k_{n,j_n},\ell_{1,i_1},\ldots,\ell_{m,i_m})$ eine Konfiguration der Knoten $K_1,\ldots,K_n$ und $L_1,\ldots,L_m$, dann gilt:

$$T(K_1,\ldots,K_n)_{k_{1,j_1},\ldots,k_{n,j_n},\ell_{1,i_1},\ldots,\ell_{m,i_m}} = \\ T(K_1,\ldots,K_n)_{k_{1,j_1},\ldots,k_{n,j_n},\widehat{\ell_{1,i_1}},\ldots,\widehat{\ell_{m,i_m}}} = T(K_1,\ldots,K_n)_{k_{1,j_1},\ldots,k_{n,j_n}}.$$

Mit $\widehat{\ell_{1,i_1}}$ wird angegeben, dass der Index $\ell_{1,i_1}$ weggelassen werden kann.

Mit $T_{1(K_1,\ldots,K_n)}$ wird die Tabelle über der Knotenmenge $\{K_1,\ldots,K_n\}$ bezeichnet, die nur Einsen als Einträge in der Tabelle besitzt. D

Mit diesen Bezeichnungen wird nun die *Tabelle bedingter Wahrscheinlichkeiten* und die *Tabelle gemeinsamer Wahrscheinlichkeiten* eingeführt.

***Definition A.1.2 (Tabelle bedingter Wahrscheinlichkeiten)***
Seien $V_1,\ldots,V_n$ die Vorgänger (Elternknoten) des Knotens $K$, bezeichne $v_{\ell,j_\ell}$ die $j_\ell$-te Hypothese des Vorgängerknotens $V_\ell$ $(1\le j\le|V_\ell|)$ und $k_i$ $(i=1,\ldots,r)$ die Hypothesen von $K$, $(|K|=r)$. Eine Tabelle $T(K,V_1,\ldots,V_n)\in\mathbb{K}^{\mathbb{V}}$ mit $\mathbb{V}=\{V_1,\ldots,V_n\}$ und $\mathbb{K}$ ist die Menge der reellen Zahlen von 0 bis 1 ($\mathbb{K}=[0..1]$) für die gilt:

1. $T(K,V_1,\ldots,V_n)_{i,j_1,\ldots,j_n}$ ist das $i,j_1,\ldots,j_n$-te Element mit
$$T(K,V_1,\ldots,V_n)_{i,j_1,\ldots,j_n}=\Pr(k_i\mid v_{1j_1},\ldots,v_{nj_n}).$$
2. $\sum_{i=1}^{r}T(K,V_1,\ldots,V_n)_{i,j_1,\ldots,j_n}=1.$

heißt *Tabelle bedingter Wahrscheinlichkeiten*, die $\Pr(K\mid V_1,\ldots,V_n)$ repräsentiert. Sind die Elternknoten von $K$ bekannt, so schreibt man für die *Tabelle bedingter Wahrscheinlichkeiten von* K auch $\theta_{(K)}$. Sie gibt die Wahrscheinlichkeit für das Eintreffen einer Hypothese $k_i$ des Knotens $K$ in Abhängigkeit von den Zuständen $v_{1,j_1},\ldots,v_{n,j_n}$ der Elternknoten $V_1,\ldots,V_n$ an $(1\le j_\ell\le|V_\ell|,\ (\ell=1,\ldots,n))$. D

***Definition A.1.3 (Tabelle gemeinsamer Wahrscheinlichkeiten)***
Seien die Knoten $K_1,\ldots,K_n$ gegeben, und bezeichne $k_{\ell,j_\ell}$ die $j_\ell$-te Hypothese des Knotens $K_\ell$ $(1 \leq j \leq |K_\ell|)$. Eine Tabelle $T(K_1,\ldots,K_n) \in \mathbb{K}^{\mathbb{V}}$ mit $\mathbb{V} = \{K_1,\ldots,K_n\}$ und $\mathbb{K}$ ist die Menge der reellen Zahlen von 0 bis 1 ($\mathbb{K} = [0..1]$) für die gilt:

1. $T(K_1,\ldots,K_n)_{j_1,\ldots,j_n}$ ist das $j_1,\ldots,j_n$-te Element mit
$$T(K_1,\ldots,K_n)_{j_1,\ldots,j_n} = \Pr(k_{1,j_1},\ldots,k_{n,j_n}).$$
2. $\sum_{1\leq j_1\leq|K_1|,\ldots,1\leq j_n\leq|K_n|} T(K_1,\ldots,K_n)_{j_1,\ldots,j_n} = 1.$

heißt *Tabelle gemeinsamer Wahrscheinlichkeiten* (*joint probability table* (JPT)), die $\Pr(K_1,\ldots,K_n)$ repräsentiert. Sie gibt die Wahrscheinlichkeit für das Eintreffen einer Hypothesenkombination $(k_{1,j_1},\ldots,k_{n,j_n})$ der Knoten $K_1,\ldots,K_n$ an $(1 \leq j_\ell \leq |K_\ell|,\ (\ell = 1,\ldots,n))$. D

Um bei einer Tabelle gemeinsamer Wahrscheinlichkeiten sicher zu stellen, dass sich die Tabelleneinträge insgesamt zu Eins addieren, kann die Tabelle mit einem Normalisierungsfaktor versehen werden. Dieser Normalisierungsfaktor wird auch als *Normalisierungskonstante* $\alpha$ bezeichnet.

***Definition A.1.4 (Wahrscheinlichkeitentabelle)***
Die Tabellen bedingter und die Tabellen gemeinsamer Wahrscheinlichkeiten werden zusammenfassend auch als *Wahrscheinlichkeitentabellen* bezeichnet. D

***Beispiel A.1.5***
Die *Wahrscheinlichkeitsverteilung* $\Pr(A)$ des Knotens $A$ mit den *Zuständen* (*Hypothesen*) $a_1,\ldots,a_n$ ist eine eindimensionale Wahrscheinlichkeitentabelle über dem Knoten $A$: Sei Pr(A) gegeben durch:

$$\Pr(a_1) = x_1,\ldots,\Pr(a_n) = x_n$$

Dann ergibt sich für die Wahrscheinlichkeitentabelle des Knotens $A$ mit den Hypothesen $a_1,\ldots,a_n$:

$$T(A) = \left[\begin{array}{ccc} a_1 & \ldots & a_n \\ \hline x_1 & \ldots & x_n \end{array}\right], \text{ d. h.}$$

der Zustand $a_i$ des Knotens $A$ trifft mit der Wahrscheinlichkeit $x_i$ ein. Bsp

***Definition A.1.6 (Gleichheit von Tabellen)***
Zwei Tabellen $T(K_1,\ldots,K_n)$ und $T(V_1,\ldots,V_n)$ heißen gleich, wenn sie über denselben Knotenmengen gebildet wurden und die sich entsprechenden Tabelleneinträge in beiden Tabellen identisch sind. Man schreibt:

$$T(K_1,\ldots,K_n) = T(V_1,\ldots,V_n).$$

D

Jetzt folgen die verschiedenen Operationen, die auf den Tabellen und die mit den Tabellen notwendig sind. Dazu zählen die *Multiplikation*, die *Division* sowie die *Marginalisation* von Tabellen.

***Definition A.1.7 (Multiplikation von Tabellen)***
Seien $T_{\mathbb{V}}$ und $T_{\mathbb{W}}$ zwei beliebige Tabellen über den Knotenmengen

$$\mathbb{V} = \{A_1, \ldots, A_a, C_1, \ldots, C_c\} \text{ und } \mathbb{W} = \{B_1, \ldots, B_b, C_1, \ldots, C_c\}$$

$$\text{mit } a+c = |\mathbb{V}|,\ b+c = |\mathbb{W}| \text{ und } \mathbb{V} \cap \mathbb{W} = \{C_1, \ldots, C_c\},$$

so nennt man die gemäß

$$K^{\mathbb{V}} \times K^{\mathbb{W}} \to K^{\mathbb{V} \cup \mathbb{W}},$$
$$(T_{\mathbb{V}}, T_{\mathbb{W}}) \mapsto T_{\mathbb{V} \cup \mathbb{W}} = T_{\mathbb{V}} \cdot T_{\mathbb{W}}$$

mit

$$T(A_1, \ldots, A_a, B_1, \ldots, B_b, C_1, \ldots, C_c)_{r_1,\ldots,r_a,s_1,\ldots,s_b,t_1,\ldots,t_c} =$$
$$= T(A_1, \ldots, A_a, C_1, \ldots, C_c)_{r_1,\ldots,r_a,t_1,\ldots,t_c} \cdot T(B_1, \ldots, B_b, C_1, \ldots, C_c)_{s_1,\ldots,s_b,t_1,\ldots,t_c}$$

gebildete Tabelle über der Knotenmenge $\mathbb{V} \cup \mathbb{W}$ das Produkt von $T_{\mathbb{V}}$ und $T_{\mathbb{W}}$. **D**

Bei der Multiplikation von zwei Tabellen entsteht somit eine Tabelle, die um die Dimension der Knoten wächst, die nicht in beiden Tabellen enthalten sind.

Im Folgenden ist hierzu ein Beispiel angegeben:

***Beispiel A.1.8 (Tabellenmultiplikation)***

Sei $T(K_1, K_2) = \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{2,1} & x_{1,1} & x_{2,1} \\ k_{2,2} & x_{1,2} & x_{2,2} \end{array}\right]$ und $T(K_1, K_3) = \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & y_{1,1} & y_{2,1} \\ k_{3,2} & y_{1,2} & y_{2,2} \end{array}\right]$.

Dann gilt:

$$T(K_1, K_2)\ T(K_1, K_3) =$$
$$= \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{2,1} & x_{1,1} & x_{2,1} \\ k_{2,2} & x_{1,2} & x_{2,2} \end{array}\right] \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & y_{1,1} & y_{2,1} \\ k_{3,2} & y_{1,2} & y_{2,2} \end{array}\right] =$$
$$= \left[\begin{array}{cc|cc} & & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & k_{2,1} & z_{1,1,1} = x_{1,1} \cdot y_{1,1} & z_{2,1,1} = x_{2,1} \cdot y_{2,1} \\ & k_{2,2} & z_{1,2,1} = x_{1,2} \cdot y_{1,1} & z_{2,2,1} = x_{2,2} \cdot y_{2,1} \\ k_{3,2} & k_{2,1} & z_{1,1,2} = x_{1,1} \cdot y_{1,2} & z_{2,1,2} = x_{2,1} \cdot y_{2,2} \\ & k_{2,2} & z_{1,2,2} = x_{1,2} \cdot y_{1,2} & z_{2,2,2} = x_{2,2} \cdot y_{2,2} \end{array}\right] = T(K_1, K_2, K_3).$$

**Bsp**

Die *Division* einer beliebigen Tabelle $T_{\mathbb{V}}$ durch eine andere beliebige Tabelle $T_{\mathbb{W}}$ verläuft ähnlich wie die Multiplikation zweier Tabellen, wobei man Vorsicht walten lassen muss, wenn der Divisor Nullen als Tabelleneinträge enthält:

***Definition A.1.9 (Division von Tabellen)***
Seien $T_{\mathbb{V}}$ und $T_{\mathbb{W}}$ zwei beliebige Tabellen über den Knotenmengen

$$\mathbb{V} = \{A_1,\ldots,A_a,C_1,\ldots,C_c\} \text{ und } \mathbb{W} = \{B_1,\ldots,B_b,C_1,\ldots,C_c\}$$
$$\text{mit } a+c = |\mathbb{V}|,\ b+c = |\mathbb{W}| \text{ und } \mathbb{V}\cap\mathbb{W} = \{C_1,\ldots,C_c\},$$

so nennt man die gemäß

$$K^{\mathbb{V}} \times K^{\mathbb{W}} \to K^{\mathbb{V}\cup\mathbb{W}},$$
$$(T_{\mathbb{V}}, T_{\mathbb{W}}) \mapsto T_{\mathbb{V}\cup\mathbb{W}} = \frac{T_{\mathbb{V}}}{T_{\mathbb{W}}}$$

mit

$$\begin{aligned} &T(A_1,\ldots,A_a,B_1,\ldots,B_b,C_1,\ldots,C_c)_{r_1,\ldots,r_a,s_1,\ldots,s_b,t_1,\ldots,t_c} = \\ &\quad = \frac{T(A_1,\ldots,A_a,C_1,\ldots,C_c)_{r_1,\ldots,r_a,t_1,\ldots,t_c}}{T(B_1,\ldots,B_b,C_1,\ldots,C_c)_{s_1,\ldots,s_b,t_1,\ldots,t_c}} = \frac{\text{Dividend}}{\text{Divisor}} = \\ &\quad = \begin{cases} 0 & \text{Dividend} = 0 \wedge \text{Divisor} = 0, \\ \frac{\text{Dividend}}{\text{Divisor}} & \text{Divisor} \neq 0, \\ * \text{ (nicht definiert)} & \text{sonst} \end{cases} \end{aligned}$$

gebildete Tabelle über der Knotenmenge $\mathbb{V}\cup\mathbb{W}$ den Quotient von $T_{\mathbb{V}}$ und $T_{\mathbb{W}}$. **D**

Dieses soll an einem Beispiel verdeutlicht werden:

***Beispiel A.1.10 (Tabellendivision)***

Sei $T(K_1,K_2) = \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{2,1} & x_{1,1} & x_{2,1} \\ k_{2,2} & x_{1,2} & x_{2,2} \end{array}\right] = \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{2,1} & 0 & 4 \\ k_{2,2} & 2 & 8 \end{array}\right]$
und $T(K_1,K_3) = \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & y_{1,1} & y_{2,1} \\ k_{3,2} & y_{1,2} & y_{2,2} \end{array}\right] = \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & 0 & 2 \\ k_{3,2} & 1 & 4 \end{array}\right]$. Dann gilt:

$$\begin{aligned} T(K_1,K_2)/T(K_1,K_3) &= \frac{T(K_1,K_2)}{T(K_1,K_3)} = T(K_1,K_2,K_3) = \\ &= \frac{\left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{2,1} & x_{1,1} & x_{2,1} \\ k_{2,2} & x_{1,2} & x_{2,2} \end{array}\right]}{\left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & y_{1,1} & y_{2,1} \\ k_{3,2} & y_{1,2} & y_{2,2} \end{array}\right]} = \frac{\left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{2,1} & 0 & 4 \\ k_{2,2} & 2 & 8 \end{array}\right]}{\left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & 0 & 2 \\ k_{3,2} & 1 & 4 \end{array}\right]} = \end{aligned}$$

$$= \left[\begin{array}{cc|cc} & & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & k_{2,1} & z_{1,1,1} = x_{1,1}/y_{1,1} & z_{2,1,1} = x_{2,1}/y_{2,1} \\ & k_{2,2} & z_{1,2,1} = x_{1,2}/y_{1,1} & z_{2,2,1} = x_{2,2}/y_{2,1} \\ k_{3,2} & k_{2,1} & z_{1,1,2} = x_{1,1}/y_{1,2} & z_{2,1,2} = x_{2,1}/y_{2,2} \\ & k_{2,2} & z_{1,2,2} = x_{1,2}/y_{1,2} & z_{2,2,2} = x_{2,2}/y_{2,2} \end{array}\right] = \left[\begin{array}{cc|cc} & & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & k_{2,1} & 0 & 2 \\ & k_{2,2} & * & 4 \\ k_{3,2} & k_{2,1} & 0 & 1 \\ & k_{2,2} & 2 & 2 \end{array}\right].$$

Die Tabelle ist insgesamt nicht definiert, da ein Tabelleneintrag undefiniert ist. **Bsp**

***Definition A.1.11 (Marginalisation)***
Sei $T_{\mathbb{V}}$ eine Tabelle über der Knotenmenge $\mathbb{V} = \{A_1, \ldots, A_a, B_1, \ldots, B_b\}$ ($a \in \mathbb{N}^*$, $b \in \mathbb{N}$) und $\mathbb{W} = \{A_1, \ldots, A_a\} \subseteq \mathbb{V}$ eine Teilmenge von $\mathbb{V}$, so nennt man die gemäß

$$K^{\mathbb{V}} \to K^{\mathbb{W}},$$
$$T_{\mathbb{V}} \mapsto T_{\mathbb{W}} = \sum_{\mathbb{V} \setminus \mathbb{W}} T_{\mathbb{V}}$$

mit

$$T(A_1, \ldots, A_a)_{r_1,\ldots,r_a} = \sum_{(s_1,\ldots,s_b)=(1,\ldots,1)}^{(|B_1|,\ldots,|B_b|)} T(A_1, \ldots, A_a, B_1, \ldots, B_b)_{r_1,\ldots,r_a,s_1,\ldots,s_b}$$

gebildete Tabelle $T_{\mathbb{W}}$ die *Marginalisation* von $T_{\mathbb{V}}$. **D**

***Beispiel A.1.12 (Marginalisation)***
Wenn $\Pr(A, B_1, \ldots, B_n)$ gegeben ist, dann berechnet sich die Wahrscheinlichkeitsverteilung $\Pr(A)$ wie folgt:

$$\Pr(a_i) = \sum_{(j_1,\ldots,j_n)=(0,\ldots,0)}^{(|B_1|,\ldots,|B_n|)} \Pr(a_i, b_{1j_1}, \ldots, b_{nj_n}) \quad , \qquad i = 1, \ldots, |A|.$$

Die gesamten Einträge einer Tabelle gemeinsamer Wahrscheinlichkeiten summieren sich zu 1. So kann man bei der Wahrscheinlichkeitsverteilung $\Pr(A)$ eines Knotens $A$ auch von einer Tabelle gemeinsamer Wahrscheinlichkeiten reden. **Bsp**

Eine wichtige Eigenschaft der Marginalisation wird im folgenden Satz beschrieben:

***Satz A.1.13***
Seien $\mathbb{V}$ und $\mathbb{W}$ disjunkte Knotenmengen, d. h. $\mathbb{V} \cap \mathbb{W} = \emptyset$, $\mathbb{U} \subseteq \mathbb{V}$ und seien $T_{\mathbb{V}}$ und $T_{\mathbb{W}}$ Tabellen über diesen Knotenmengen. Dann gilt:

$$\sum_{\mathbb{U}} (T_{\mathbb{W}}\ T_{\mathbb{V}}) = T_{\mathbb{W}}\ \sum_{\mathbb{U}} T_{\mathbb{V}}$$

**S**

Ein Beweis zu diesem Satz lässt sich leicht herleiten und bleibt dem interessierten Leser überlassen. Tabellen können also aus der Marginalisation herausgeholt werden, wenn sie keine Knoten enthalten, über die marginalisiert wird. Dies wird im folgenden Beispiel A.1.14 illustriert.

***Beispiel A.1.14 (Produkt-Marginalisation)***

Sei $T(K_1,K_2) = \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{2,1} & x_{1,1} & x_{2,1} \\ k_{2,2} & x_{1,2} & x_{2,2} \end{array}\right]$ und $T(K_1,K_3) = \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & y_{1,1} & y_{2,1} \\ k_{3,2} & y_{1,2} & y_{2,2} \end{array}\right]$.

Das Produkt beider Tabellen berechnet sich zu

$$T(K_1,K_2)\ \ T(K_1,K_3) = \left[\begin{array}{cc|cc} & & k_{1,1} & k_{1,2} \\ \hline k_{3,1} & k_{2,1} & x_{1,1}\cdot y_{1,1} & x_{2,1}\cdot y_{2,1} \\ & k_{2,2} & x_{1,2}\cdot y_{1,1} & x_{2,2}\cdot y_{2,1} \\ k_{3,2} & k_{2,1} & x_{1,1}\cdot y_{1,2} & x_{2,1}\cdot y_{2,2} \\ & k_{2,2} & x_{1,2}\cdot y_{1,2} & x_{2,2}\cdot y_{2,2} \end{array}\right] = T(K_1,K_2,K_3).$$

Es gilt:

$$\begin{aligned}
\sum_{K_3} T(K_1,K_2)\ T(K_1,K_3) &= \sum_{K_3} T(K_1,K_2,K_3) = \\
&= \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{2,1} & x_{1,1}\cdot y_{1,1}+x_{1,1}\cdot y_{1,2} & x_{2,1}\cdot y_{2,1}+x_{2,1}\cdot y_{2,2} \\ k_{2,2} & x_{1,2}\cdot y_{1,1}+x_{1,2}\cdot y_{1,2} & x_{2,2}\cdot y_{2,1}+x_{2,2}\cdot y_{2,2} \end{array}\right] = \\
&= \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{2,1} & x_{1,1}\cdot (y_{1,1}+y_{1,2}) & x_{2,1}\cdot (y_{2,1}+y_{2,2}) \\ k_{2,2} & x_{1,2}\cdot (y_{1,1}+y_{1,2}) & x_{2,2}\cdot (y_{2,1}+y_{2,2}) \end{array}\right] = \\
&= \left[\begin{array}{c|cc} & k_{1,1} & k_{1,2} \\ \hline k_{2,1} & x_{1,1} & x_{2,1} \\ k_{2,2} & x_{1,2} & x_{2,2} \end{array}\right] \left[\begin{array}{cc} k_{1,1} & k_{1,2} \\ \hline y_{1,1}+y_{1,2} & y_{2,1}+y_{2,2} \end{array}\right] = T(K_1,K_2)\ \sum_{K_3} T(K_1,K_3).
\end{aligned}$$

Bei der Berechnung von $\sum_{K_3} T(K_1,K_2)\ T(K_1,K_3)$ sind 8 Multiplikationen und 4 Additionen notwendig, wohingegen beim Term $T(K_1,K_2)\ \sum_{K_3} T(K_1,K_3)$ nur 4 Multiplikationen und 2 Additionen anfallen. Durch geschicktes Herausziehen von Faktoren aus der Marginalisation wird also der Rechenaufwand enorm verringert. **Bsp**

## A.2 Multivariate Polynome

Im folgenden werden die *multivariaten Polynome* formal definiert und eine Kurzschreibweise eingeführt.

Dann werden mit der Kurzschreibweise notwendige Operationen zwischen multivariaten Polynomen definiert. Dies sind die Addition und die Multiplikation zweier multivariater Polynome miteinander und die Skalarmultiplikation von einer Konstanten mit einem multivariaten Polynom. Für das Verständnis des Lesers trägt die Kurzschreibweise bei, wenn er sich im Anfang bei der Definition der Operationen auf die Betrachtung von "normalen" Polynomen mit einer Variablen (den sogenannten *univariaten Polynomen*) einschränkt. Für die Handhabung der Polynome in einem Rechner wird der Begriff der *Ordnung* eingeführt. Die Polynome können dadurch so in einer Form im Rechner dargestellt werden, dass schnell Vereinfachungen möglich sind. Zum Beispiel lässt sich das Polynom $x \cdot y \cdot z + z \cdot x \cdot y$ vereinfacht als $2 \cdot x \cdot y \cdot z$ darstellen.

Dazu im folgenden ein paar Definitionen und Schreibweisen für Polynome.

***Definition A.2.1 (Polynom)***
Sei $R$ ein kommutativer Ring mit 1. Dann ist

$$f = f(x_1, \ldots, x_n) = \sum_{\kappa_1 \geq 0, \ldots, \kappa_n \geq 0} a_{\kappa_1, \ldots, \kappa_n} \cdot x_1^{\kappa_1} \cdot \ldots \cdot x_n^{\kappa_n}$$

ein Polynom in den Unbestimmten $x_1, \ldots, x_n$ mit Koeffizienten $a_\kappa$ in $R$.
Man schreibt auch:

$$f = \sum_{\kappa, a_\kappa \neq 0} a_\kappa \cdot X^\kappa.$$

Ist $n = 1$, so redet man von *univariaten* Polynomen. Im Falle $n > 1$ heißen die Polynome *multivariat*. **D**

Für die Kurzschreibweise von Polynomen werden *Multiindizes* eingeführt.

***Bemerkung A.2.2 (Multiindizes)***
In der vorhergehenden Definition wird $\kappa$ als ein *Multiindex* verwendet. Es gilt:

1. $\kappa = (\kappa_1, \ldots, \kappa_n) \in \mathbb{N}^n$,
2. $\kappa + \lambda = (\kappa_1 + \lambda_1, \ldots, \kappa_n + \lambda_n)$,
3. $|\kappa| = \kappa_1 + \ldots + \kappa_n$,
4. $a_\kappa = a_{\kappa_1, \ldots, \kappa_n}$ und
5. $X^\kappa = x_1^{\kappa_1} \cdot \ldots \cdot x_n^{\kappa_n}$.

**Bem**

Jetzt muss noch geklärt werden, wie zwei Polynome miteinander addiert bzw. multipliziert werden. Bezüglich der Multiplikation von Polynomen ist zu beachten, dass es zu einem nichtkonstanten Polynom kein Inverses existiert. Die Menge der Polynome zusammen mit den Operationen $+$ und $\cdot$ bildet also keinen Körper; man spricht daher von einem Polynom*ring* $(R[x_1,\ldots,x_n],+,\cdot)$.

***Definition A.2.3 (Polynomring)***
Sei $R$ kommutativer Ring mit 1. Die Menge aller Polynome in $x_1,\ldots,x_n$ mit Koeffizienten in $R$ ($R[x_1,\ldots,x_n]$) bildet zusammen mit

$$\sum_\kappa a_\kappa X^\kappa + \sum_\kappa b_\kappa X^\kappa := \sum_\kappa (a_\kappa + b_\kappa) X^\kappa$$
$$\Big(\sum_\kappa a_\kappa X^\kappa\Big) \cdot \Big(\sum_\lambda b_\lambda X^\lambda\Big) := \sum_\mu \Big(\sum_{\kappa+\lambda=\mu} a_\kappa b_\lambda\Big) X^\mu$$
$$\nu\Big(\sum_\kappa a_\kappa X^\kappa\Big) := \sum_\kappa \nu a_\kappa X^\kappa \quad , \qquad \nu \in R$$

einen *Polynomring* (kommutative $R$-Algebra mit 1). D

Für den Rechner sind die beiden Polynome $2*x+6*y$ und $2*(3*y+x)$ verschieden, obwohl sie dieselben Ergebnisse liefern. Es ist also notwendig, dass eine eindeutige Repräsentation der Polynome gewährleistet wird. Wenn Elemente einer Menge miteinander vergleichbar sind, so spricht man von einer *partielle Ordnung*.

***Definition A.2.4 (teilweise Ordnung)***
Sei $X$ eine Menge. Eine Relation $R = X \times X$ heißt *teilweise Ordnung* auf $X$, falls gilt:

1. $(x,x) \in R \quad \forall x \in X$
2. $(x,y) \in R \wedge (y,x) \in R \Rightarrow y = x$
3. $(x,y) \in R \wedge (y,z) \in R \Rightarrow (x,z) \in R$

Wir sprechen dann auch von einer teilweisen Ordnung $\succ$ (abkürzend für $\succ_R$) auf $R$ und schreiben $x \preceq y$ oder $y \succeq x$, falls $(x,y) \in R$ bzw. $x \prec y$ oder $y \succ x$, falls $x \preceq y$ und $x \neq y$. D

Für den besonderen Fall, dass alle Elemente einer Menge untereinander vergleichbar sind, so bezeichnet man die teilweise Ordnung auch als *totale Ordnung*.

***Definition A.2.5 (totale Ordnung)***
Eine teilweise Ordnung $\succ$ auf $X$ heißt *totale Ordnung*, falls sich je zwei Elemente von $X$ bzgl. $\succ$ vergleichen lassen:

$$x,y \in X \Rightarrow x \prec y \text{ oder } x = y \text{ oder } x \succ y.$$

D

# Anhang B

# Bayessche Netze: Architekturen und Algorithmen

Im folgenden wird für die Aalborg- und die Shafer-Shenoy-Architektur jeweils dasselbe Beispielnetz durchgerechnet. Ein Abschnitt über das D-Separations-Kriterium rundet diesen Anhang ab.

## B.1 Aalborg-Architektur

In diesem Anhang wird ein Beispiel für die Aalborg-Architektur umfassend vorgerechnet. Die einzelnen Schritte der beiden Phasen Collect Evidence und Distribute Evidence werden sowohl bei der Initialisierung wie auch bei der Auffrischung des Junction Trees einmal in einer Liste und auch graphisch dargestellt. Konkrete Berechnungen mit Zahlenwerten, wie zum Beispiel die Absorption von einer Clique durch eine andere Clique, runden das Beispiel ab.

***Beispiel B.1.1 (Asienbesuch (Aalborg))***
In Abbildung B.1 ist ein weiterer Junction Tree für das Bayessche Netz Asienbesuch in Abbildung 2.2 angegeben.

Bevor am Junction Tree aus Abbildung B.1 nun die beiden Phasen Collect Evidence und Distribute Evidence mit Absorption vorgestellt werden, müssen noch die Knoten des Bayesschen Netzes den Cliquen des Junction Trees eindeutig zugeordnet werden. Die Knoten $A$ und $B$ werden der Clique $Clq_6$, der Knoten $C$ der Clique $Clq_5$, die Knoten $E$, $F$ und $G$ der Clique $Clq_3$, der Knoten $D$ der Clique $Clq_1$ zugeordnet. In Abbildung B.1 sind die Knoten der Clique unterstrichen, die der Clique zugeordnet sind. In den Cliquen des Junction Trees stehen nun $\psi$-Tabellen. Die Clique $Clq_1$ enthält beispielsweise die $\psi$-Tabelle $T_{\psi(Clq_1)} = T_{1(CD)}\ \theta_{(D)}$. $\theta_{(D)}$ repräsentiert dabei die bedingten Wahrscheinlichkeiten des Knotens $D$, nämlich $\Pr(D \mid C)$. Die jeweilige Berechnungsvorschrift für

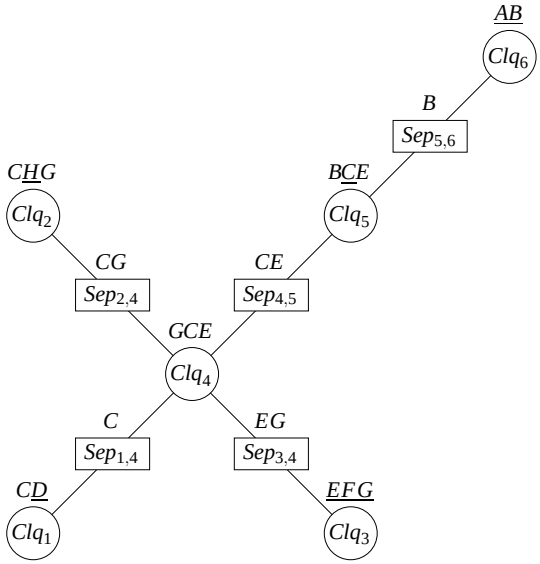



Abbildung B.1: Weiteres Beispiel eines Junction Trees für das Beispielnetz Asienbesuch

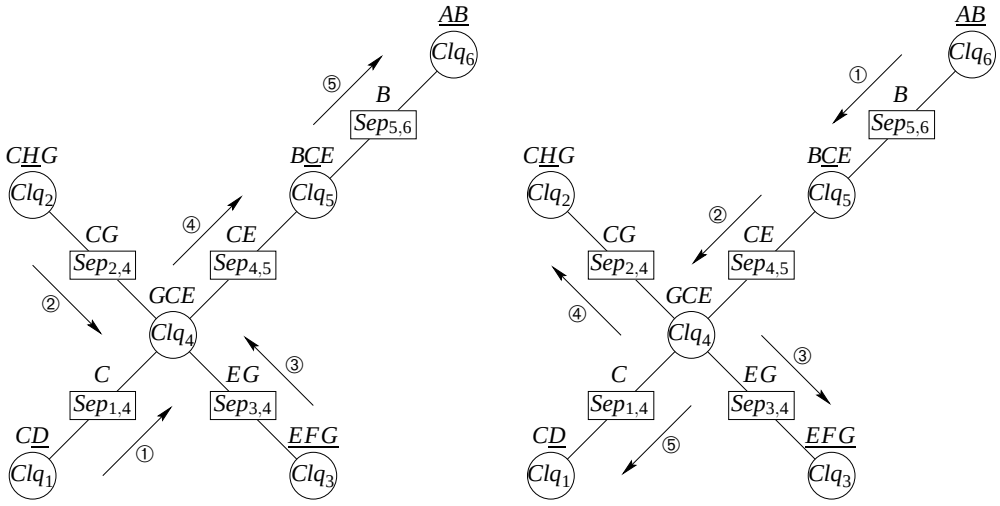
die $\psi$-Tabelle der anderen Cliquen sind in Tabelle B.1 dargestellt. Die der Clique zugeordneten Knoten sind dabei unterstrichen. In den Separatoren des Junction Trees stehen Wahrscheinlichkeitentabellen über den Knoten, die in den Separatoren enthalten sind. Alle Einträge in den Tabellen sind dabei 1.

| *Clique* | *Berechnungsvorschrift der $\psi$-Tabelle* |
|---|---|
| $Clq_1 = \{\underline{D}, C\}$ | $T_{\psi(Clq_1)} = T_{1(CD)}\ \theta_{(D)}$ |
| $Clq_2 = \{C, G, \underline{H}\}$ | $T_{\psi(Clq_2)} = T_{1(CGH)}\ \theta_{(H)}$ |
| $Clq_3 = \{\underline{E}, \underline{F}, \underline{G}\}$ | $T_{\psi(Clq_3)} = T_{1(EFG)}\ \theta_{(E)}\ \theta_{(F)}\ \theta_{(G)}$ |
| $Clq_4 = \{E, C, G\}$ | $T_{\psi(Clq_4)} = T_{1(CEG)} = 1^{\lvert E\rvert\times\lvert C\rvert\times\lvert G\rvert}$ |
| $Clq_5 = \{B, \underline{C}, E\}$ | $T_{\psi(Clq_5)} = T_{1(BCE)}\ \theta_{(C)}$ |
| $Clq_6 = \{\underline{A}, \underline{B}\}$ | $T_{\psi(Clq_6)} = T_{1(AB)}\ \theta_{(A)}\ \theta_{(B)}$ |

Tabelle B.1: Aalborg: $\psi$-Tabellen der Cliquen nach der Konstruktion des Junction Trees

**Initialisierung.** Wählt man als Wurzelknoten die Clique $Clq_6$, dann sieht *Collect Evidence* bei der Initialisierung des Junction Trees wie folgt aus (siehe auch Abbildung B.2(a)):

1. Die Cliquen $Clq_1$, $Clq_2$ und $Clq_3$ schicken ihre Nachricht an die Clique




(a) Collect Evidence (b) Distribute Evidence

Abbildung B.2: Aalborg: Initialisierung eines Junction Trees

$Clq_4$. (Man sagt auch: Die Clique $Clq_4$ absorbiert von den Cliquen $Clq_1$, $Clq_2$ und $Clq_3$.)

2. Jetzt schickt Clique $Clq_4$ ihre Nachricht an Clique $Clq_5$.
3. Darauf schickt Clique $Clq_5$ ihre Nachricht an Clique $Clq_6$ und Collect Evidence terminiert.

Daraufhin erfolgt *Distribute Evidence* (siehe auch Abbildung B.2(b)):

1. Die Clique $Clq_6$ schickt nun ihrerseits ihre Nachricht an Clique $Clq_5$.
2. Die Clique $Clq_5$ schickt ihre Nachricht an Clique $Clq_4$.
3. Zum Abschluss verschickt Clique $Clq_4$ ihre Nachrichten an die Cliquen $Clq_1$, $Clq_2$ und $Clq_3$, wobei die Nachrichten an die verschiedenen Cliquen unterschiedlich sind. Distribute Evidence terminiert.

Exemplarisch wird nun gezeigt, wie sich die $\psi$-Tabelle der Clique $Clq_4$ in der Phase Distribute Evidence berechnet:

1. Im Anfang[1] bestimmt sich die $\psi$-Tabelle der Clique $Clq_4$ zu $T_{\psi(Clq_4)}$:

$$T_{\psi(Clq_4)} = \left[\begin{array}{cc|cc} & & c_1 & c_2 \\ \hline e_1 & g_1 & 0.0315 & 0.0315 \\ & g_2 & 0.0235 & 0.0235 \\ e_2 & g_1 & 0.4185 & 0.4185 \\ & g_2 & 0.5265 & 0.5265 \end{array}\right].$$

[1] nach Collect Evidence

2. Clique $Clq_4$ erhält nun durch Absorption die Information der Clique $Clq_5$ wie folgt:

   (a) Clique $Clq_5$ enthält die Tabelle $T_{\psi(Clq_5)}$:

$$T_{\psi(Clq_5)} = \left[\begin{array}{cc|cc} & & b_1 & b_2 \\ \hline c_1 & e_1 & 0.000572 & 0.054428 \\ & e_2 & 0.009828 & 0 \\ c_2 & e_1 & 0 & 0 \\ & e_2 & 0 & 0.935172 \end{array}\right].$$

   (b) Es wird die Tabelle $T^*_{\psi(Sep_{4,5})}$ berechnet (Marginalisation):

$$T^*_{\psi(Sep_{4,5})} = \sum_B \left[\begin{array}{cc|cc} & & b_1 & b_2 \\ \hline c_1 & e_1 & 0.000572 & 0.054428 \\ & e_2 & 0.009828 & 0 \\ c_2 & e_1 & 0 & 0 \\ & e_2 & 0 & 0.935172 \end{array}\right] =$$

$$= \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline e_1 & 0.000572 + 0.054428 & 0 + 0 \\ e_2 & 0.009828 + 0 & 0 + 0.935172 \end{array}\right] =$$

$$= \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline e_1 & 0.055 & 0 \\ e_2 & 0.009828 & 0.935172 \end{array}\right] = \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline e_1 & 0.055 & 0 \\ e_2 & 0.009\underline{8} & 0.935\overline{2} \end{array}\right]$$

   (c) Nun wird die neue $\psi$-Tabelle des Separators durch die alte $\psi$-Tabelle des Separators dividiert und der $\psi$-Tabelle der Clique $Clq_4$ hinzumultipliziert (Absorption):

$$\left[\begin{array}{cc|cc} & & c_1 & c_2 \\ \hline e_1 & g_1 & 0.0315 & 0.0315 \\ & g_2 & 0.0235 & 0.0235 \\ e_2 & g_1 & 0.4185 & 0.4185 \\ & g_2 & 0.5265 & 0.5265 \end{array}\right] \frac{\left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline e_1 & 0.055 & 0 \\ e_2 & 0.009828 & 0.935172 \end{array}\right]}{\left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline e_1 & 0.055 & 0.055 \\ e_2 & 0.945 & 0.945 \end{array}\right]} =$$

$$= \left[\begin{array}{cc|cc} & & c_1 & c_2 \\ \hline e_1 & g_1 & 0.0315 & 0.0315 \\ & g_2 & 0.0235 & 0.0235 \\ e_2 & g_1 & 0.4185 & 0.4185 \\ & g_2 & 0.5265 & 0.5265 \end{array}\right] \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline e_1 & 1 & 0 \\ e_2 & 0.0104 & 0.9896 \end{array}\right] =$$

$$= \left[\begin{array}{cc|cc} & & c_1 & c_2 \\ \hline e_1 & g_1 & 0.0315 & 0 \\ & g_2 & 0.0235 & 0 \\ e_2 & g_1 & 0.0043524 & 0.4141476 \\ & g_2 & 0.0054756 & 0.5210244 \end{array}\right].$$

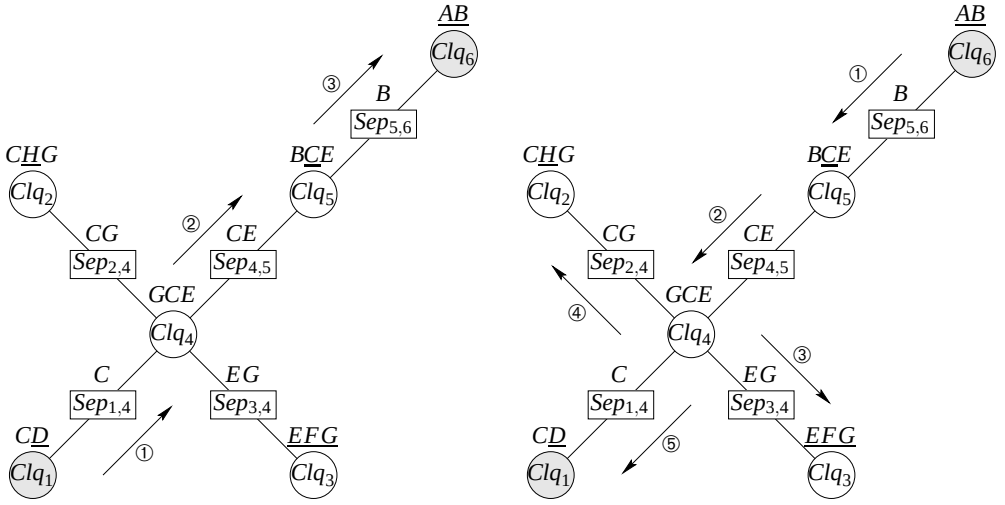
3. Die neue $\psi$-Tabelle des Separators $Sep_{4,5}$ bestimmt sich zu:

$$T_{\psi(Sep_{4,5})} = \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline e_1 & 0.055 & 0 \\ e_2 & 0.009828 & 0.935172 \end{array}\right].$$

4. Die neue $\psi$-Tabelle der Clique $Clq_4$ bestimmt sich zu:

$$T_{\psi(Clq_4)} = \left[\begin{array}{cc|cc} & & c_1 & c_2 \\ \hline e_1 & g_1 & 0.0315 & 0 \\ & g_2 & 0.0235 & 0 \\ e_2 & g_1 & 0.0043524 & 0.4141476 \\ & g_2 & 0.0054756 & 0.5210244 \end{array}\right].$$

**Auffrischung.** Es liegen nun die folgenden Beobachtungen vor: Der Patient hat vor kurzem Asien besucht aber sein Röntgenbefund ist negativ. Als *findings* ergeben sich die folgenden Tabellen $\lambda_A = \left[\frac{a_1 \quad a_2}{1 \quad 0}\right]$ und $\lambda_D = \left[\frac{d_1 \quad d_2}{0 \quad 1}\right]$, die den $\psi$-Tabellen der Cliquen $Clq_6$ und $Clq_1$ hinzu multipliziert werden. Die Cliquen des Junction Trees sind jetzt nicht mehr *konsistent* zueinander, d. h. die Information, die eine Clique über einen Knoten enthält, stimmt nicht mehr mit der Information einer anderen Clique überein, in der sich ebenfalls dieser Knoten befindet. Man sagt auch, dass der Junction Tree nicht mehr *kalibriert* ist. Es erfolgt eine *Auffrischung* auf dem Junction Tree. *Collect Evidence* sieht nun wie folgt aus (siehe auch Abbildung B.3(a)):




(a) Collect Evidence (b) Distribute Evidence

Abbildung B.3: Aalborg: Auffrischung eines Junction Trees nach der Eingabe von neuen Beobachtungen (durch grau schraffierte Kreise gekennzeichnet)

1. Clique $Clq_1$ schickt ihre Nachricht an Clique $Clq_4$. Von den Cliquen $Clq_2$ und $Clq_3$ wird keine Nachricht an Clique $Clq_4$ geschickt, da der Junction Tree vor dem Eintragen der Evidenz schon kalibriert war.
2. Jetzt schickt Clique $Clq_4$ ihre Nachricht an Clique $Clq_5$.
3. Darauf schickt Clique $Clq_5$ ihre Nachricht an Clique $Clq_6$ und Collect Evidence terminiert.

Daraufhin erfolgt *Distribute Evidence* (siehe auch Abbildung B.3(b)):

1. Clique $Clq_6$ schickt nun ihrerseits ihre Nachricht an Clique $Clq_5$.
2. Clique $Clq_5$ schickt ihre Nachricht an Clique $Clq_4$.
3. Clique $Clq_4$ verschickt zum Abschluss Nachrichten an die Cliquen $Clq_1$, $Clq_2$ und $Clq_3$. Die Nachrichten an die verschiedenen Cliquen sind dabei unterschiedlich. Distribute Evidence terminiert.

Der Junction Tree ist nun wieder kalibriert.

Die BEL-Werte für die einzelnen Knoten des Bayesschen Netzes können aus jeder $\psi$-Tabelle, in der sich der Knoten befindet, durch Marginalisation bestimmt werden. Für die Knoten $C$ und $H$ wird dies hier exemplarisch vorgestellt:

$$\begin{aligned}
\mathrm{BEL}(C) &= \alpha \sum_{Clq_1 \setminus \{C\}} T_{\psi(Clq_1)} = \alpha \sum_{Clq_1 \setminus \{C\}} \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline d_1 & 0 & 0 \\ d_2 & 0.00002045 & 0.008528622 \end{array}\right] \\
&= \alpha \sum_{Sep_{1,4} \setminus \{C\}} T_{\psi(Sep_{1,4})} = \alpha \sum_{Sep_{1,4} \setminus \{C\}} \left[\begin{array}{cc} c_1 & c_2 \\ \hline 0.00002045 & 0.008528622 \end{array}\right] \\
&= \alpha \begin{pmatrix} 0.00002045 \\ 0.008528622 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
\mathrm{BEL}(H) &= \alpha \sum_{Clq_2 \setminus \{H\}} T_{\psi(Clq_2)} = \alpha \sum_{Clq_2 \setminus \{H\}} \left[\begin{array}{cc|cc} & & c_1 & c_2 \\ \hline h_1 & g_1 & 0.000009436 & 0.003021569 \\ & g_2 & 0.000006975 & 0.000475166 \\ h_2 & g_1 & 0.000001048 & 0.000755392 \\ & g_2 & 0.000002989 & 0.004276495 \end{array}\right] \\
&= \alpha \begin{pmatrix} 0.0035131457 \\ 0.005035924 \end{pmatrix}
\end{aligned}$$

Wie man sieht, müssen die durch die Marginalisation gewonnenen Ergebnisse noch normalisiert werden, damit sich die Werte für die verschiedenen Hypothesen zu Eins summieren. Dass der Vektor noch normalisiert werden muss, wird durch den Normalisierungsfaktor $\alpha$ angezeigt. Damit ergeben sich die BEL-Werte $\mathrm{BEL}(C) = \begin{pmatrix} 0.0024 \\ 0.9976 \end{pmatrix}$ und $\mathrm{BEL}(H) = \begin{pmatrix} 0.4109 \\ 0.5891 \end{pmatrix}$.

**Bsp**

## B.2 Shafer-Shenoy-Architektur

In diesem Anhang wird ein Beispiel für die Shafer-Shenoy-Architektur umfassend vorgerechnet. Die einzelnen Schritte der beiden Phasen Collect Evidence und Distribute Evidence werden sowohl bei der Initialisierung wie auch bei der Auffrischung des Junction Trees einmal in einer Liste und auch graphisch dargestellt. Konkrete Berechnungen mit Zahlenwerten, wie zum Beispiel die Absorption von einer Clique durch eine andere Clique, runden das Beispiel ab.

Damit die Beispielrechnungen mit den Beispielrechnungen in Kapitel B.1 vergleichbar sind, wurde wieder dasselbe Beispiel gewählt.

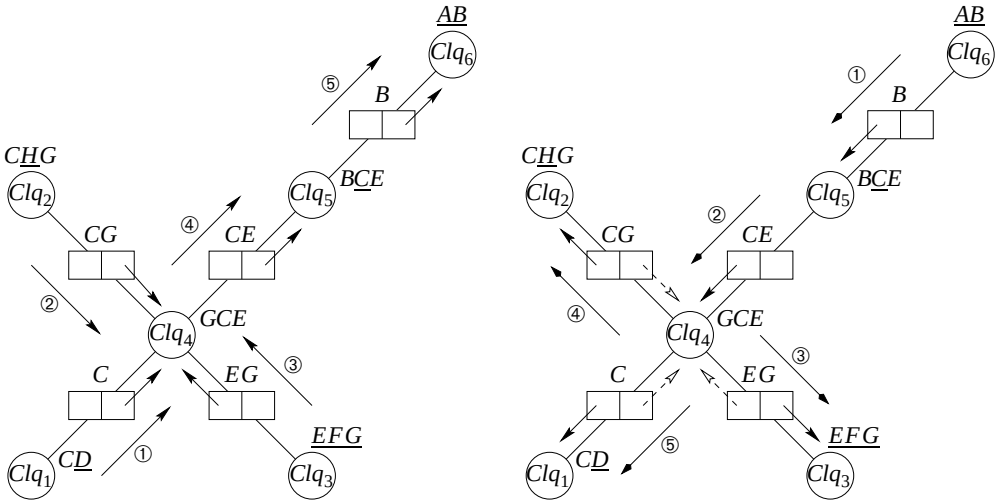
***Beispiel B.2.1 (Asienbesuch (Shafer-Shenoy))***
Für die Inferenz wird wieder der Junction Tree in Abbildung B.1 für das Bayessche Netz Asienbesuch in Abbildung 2.2 genommen. Die Knoten des Bayesschen Netzes seien wie im Beispiel B.1.1 den Cliquen des Junction Trees zugeordnet.

In den Cliquen des Junction Trees stehen nun die Berechnungsvorschriften für die $\psi$-Tabellen: Clique $Clq_1$ enthält z. B. $T_{\psi(Clq_1)} = T_{1(CD)}\ \theta_{(D)}\ T_{Clq_4 \triangleright Clq_1}$ als Berechnungsvorschrift für die $\psi$-Tabelle. $T_{1(CD)}$ stellt dabei sicher, dass $T_{\psi(Clq_1)}$ eine Tabelle über den beiden Knoten $C$ und $D$ ist. $\theta_{(D)}$ repräsentiert dabei die bedingten Wahrscheinlichkeiten des Knotens $D$, nämlich $\Pr(D \mid C)$. Die Tabelle $T_{Clq_4 \triangleright Clq_1}$ enthält die Information, die Clique $Clq_4$ an Clique $Clq_1$ schickt. In den Separatoren des Junction Trees stehen jeweils zwei Wahrscheinlichkeitentabellen über den Knoten, die in den Separatoren enthalten sind. Alle Einträge in den Tabellen sind dabei 1.

| *Clique* | *Berechnungsvorschrift für die $\psi$-Tabelle* |
|---|---|
| $Clq_1 = \{\underline{D}, C\}$ | $T_{\psi(Clq_1)} = T_{1(CD)}\ \theta_{(D)}\ T_{Clq_4 \triangleright Clq_1}$ |
| $Clq_2 = \{C, G, \underline{H}\}$ | $T_{\psi(Clq_2)} = T_{1(CGH)}\ \theta_{(H)}\ T_{Clq_4 \triangleright Clq_2}$ |
| $Clq_3 = \{\underline{E}, \underline{F}, \underline{G}\}$ | $T_{\psi(Clq_3)} = T_{1(EFG)}\ \theta_{(E)}\ \theta_{(F)}\ \theta_{(G)}\ T_{Clq_4 \triangleright Clq_3}$ |
| $Clq_4 = \{E, C, G\}$ | $T_{\psi(Clq_4)} = T_{1(CEG)}\ T_{Clq_1 \triangleright Clq_4}\ T_{Clq_2 \triangleright Clq_4}\ T_{Clq_3 \triangleright Clq_4}\ T_{Clq_5 \triangleright Clq_4}$ |
| $Clq_5 = \{B, \underline{C}, E\}$ | $T_{\psi(Clq_5)} = T_{1(BCE)}\ \theta_{(C)}\ T_{Clq_4 \triangleright Clq_5}\ T_{Clq_6 \triangleright Clq_5}$ |
| $Clq_6 = \{\underline{A}, \underline{B}\}$ | $T_{\psi(Clq_6)} = T_{1(AB)}\ \theta_{(A)}\ \theta_{(B)}\ T_{Clq_5 \triangleright Clq_6}$ |

Tabelle B.2: Shafer-Shenoy: $\psi$-Tabellen der Cliquen nach der Konstruktion des Junction Trees

**Initialisierung.** Wählt man als Wurzel die Clique $Clq_6$, dann sieht *Collect Evidence* bei der Initialisierung des Junction Trees wie folgt aus (siehe auch Abbildung B.4(a)):




(a) Collect Evidence (b) Distribute Evidence

Abbildung B.4: Shafer-Shenoy: Initialisierung eines Junction Trees

1. Die Cliquen $Clq_1$, $Clq_2$ und $Clq_3$ schicken ihre Nachricht an die Clique $Clq_4$.
2. Jetzt schickt Clique $Clq_4$ ihre Nachricht an Clique $Clq_5$.
3. Darauf schickt Clique $Clq_5$ ihre Nachricht an Clique $Clq_6$ und Collect Evidence terminiert.

Daraufhin erfolgt *Distribute Evidence* (siehe auch Abbildung B.4(b)):

1. Die Clique $Clq_6$ schickt nun ihrerseits ihre Nachricht an Clique $Clq_5$.
2. Die Clique $Clq_5$ schickt ihre Nachricht an Clique $Clq_4$.
3. Zum Abschluss verschickt Clique $Clq_4$ ihre Nachrichten an die Cliquen $Clq_1$, $Clq_2$ und $Clq_3$. Distribute Evidence terminiert.

Beispielhaft zeigen wir nun, wie sich die $\psi$-Tabelle der Clique $Clq_4$ in der Phase Distribute Evidence berechnet, wobei es hier vor allem darauf ankommt, Tabelle $T_{Clq_5 \triangleright Clq_4}$ zu bestimmen:

1. Im Anfang bestimmt sich Tabelle $T_{Clq_6 \triangleright Clq_5}$ zu $\left[\begin{array}{cc} b_1 & b_2 \\ \hline 0.0104 & 0.9896 \end{array}\right]$.
2. Clique $Clq_4$ erhält nun durch Absorption die Information ($T_{Clq_5 \triangleright Clq_4}$) der Clique $Clq_5$ wie folgt:

$$T_{Clq_5 \triangleright Clq_4} = \sum_{Clq_5 \setminus Clq_4} T_{\psi(Clq_5 \setminus Clq_4)} = \sum_{Clq_5 \setminus Clq_4} \theta_{(C)}\, T_{Clq_6 \triangleright Clq_5} =$$

$$= \sum_B \left[\begin{array}{ccc|cc} & & & c_1 & c_2 \\ \hline b_1 & e_1 & & 1 & 0 \\ & e_2 & & 1 & 0 \\ b_2 & e_1 & & 1 & 0 \\ & e_2 & & 0 & 1 \end{array}\right] \left[\begin{array}{cc} b_1 & b_2 \\ \hline 0.0104 & 0.9896 \end{array}\right] = \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline e_1 & 1 & 0 \\ e_2 & 0.0104 & 0.9896 \end{array}\right]$$

3. Der Wert der $\psi$-Tabelle bestimmt sich damit zu (Man beachte, dass Clique $Clq_4$ keine Knoten zugeordnet sind.):

$$T_{\psi(Clq_4)} = T_{Clq_1 \triangleright Clq_4}\, T_{Clq_2 \triangleright Clq_4}\, T_{Clq_3 \triangleright Clq_4}\, T_{Clq_5 \triangleright Clq_4} =$$

$$= \left[\begin{array}{cc} c_1 & c_2 \\ \hline 1 & 1 \end{array}\right] \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline g_1 & 1 & 1 \\ g_2 & 1 & 1 \end{array}\right] \left[\begin{array}{c|cc} & e_1 & e_2 \\ \hline g_1 & 0.0315 & 0.4185 \\ g_2 & 0.0235 & 0.5265 \end{array}\right] \left[\begin{array}{c|cc} & c_1 & c_2 \\ \hline e_1 & 1 & 0 \\ e_2 & 0.0104 & 0.9896 \end{array}\right] =$$

$$= \left[\begin{array}{cc|cc} & & c_1 & c_2 \\ \hline e_1 & g_1 & 0.0315 & 0 \\ & g_2 & 0.0235 & 0 \\ e_2 & g_1 & 0.0043524 & 0.4141476 \\ & g_2 & 0.0054756 & 0.5210244 \end{array}\right]$$

**Auffrischung.** Es liegen nun die folgenden Beobachtungen vor: Der Patient hat vor kurzem Asien besucht aber sein Röntgenbefund ist negativ. Als *findings* ergeben sich die folgenden Tabellen $\lambda_A = \left[\begin{array}{cc} a_1 & a_2 \\ \hline 1 & 0 \end{array}\right]$ und $\lambda_D = \left[\begin{array}{cc} d_1 & d_2 \\ \hline 0 & 1 \end{array}\right]$, die den $\psi$-Tabellen der Cliquen $Clq_6$ und $Clq_1$ hinzu multipliziert werden. Die Cliquen des Junction Trees sind jetzt nicht mehr zueinander kalibriert. Es erfolgt eine *Auffrischung* auf dem Junction Tree. Collect Evidence sieht nun wie folgt aus (siehe auch Abbildung B.5(a)):

1. Die Cliquen $Clq_1$ schickt ihre Nachricht an die Clique $Clq_4$. Von den Cliquen $Clq_2$ und $Clq_3$ wird keine Nachricht an Clique $Clq_4$ geschickt, da der Junction Tree vor dem Eintragen der Evidenz schon kalibriert war.

2. Jetzt schickt Clique $Clq_4$ ihre Nachricht an Clique $Clq_5$.

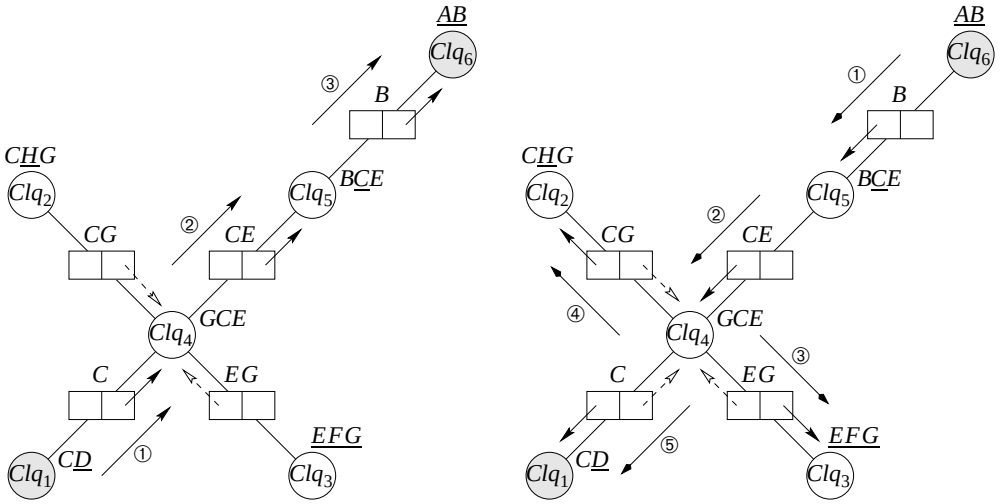
3. Darauf schickt Clique $Clq_5$ ihre Nachricht an Clique $Clq_6$ und Collect Evidence terminiert.

Daraufhin erfolgt Distribute Evidence (siehe auch Abbildung B.5(b)):

1. Die Clique $Clq_6$ schickt nun ihrerseits ihre Nachricht an Clique $Clq_5$.

2. Die Clique $Clq_5$ schickt ihre Nachricht an Clique $Clq_4$.

3. Zum Abschluss verschickt Clique $Clq_4$ ihre Nachrichten an die Cliquen $Clq_1$, $Clq_2$ und $Clq_3$. Distribute Evidence terminiert.




(a) Collect Evidence (b) Distribute Evidence

Abbildung B.5: Shafer-Shenoy: Auffrischung eines Junction Trees nach der Eingabe von neuen Beobachtungen (durch grau schraffierte Kreise gekennzeichnet)

Der Junction Tree ist nun wieder kalibriert.

Die BEL-Werte für die einzelnen Knoten des Bayesschen Netzes können aus jeder $\psi$-Tabelle, in der sich der Knoten befindet, durch Marginalisation bestimmt werden. Für den Knoten $C$ wird dies hier exemplarisch gemacht:

$$\begin{aligned}
\mathrm{BEL}(C) &= \alpha \sum_{Clq_1 \setminus \{C\}} T_{\psi(Clq_1)} = \alpha \sum_{Clq_1 \setminus \{C\}} T_{1(CD)}\ \theta_{(D)}\ T_{Clq_4 \triangleright Clq_1}\ \lambda_D \\
&= \alpha \sum_{Clq_1 \setminus \{C\}} \left[\begin{array}{c|cc} & d_1 & d_2 \\ \hline c_1 & 0.98 & 0.02 \\ c_2 & 0.05 & 0.95 \end{array}\right] \left[\begin{array}{cc} c_1 & c_2 \\ \hline 0.0010225 & 0.0089775 \end{array}\right] \left[\begin{array}{cc} d_1 & d_2 \\ \hline 0 & 1 \end{array}\right] \\
&= \alpha \sum_{Clq_1 \setminus \{C\}} \left[\begin{array}{c|cc} & d_1 & d_2 \\ \hline c_1 & 0 & 2.045\text{e-}5 \\ c_2 & 0 & 0.008528625 \end{array}\right] = \begin{pmatrix} 0.0024 \\ 0.9976 \end{pmatrix}
\end{aligned}$$

**Bsp**

## B.3 D-Separations-Kriterium

Aus Gründen der Übersichtlichkeit ist an dieser Stelle das D-Separations-Kriterium, wie es schon im Abschnitt 3.4.1 definiert ist, noch einmal aufgeführt.

***Definition B.3.1 (D-Separations-Kriterium I)***
Seien $\mathbb{X}$, $\mathbb{Y}$ und $\mathbb{Z}$ drei disjunkte Teilmengen von Knoten des gerichteten azyklischen Graphen $\mathscr{G}$ des Bayesschen Netzes $\mathscr{BN}$. Dabei muss die Vereinigung von $\mathbb{X}$, $\mathbb{Y}$ und $\mathbb{Z}$ nicht die gesamte Knotenmenge des Graphen $\mathscr{G}$ ergeben. Dann *d-separiert* die Menge $\mathbb{Z}$ die beiden Mengen $\mathbb{X}$ und $\mathbb{Y}$ voneinander (Schreibweise: $< \mathbb{X} \mid \mathbb{Z} \mid \mathbb{Y} >_{\mathscr{G}}$), wenn es entlang jeden Pfades zwischen einem Knoten aus $\mathbb{X}$ und einem Knoten aus $\mathbb{Y}$ ein Knoten *w* existiert, der eine der beiden nachfolgenden Eigenschaften erfüllt:

1. *w* hat Vorgängerknoten und weder *w* noch seine Nachfolgerknoten sind in $\mathbb{Z}$, oder

2. *w* hat keine Vorgängerknoten, und *w* ist in $\mathbb{Z}$.

D

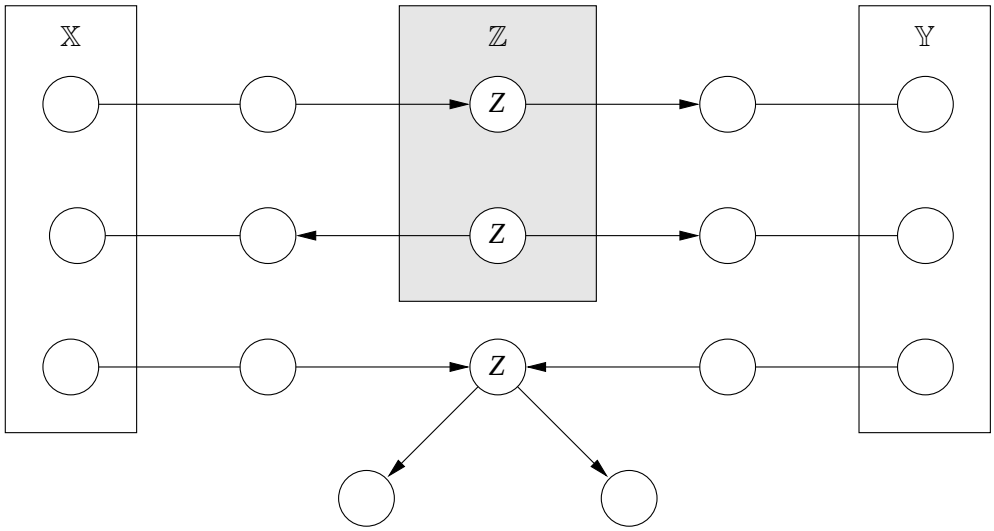



Abbildung B.6: D-Separations-Kriterium: Drei Arten wie ein Pfad von $\mathbb{X}$ nach $\mathbb{Y}$ durch die Menge $\mathbb{Z}$ blockiert werden kann. Wird jeder Pfad von $\mathbb{X}$ nach $\mathbb{Y}$ blockiert, so sagt man, dass die Menge $\mathbb{Z}$ die beiden Menge $\mathbb{X}$ und $\mathbb{Y}$ voneinander d-separiert (entnommen aus [Russell & Norvig 03]).

Die folgende Definition besagt dasselbe bzgl. des D-Separations-Kriteriums wie die obige Definition. Beide Definitionen zusammen tragen jedoch gut zum Verständnis des D-Separations-Kriteriums bei.

***Definition B.3.2 (D-Separations-Kriterium II)***
Eine Menge $\mathbb{Z}$ von Knoten *d-separiert* zwei Mengen von Knoten $\mathbb{X}$ und $\mathbb{Y}$, wenn jeder ungerichtete Pfad von einem Knoten in $\mathbb{X}$ zu einem Knoten aus $\mathbb{Y}$ durch $\mathbb{Z}$

*blockiert* wird. Ein Pfad wird durch eine Menge $\mathbb{Z}$ von Knoten blockiert, wenn es einen Knoten $Z$ auf dem Pfad gibt, für den eine der drei folgenden Bedingungen gilt:

1. $Z \in \mathbb{Z}$ und $Z$ hat auf dem Pfad eine eingehende Kante und eine austretende Kante.
2. $Z \in \mathbb{Z}$ und beide Kanten von $Z$ auf dem Pfad zeigen von $Z$ weg.
3. Weder $Z$ noch einer seiner Nachfolgerknoten sind in der Knotenmenge $\mathbb{Z}$, und beide Kanten auf dem Pfad zeigen zu $Z$ hin.

**D**

Die obige Definition ist in Abbildung B.6 grafisch veranschaulicht.
Abschließend jetzt noch zwei Beispiele bzgl. des D-Separations-Kriteriums.
Das folgende Beispiel ist aus [Pearl 88] entnommen.

***Beispiel B.3.3 (blockieren, aktivieren)***
In Abbildung B.7 d-separiert die Menge $\mathbb{Z} = \{A\}$ die beiden Mengen $\mathbb{X} = \{B\}$ und $\mathbb{Y} = \{C\}$ voneinander, denn der Knoten $A \in \mathbb{Z}$ blockiert den Pfad $B \leftarrow A \rightarrow C$, und der Pfad $B \rightarrow D \leftarrow C$ wird blockiert, da der Knoten $D$ und seine Nachfolger nicht in der Menge $\mathbb{Z}$ enthalten sind.

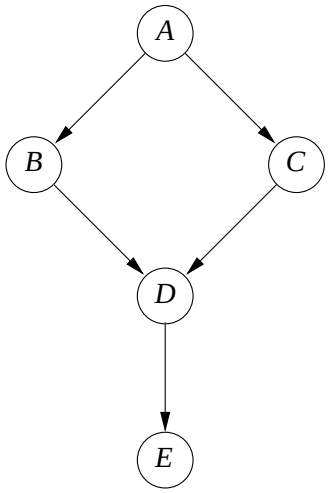



Abbildung B.7: D-Separations-Kriterium: Verdeutlichung zur Definition von *blockieren* und *aktivieren*

Andererseits separiert die Menge $\mathbb{Z}' = \{A, E\}$ nicht die beiden anderen Mengen $\mathbb{X} = \{B\}$ und $\mathbb{Y} = \{C\}$, denn der Pfad $B \rightarrow D \leftarrow C$ wird durch den Knoten $E$ aktiviert.

**Bsp**

Das nun folgende Beispiel stammt aus [Russell & Norvig 03].

***Beispiel B.3.4 (D-Separations-Kriterium)***
Sei das Bayessche Netz in Abbildung B.8 gegeben. Es ist ein einfaches Bayessches Netz zur Darstellung des Sachverhaltes "Auto", wobei nur die Zusammenhänge zwischen Batterie, Radio, Zündung, Benzin, Starten und Fahren betrachtet werden.

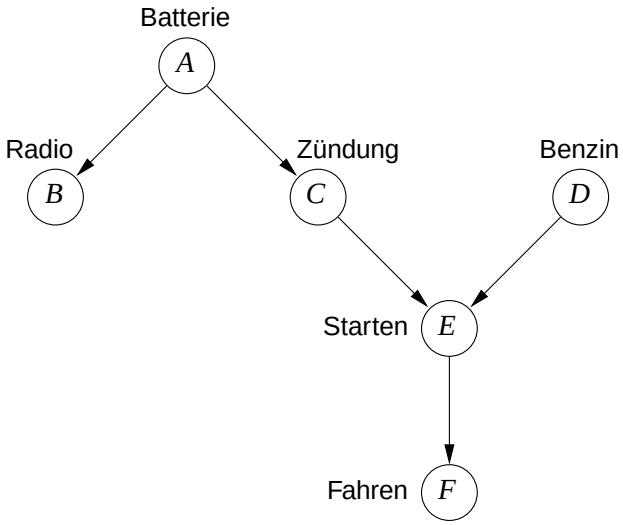



Abbildung B.8: D-Separations-Kriterium: Beispielnetz Auto

Aus dem Graphen lassen sich mit dem D-Separations-Kriterium z. B. die folgenden Abhängigkeiten folgern:

1. Benzin und Radio sind unabhängig voneinander, wenn Batterie bekannt ist.

2. Benzin und Radio sind unabhängig voneinander, wenn gar keine Evidenz vorliegt.

3. Benzin und Radio sind abhängig voneinander, wenn Evidenz über Starten vorliegt. Lässt sich das Auto nicht starten, dann ist das spielende Radio ein Indiz dafür, dass die Batterie in Ordnung ist und das Benzin alle sein muss.

4. Benzin und Radio sind abhängig voneinander, wenn Evidenz über Fahren vorliegt. Fahren ist ein Nachfolger von Starten.

**Bsp**

## B.4 Algorithmen für Bayessche Netze

### B.4.1 Minimal aufspannender Baum

Wie wir in Abschnitt 2 gesehen haben, kann auf dem Bayesschen Netz nicht direkt gerechnet werden. Es muss vielmehr eine neue Struktur bestimmt werden, der sogenannte Junction Tree, auf dem gerechnet werden kann. Dieser Junction Tree wird mit dem nachfolgenden Algorithmus aus einem Junction Graphen des Bayesschen Netzes bestimmt. Ein mehrfach verknüpfter Graph wird dabei in einen Baum umgewandelt, wobei die Gesamtheit der verbleibenden Kanten die Junction Tree Eigenschaft erfüllt.

MST-Kruskal($\mathscr{J}$,$\gamma$)
(1) ordne jedem Knoten $V$ des Graphen $\mathscr{J}$ einen eigenen Baum $\mathscr{B}(V) = (\{V\},\emptyset)$ zu
(2) sortiere alle Kanten $E = \{U,V\} \in \mathbb{E}_\gamma$ des Graphen $\mathscr{J}$ absteigend bzgl. $\gamma$
(3) **for** jede Kante $E = \{U,V\}$ des Graphen $\mathscr{J}$ nach Sortierung **do**
(4) **if** $\mathscr{B}(U) \neq \mathscr{B}(V)$ **then**
(5) verbinde die beiden Bäume $\mathscr{B}(U) = (\mathbb{N}_U,\mathbb{E}_U)$ und $\mathscr{B}(V) = (\mathbb{N}_V,\mathbb{E}_V)$ durch die Kante $E$ zu einem Baum $\mathscr{B} = (\mathbb{N}_U \cup \mathbb{N}_V, \mathbb{E}_U \cup \mathbb{E}_V \cup E)$
(6) **fi**
(7) **od**
(8) **return** $\mathscr{B}$

Es wird der Junction Graph $\mathscr{J}$ mit der Gewichtung $\gamma$ übergeben. In Programmzeile (1) wird jedem Knoten $V$ des Junction Graphen $\mathscr{J}$ ein eigener Baum $\mathscr{B}(V)$ zugeordnet. Im Anfang enthält dieser Baum nur den jeweiligen Knoten $V$ und noch keine Kanten. In Zeile (2) werden die Kanten des Graphen $\mathscr{J}$ bzgl. ihres Gewichtes $\gamma$ absteigend sortiert. In den Programmzeilen (3) bis (7) wird der minimal aufspannende Graph bestimmt. In Zeile (8) wird der minimal aufspannende Graph $\mathscr{B}$ als Ergebnis zurückgegeben.

### B.4.2 Minimale Triangulierung

In [Kjærulff 93] wird der folgende Algorithmus vorgeschlagen, um redundante Triangulationskanten einer nichtminimalen Triangulation $\mathbb{T}$ eines Graphen $\mathscr{G}$ zu entfernen. Die Komplexität liegt bei $O(c^2\,|\,\mathbb{T}\,|^2)$. Der Algorithmus wird mit minT($\mathbb{T}$,$\mathscr{G} = (\mathbb{V},\mathbb{E} \cup \mathbb{T})$,$\mathbb{T}$) aufgerufen.

```
minT(𝕋,𝒢 = (𝕍, 𝔼 ∪ 𝕋),ℝ)
(1)  ℝ′ ← {E1 ∈ 𝕋 | ∃E2 ∈ ℝ : E1 ∩ E2 ≠ ∅}
(2)  𝕋′ ← {{V,W} ∈ ℝ′ |
           Graph(adj(V,𝒢) ∩ adj(W,𝒢)) ist zusammenhängend}
(3)  if 𝕋′ ≠ ∅ then
(4)      return (minT(𝕋 \ 𝕋′, 𝒢 = (𝕍, 𝔼 ∪ 𝕋 \ 𝕋′), 𝕋′))
(5)  else
(6)      return (𝕋)
(7)  fi
```

In Programmzeile (1) wird die Menge $\mathbb{R}'$ bestimmt, d. h. die Menge der Kanten aus $\mathbb{T}$, die einen Knoten gemeinsam haben mit einer Kante aus $\mathbb{R}$. Im Anfang ist $\mathbb{R} = \mathbb{T}$. Um $\mathbb{R}'$ zu bestimmen, gehe alle Kanten aus $\mathbb{T}$ durch und überprüfe, ob eine Kante aus $\mathbb{R}$ wenigstens einen Knoten mit ihr gemeinsam hat. Ist dies der Fall, so gehört die Kante zu $\mathbb{R}'$. Im nächsten Schritt wird $\mathbb{T}'$ bestimmt. Das ist die Menge der Kanten aus $\mathbb{R}'$, die entfernt werden kann, ohne dass der Graph nicht mehr trianguliert ist. Um $\mathbb{T}'$ zu bestimmen, müssen alle ungerichteten Kanten $\{V,W\} \in \mathbb{R}'$ darauf überprüft werden, ob die Schnittmenge $\text{adj}(V,\mathscr{G}) \cap \text{adj}(W,\mathscr{G})$ einen zusammenhängenden Graphen induziert.

# Anhang C

# Mobile Erfassung und Auswertung physiologischer Daten

## C.1 Varioport

Das Varioport ist ein kompaktes und leichtes Messsystem zur Erfassung und Aufzeichnung von psychophysiologischen oder auch Umweltdaten über einen Zeitraum von bis zu einer Woche. Die Besonderheit des Gerätes besteht darin, dass für die Analogkanäle lediglich Endverstärker integriert sind, die Vorverstärker sich jedoch in den einzelnen Messkabeln befinden. Dies hat neben der besseren Signalqualität durch die nahe bei den Elektroden sitzenden Vorverstärker den Vorteil,

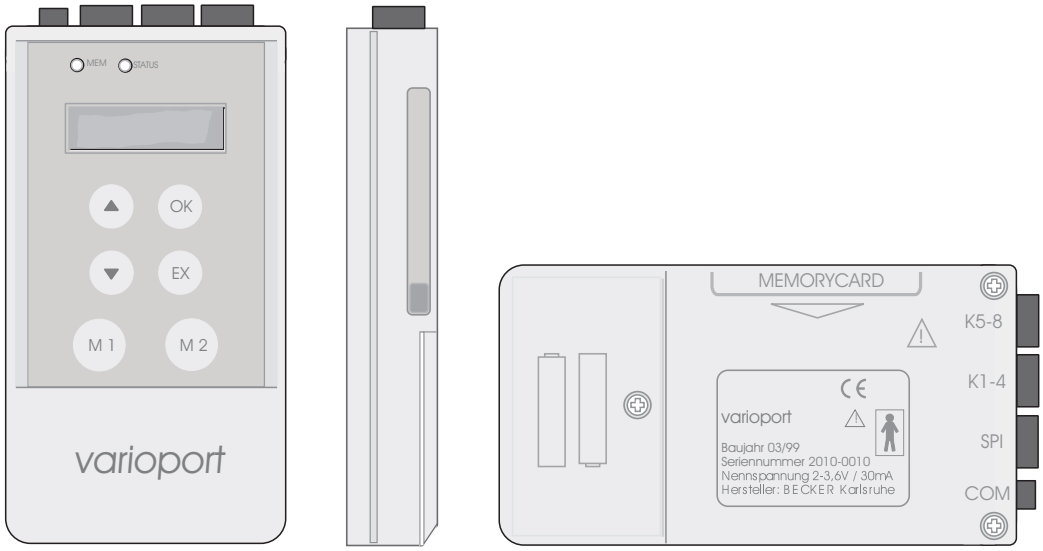



Abbildung C.1: Ansichten vom Varioport.

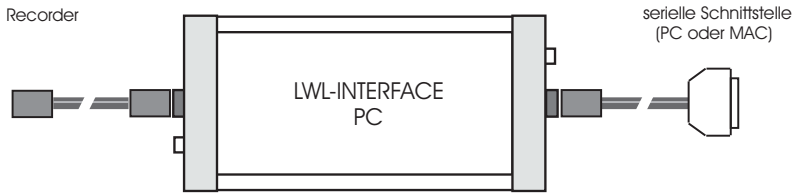



Abbildung C.2: LWL-Interface

dass die Belegung der Kanäle lediglich durch das angeschlossene Kabel bestimmt wird, also frei wählbar ist. Dadurch ist das Varioport bei wechselnden Versuchsbedingungen sehr flexibel einsetzbar. In Abbildung C.1 ist das Varioport in verschiedenen Ansichten zu sehen.

An der oberen Stirnseite des Gerätes befinden sich die folgenden Anschlussstecker:

**COM (vierpolige Buchse)** Über diese serielle Schnittstelle mit Übertragungsraten bis zu 230 KBaud wird das Gerät über das mitgelieferte Glasfaser- bzw. Lichtwellenleiter-Interface (siehe Abbildung C.2) mit einem PC, Laptop oder PDA verbunden.

Es enthält eine interne Übertragungsstrecke aus Lichtwellenleitern und sorgt dafür, dass ein am Rekorder verkabelter Patient auf keinen Fall in galvanischen Kontakt mit netzführenden Geräten kommt. Der Datenverkehr wird durch zwei LEDs an den Stirnseiten des Gehäuses angezeigt. Die Stromversorgung der 'Primärseite' erfolgt durch den Rekorder; die 'Sekundärseite' versorgt sich aus dem PC (Hier kommt es zu Problemen, wenn das Gerät mit einem Handheld-Gerät verbunden wird, das diese Stromversorgung nicht liefert. Zur Behebung des Problems muss ein Kabel verwendet werden, das keine Stromversorgung benötigt.).

**SPI (siebenpolige Buchse)** Dies ist eine serielle Hochgeschwindigkeits-Schnittstelle, über die zusätzliche Peripheriegeräte oder Erweiterungseinheiten an das Varioport angeschlossen werden können (z. B. eine hochauflösende EDA-Messeinheit (siehe Abbildung C.3) oder EEG/EMG-Module mit bis zu 16 Kanälen).

**Messeingänge (zwei achtpolige Buchsen für die Kanäle 1–4 bzw. 5–8 plus Marker)** Hier können über die im Lieferumfang enthaltenen Verteilerstecker die acht Vorverstärker, eine externe Markertaste, sowie ein zentrales Anschlusskabel für die Patientenmasse angeschlossen werden.

Als Biosensoren stehen beispielsweise zur Verfügung:

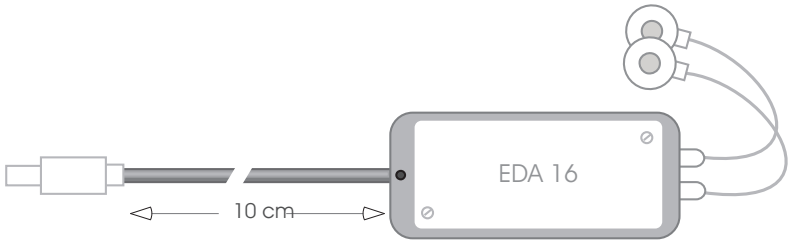



Abbildung C.3: EDA-Sensor

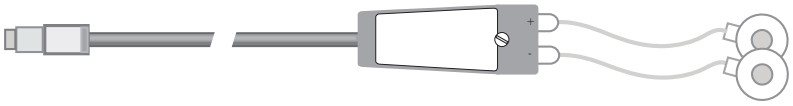


Abbildung C.4: EMG-Sensor

- Elektromyogramm (EMG) für die Erfassung von Muskelanspannung (siehe Abbildung C.4)
- Elektrookulogramm (EOG) für die Erfassung von Augenbewegungen und Blinzeln
- Elektrokardiogramm (EKG) für die Erfassung der Herzschlagrate (HR) und der Interbeat-Intervalle (IBIs)
- Elektrodermale Aktivität (EDA) für die Erfassung von Spontanfluktuationen und der durchschnittlichen Hautleitfähigkeit (siehe Abbildung C.3)
- Atemgurt für die Erfassung der Atemtätigkeit oder um festzustellen, ob jemand spricht (siehe Abbildung C.5)

Verfügbare Umgebungssensoren sind zum Beispiel:

- Beschleunigungssensor (ACC) für die drei-dimensionale Erfassung von Beschleunigungen

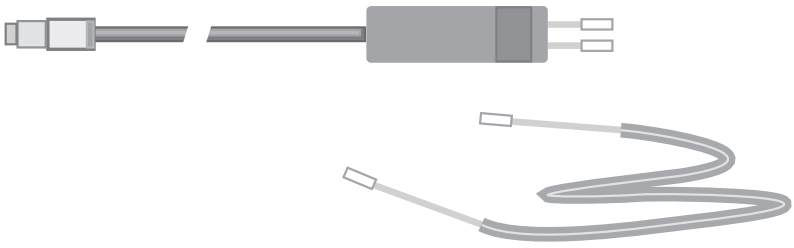


Abbildung C.5: Atemgurt

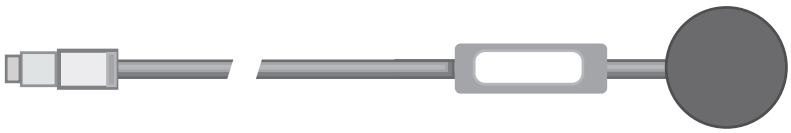


Abbildung C.6: Piezo-Sensor

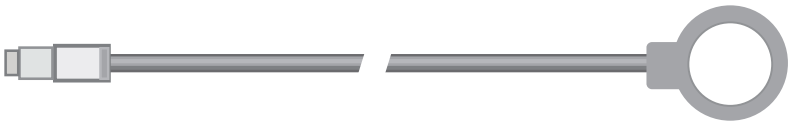


Abbildung C.7: (Körper-) Temperatur

- Bewegungssensor (siehe Abbildung C.6) für das Feststellen von Bewegungen
- Temperatursensor (siehe Abbildung C.7) für die Erfassung der direkten Umgebungstemperatur. Dieser Sensor kann aber auch als Biosensor angesehen werden, wenn mit ihm die Körpertemperatur gemessen wird.

Auf dem Gerät selbst findet schon eine Vorverarbeitung der einkommenden Daten statt (Filtern, Vorverstärkung, Digitalisierung) wobei die Erfassungs- und die Speicher-Frequenz auf dem Gerät einstellbar ist. Zusätzlich kann der eingebaute 32 Bit-Prozessor (siehe Abbildung C.8) spezielle Vorverarbeitungen auf einzelnen Sensorkanälen durchführen, z. B. kann er die Herzschlagrate aus den unverarbeiteten EKG-Daten bestimmen (siehe [Jain et al. 03]). Diese Möglichkeiten werden allerdings in der aktuellen Javario-Version noch nicht genutzt.

Über die serielle Schnittstelle 'COM' werden die Daten vom Gerät ausgelesen und auch das Gerät kontrolliert: Dazu wird eine Definitionsdatei auf den Varioport übertragen, die die Aufnahme konfiguriert und kontrolliert; die Messungen können durch Befehle gestartet und gestoppt werden, und die aktuellen Werte der Sensoren können online überwacht werden.

Es sind Scan- und Speicherraten von bis zu 512 Hz möglich. Die gemessenen Daten können sowohl auf einer lokalen CF-Speicherkarte gespeichert als auch gleichzeitig über die serielle Schnittstelle beispielsweise an einen PDA oder Laptop geschickt werden. Momentan schränkt jedoch die Bandbreite der seriellen Schnittstelle die Online-Übertragung der Sensordaten an andere Geräte ein, so dass nur drei Sensoren gleichzeitig ausgelesen werden können.

Um die Daten weiter auswerten zu können, die vom Varioport über die serielle Schnittstelle einströmen, muss auf dem angeschlossenen Gerät eine entsprechende Schnittstellensoftware für den Empfang und die Auswertung der Daten laufen. Dazu wird die Javario API entwickelt, die im folgenden kurz vorgestellt wird.

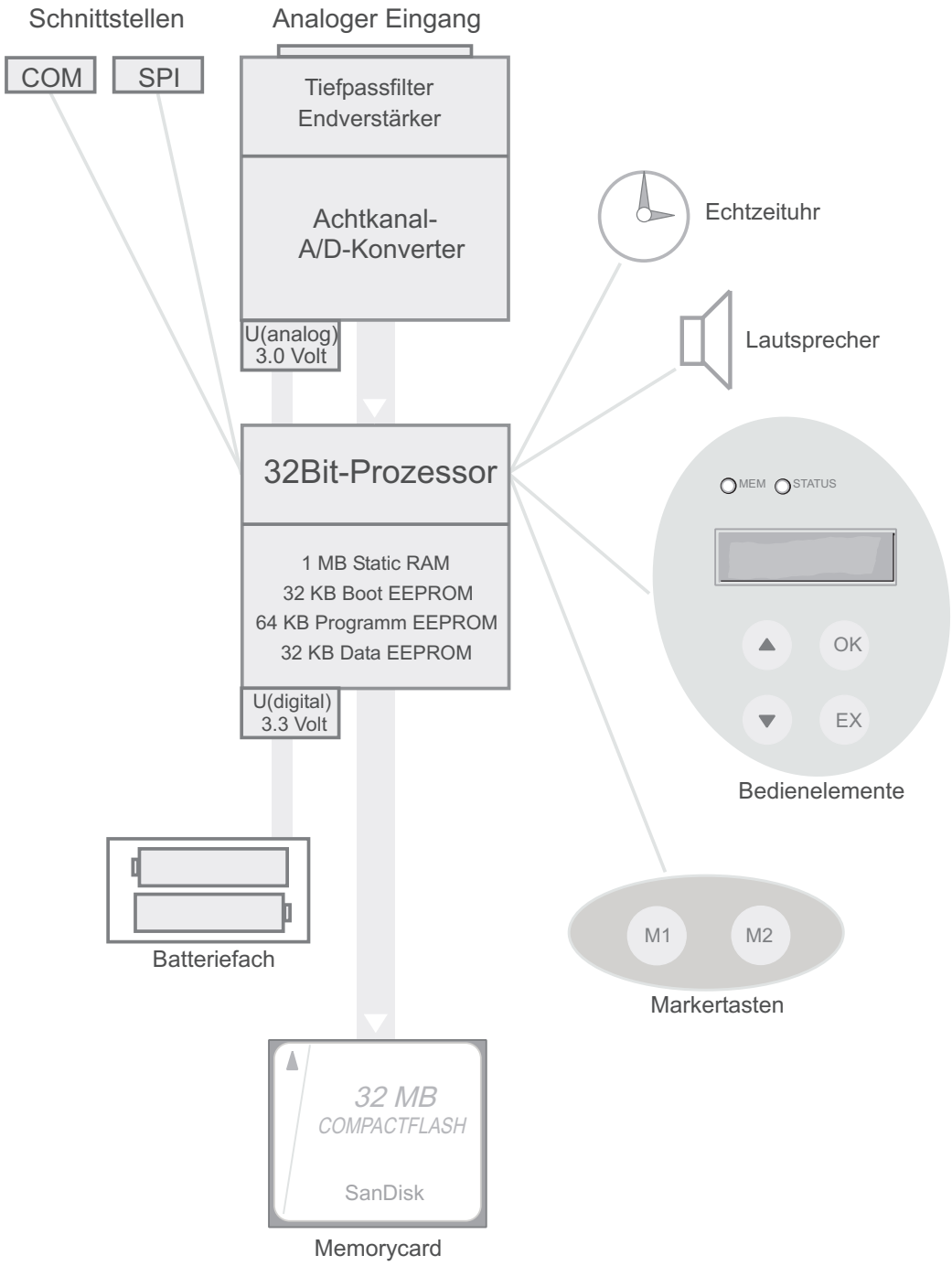



Abbildung C.8: Blockschaltbild des Varioports.

## C.2 Javario API

Die Javario-Bibliothek stellt für die Verwaltung und das Auslesen der Sensoren des Varioports mehrere Funktionen zur Verfügung. Sie wurde in Java entwickelt, um eine plattformübergreifende Schnittstelle zur Verfügung zu stellen, die einfach in unsere bestehende Architektur eingebunden werden kann. Ebenso sollte die Software auf verschiedenen Handheld-Geräten lauffähig sein, für die schon eine virtuelle Maschine vorhanden ist. Javario muss sich einerseits an die eingeschränkten Ressourcen des Gerätes und andererseits an die Anforderungen der speziellen Messung anpassen.

Von Javario werden als erster Vorverarbeitungsschritt für jeden Sensor Aktivierungsschwellwerte berechnet. Die Messungen des Beschleunigungssensors werden weitergehend interpretiert, um die horizontale Lage des Sensors zu bestimmen. Schritte werden durch die Anwendung von gewissen Schwellwerten erkannt, so dass die Gehgeschwindigkeit des Benutzers aufgrund der Frequenz der erkannten Schritte in langsam, normal bzw. schnell eingeordnet werden kann.

Ähnlich dem Programm JavaDBN könnte Javario Quellcode generieren, der auf eingebetteten Systemen verarbeitet werden kann und die entsprechenden Sensoren ausliest.

## C.3 Anbringung der Biosensoren

Zur Vorbereitung der entsprechenden Hautpartien ist je nach Sensor, der angebracht werden soll, unterschiedlich vorzugehen.

Beim *EKG-*, *EMG-* bzw. *EOG-Sensor* muss mit Alkohol und einer abrasiven Reinigungspaste (wie z. B. Skin Pure) die Haut vorbereitet werden. Zuerst reinigt man die entsprechende Stelle mit Alkohol und anschließend präpariert man die Stelle mit der abrasiven Reinigungspaste, indem man eine tropfengroße Menge an Paste auf der Spitze eines Wattestäbchens aufbringt und dieses mit leichtem Druck auf der Haut, wo die Elektroden appliziert werden sollen, ca. fünf Sekunden hin- und herdreht. Die abrasive Paste, die sich dann dort noch befindet, wird nicht abgewischt, sondern die Elektroden können für die Ableitung sofort aufgeklebt werden. Je nach Anfertigung werden Einmal- oder Dauerelektroden verwendet. Die Dauerelektroden müssen noch mit einem Klebering und Elektrodenpaste versehen werden, bevor sie aufgeklebt werden können. Die Einmalelektroden bedürfen hingegen keiner weiteren Vorbereitung; sie bestehen aus einem mit Elektrodenpaste getränkten Vlies, das selbstklebend ist.

Der Hautwiderstand, der vom *EDA-Sensor* gemessen wird, ist eine exosomatische Größe, die nur unter Zufuhr von Energie bzw. Spannung ableitbar ist. Beim *EDA-Sensor* darf kein Alkohol und keine Sandpaste verwendet werden; die Haut

Abbildung C.9: EDA-Sensor an Handinnenfläche mit Zugentlastungsschlaufe

ist lediglich mit Wasser zu reinigen. In unserer Anfertigung wird eine Dauerelektrode verwendet, die noch mit einem Klebering und einer speziellen EDA-Paste, die 0,5% NaCl (Natriumchlorid oder auch als Kochsalz bekannt) enthält, versehen werden muss.

Bei der Datenerfassung werden oft auch Signale aufgefangen, die einen anderen Ursprung haben als die Daten, die vom Sensor erfasst werden sollen. Diese Störsignale werden *Artefakte* genannt. Man unterscheidet dabei (a) Artefakte *physiologischer Herkunft* wie andere Biosignale (wie Muskelaktivität, die EEG-Messungen überlagert) oder Bewegungsartefakte (wie Widerstandsänderungen aufgrund von sich bewegenden Elektroden und Elektrodenkabeln) und (b) Artefakte durch *elektrische Einstreuungen*. Andere Biosignale werden beispielsweise bereits durch entsprechende Filtereinstellungen bei der Aufzeichnung unterdrückt. Damit möglichst wenig Bewegungsartefakte auftreten, werden die Dauerelektroden zusätzlich noch mit Tesaband auf der Haut fixiert, und bei allen Sensoren wird eine *Zugentlastungsschlaufe* angelegt, so dass bei Spannung an den Kabeln nicht direkt an der Elektrode gezogen wird und die Messungen dadurch gestört werden. In Abbildung C.9 ist zu sehen, wie ein EDA-Sensor an der Handinnenfläche angebracht wurde. Deutlich ist links unterhalb der Armbanduhr die Zugentlastungsschlaufe zu sehen. Die Dauerelektroden wurden zusätzlich noch mit Tesaband fixiert.

Je nach Sensortyp gibt es verschiedene Ableitungsorte zu beachten. Der EDA-Sensor kann beispielsweise an der Handinnenfläche oder unter der Fußsohle angebracht werden. Je nach Anwendung ist einmal die Handinnenfläche und ein anderes mal die Fußsohle besser geeignet.

Beim EKG-Sensor gibt es beispielsweise die Einthoven-Ableitung I, bei der die Elektroden am rechten und am linken Arm angebracht werden, die Einthoven-Ableitung II (rechter Arm, linkes Bein) oder die Einthoven-Ableitung III (linker Arm, linkes Bein), wobei über eine Elektrode am rechten Bein geerdet wird. Diese Ableitungen sind jedoch im mobilen Szenario wenig geeignet, da die Kabel, mit denen die Elektroden verbunden werden müssen, bei der Bewegung durch ihre Länge sehr hinderlich sein können. Ebenso gestaltet sich eine bequeme Kabelführung über die Gelenke sehr schwierig. Wir verwenden daher in unserem mobilen Szenario die Brustwandableitung (Manubrium sterni, Schwertfortsatz), wobei hier über eine Elektrode an der linken untersten Rippe geerdet wird. Dabei sind die Kabellängen relativ kurz, und es müssen keine Gliedmaßen verkabelt werden. Sowohl bei der Einthoven-Ableitung II. als auch der Brustwandableitung, die wir verwenden, erhalten wir ausgeprägte R-Zacken, die für die Erkennung des Herzschlages notwendig sind.

## C.4 Abbildung von Sensoren in dynamische Bayessche Netze

In [Brandherm 01] werden erste Überlegungen vorgestellt, wie sich aufbereitete Daten aus Bio- und Umgebungssensoren zur Bewertung der Ressourcenlage einer Person (Zeitdruck und Arbeitsgedächtniskapazität) einsetzen lassen. Mittels einer Literaturrecherche (siehe u.a. [Healey & Picard 00], [Healey 00], [Rowe et al. 98], [Picard 97] und [Schandry 98]) wurden einige Bio- und Umgebungsdaten selektiert, die erfolgversprechend für die Erkennung von Stress oder Zeitdruck eines Benutzers erscheinen, und die Struktur eines Bayessches Netz für diese Daten modelliert, das für weitere Bayessche Netze als Ausgangsnetz bzw. Inspiration dienen sollte wie z. B. für das dynamische Bayessche Netz des Alarm Managers (siehe Abbildung 7.9 in Abschnitt 7.2).

Im folgenden Abschnitt beschreiben wir nun die ausgewählten Daten und ihre kausalen Zusammenhänge, die als gerichteter azyklischer Graph in Abbildung C.10 modelliert werden. Durchgezogene Kanten beschreiben eine positive kausale Beeinflussung, wohingegen gestrichelte Kanten eine negative kausale Beeinflussung darstellen.

### C.4.1 Biosensoren

Mit dem Elektrokardiogrammsensor lassen sich verschiedene Merkmale der Herztätigkeit wie z. B. Herzschlagfrequenz (HR)[1] oder Herzfrequenzvariabilität (HRV)[2] (siehe die beiden Knoten Herzschlagfrequenz und Herzfrequenzvariabilität in Abbildung C.10) bestimmen. Diese Merkmale werden durch innere und äußere Faktoren beeinflusst. Wir betrachten hier keine Größen, die als statisch angesehen werden können wie z. B. Alter oder Trainiertheit der Person. Wir nehmen an, dass durch den Sensor die entsprechende Bewertung erfolgt, d. h. wenn die Herzschlagfrequenz einer Person durch ihr hohes Alter und ihre Untrainiertheit grundsätzlich sehr hoch ist, so wird diese Herzschlagfrequenz für sie als normal angesehen und die Hypothese "normal" im Knoten Herzschlagfrequenz instantiiert. Eine erhöhte emotionale oder kognitive Belastung sowie verstärkte körperliche Tätigkeit (siehe die Knoten emotionale Belastung, kognitive Belastung und körperliche Tätigkeit) lassen i. a. die Herzschlagfrequenz ansteigen und die Herzfrequenzvariabilität abnehmen. Wurde auf den Benutzer beruhigend eingewirkt oder ist er erholt, so ist bei ihm eine verringerte Herzschlagfrequenz und eine erhöhte Herzfrequenzvariabilität feststellbar. Nach [Schandry 98] sinkt die Herzfrequenzvariabilität mit steigender Herzschlagfrequenz. Plötzliche Geräusche (durch den Knoten äußere Reize repräsentiert) lassen die Herzschlagfrequenz ansteigen. Ein weiterer Einflussfaktor ist die Atmung (siehe den Knoten Atmung); beim Ein- bzw. Ausatmen wird die Herzschlagfrequenz beschleunigt bzw. verlangsamt (sogenannte Atmungsarrhythmie).

Mit einem Sensor für Atemtätigkeit (siehe Abbildung C.5) lassen sich Atemrhythmus ("schnell" oder "langsam") und -amplitude ("tief" oder "flach") eines Benutzers bestimmen und, ob er ein- oder ausatmet, spricht oder einen kurzen Atemaussetzer hatte. Eine schnelle und tiefe Atmung weist auf eine emotionale Belastung oder körperliche Tätigkeit des Benutzers hin. Atmet er jedoch langsam und flach, so ist dies ein Anzeichen für den entspannten und ausgeruhten Zustand des Benutzers. Plötzliche Stressoren (Mit Stressor bezeichnet man einen Stress bewirkenden Faktor.) können den Benutzer erschrecken und dabei zu kurzzeitigen Atemaussetzern führen. Der Knoten Atmung enthält beispielsweise die Hypothesen "langsame und flache Atmung", "schnelle und tiefe Atmung", "Sprechen", "Atemaussetzer", "unregelmäßige Atmung" und "sonstige". Der Knoten Ein-/Ausatmen enthält z. B. die Hypothesen "einatmen", "ausatmen" und "sonstige". Durch einen gepunkteten Strich zwischen beiden Knoten wird signalisiert, dass in diesem Modell vereinfachend angenommen wird, dass sie sich gegenseitig nicht beeinflussen.

Je mehr kurzfristige Schwankungen des Hautleitwertsniveau, d. h. sogenannte

[1] HR ist ein Akronym für den englischen Begriff *heart rate*.
[2] HRV ist ein Akronym für den englischen Begriff *heart rate variability*.

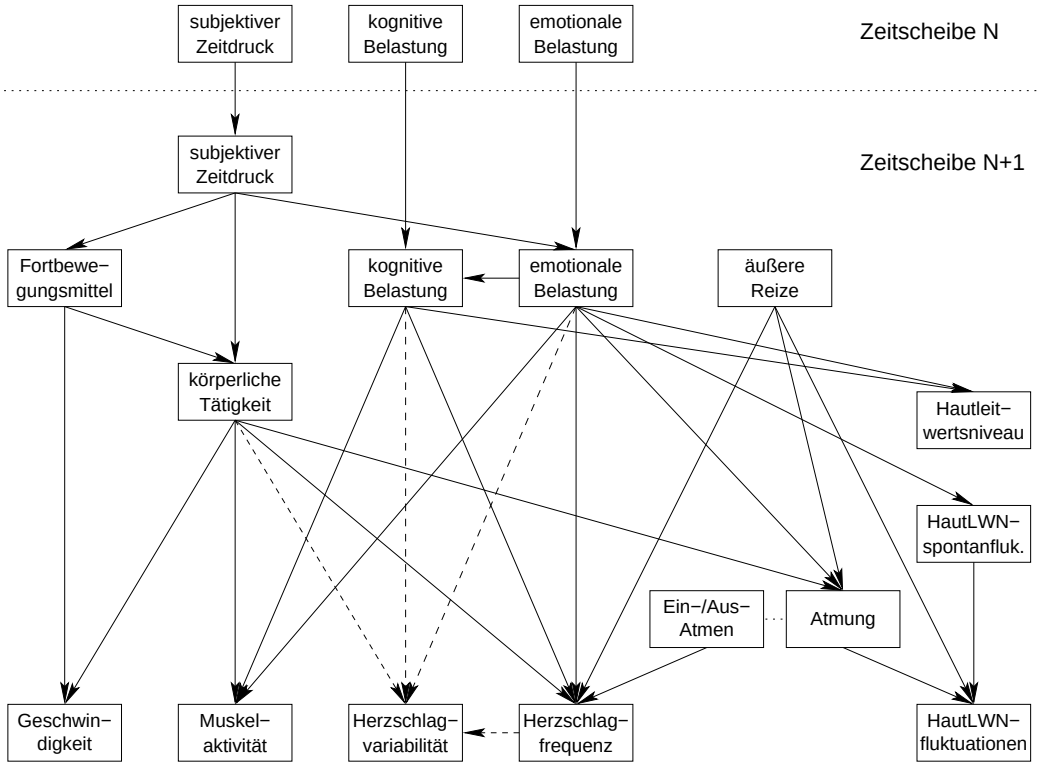



Abbildung C.10: Dynamisches Bayessches Netz für Biosensor- und subsumierte Umgebungsdaten.

spontane phasische Hautleitwertsniveaufluktuationen (siehe Knoten HautLWNspontanfluktuationen), in einer Zeiteinheit auftreten, desto größer ist die emotionale Belastung des Benutzers. Dabei dürfen die durch äußere Reize (Knoten äußere Reize) oder unregelmäßige Atmung (Atmung) wie Husten, Räuspern oder tiefes Atemholen ausgelösten Fluktuationen (HautLWNfluktuationen) nicht mitgezählt werden. Das Hautleitwertsniveau eines Benutzers (Knoten Hautleitwertsniveau) wird sowohl durch kognitive als auch emotionale Belastung erhöht.

Mit einem *Elektromyogrammsensor* lässt sich Muskeltätigkeit bzw. Muskelspannung (siehe Knoten Muskelaktivität) erfassen. Wenn ohne eine erkennbare entsprechende körperliche Tätigkeit die Muskeltäigkeit ansteigt, so deutet dies auf eine emotionale oder kognitive Belastung hin. Das Fortbewegungsmittel (die Person kann zu Fuß, per Laufband oder Auto/Zug unterwegs sein) und die körperliche Tätigkeit bestimmen die Geschwindigkeit, in der sich der Benutzer fortbewegt. Die Hypothesen für den Knoten körperliche Tätigkeit könnten beispielsweise “Stehen”, “Schlendern”, “Gehen”, “Laufen” und “Rennen” lauten. Die Wahl des Fortbewegungsmittel beeinflusst dabei den Umfang der ausführbaren körperlichen Tätigkeiten (so wird auf einer Rolltreppe die Person eher stehen oder gehen, aber

weniger laufen oder rennen). Die Geschwindigkeit ist durch einen Geschwindigkeitssensor messbar, und das Fortbewegungsmittel lässt sich mittels Umgebungssensoren bestimmen.

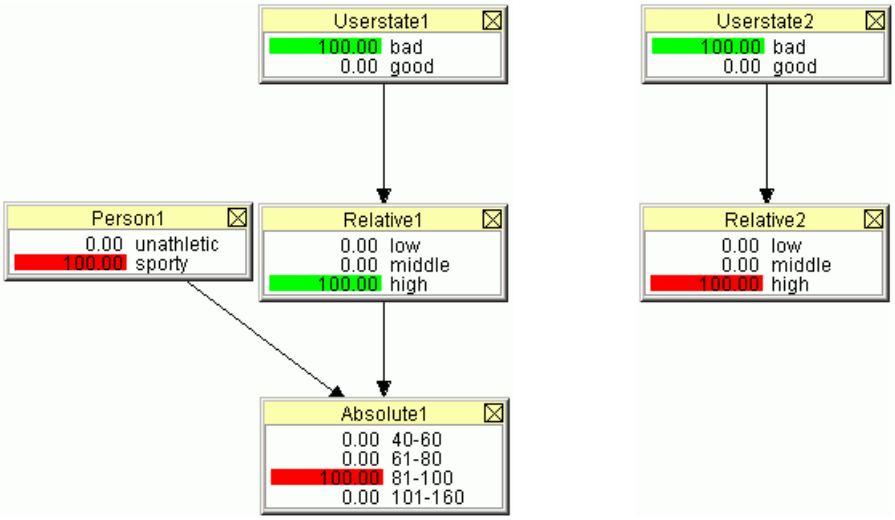
Bei erhöhtem subjektiven Zeitdruck wird die Person eher zu einer schnelleren Gangart übergehen, als wenn sie keinen Zeitdruck verspürt.

### C.4.2 Umgebungssensoren

Wie im vorhergehenden Abschnitt gezeigt, lassen einige Biodaten unter Hinzunahme von Umgebungsdaten aussagekräftigere Schlüsse zu. Beispielsweise liefern Audio- und Videosensoren Daten darüber, dass in Benutzernähe geredet wird. Werden diese mit Atemsensordaten ergänzt, lässt sich erkennen, ob der Benutzer am Gespräch teilnimmt. Einige der Daten, die sich aus den Umgebungssensoren bestimmen lassen, sind im Bayesschen Netz in Abbildung C.10 in den Knoten äußere Reize, Geschwindigkeit oder Fortbewegungsmittel subsumiert. Im Knoten äußere Reize sind z. B. die Hypothesen “Stressor” und “lautes Geräusch” enthalten.

### C.4.3 Zusammenfassung und Ausblick

Das dynamische Bayessche Netz in Abbildung C.10 diente u. a. dem dynamischen Bayesschen Netz des Alarm Managers (siehe Abbildung 7.9) als Vorlage. Im dynamischen Bayesschen Netz des Alarm Managers sind im Gegensatz zu diesem dynamischen Bayesschen Netz die Sensorknoten mit ihren entsprechenden Eigenschaften modelliert, d. h. wir haben einen Knoten (z. B. Knoten EMG), der den tatsächlichen Wert einschätzt, und einen Knoten (in unserem Beispiel dann der Knoten EMGSensor), der mit dem Wert instantiiert wird, der vom dazugehörenden Sensor gemessen, bzw. vom entsprechenden Vorverarbeitungs- bzw. Klassifikationsverfahren berechnet wurde. Wie man in Abbildung C.10 erkennen kann, werden hier die Sensoren mit ihren Eigenschaften noch nicht modelliert, sondern es wird in den Knoten direkt instantiiert. Die dem Knoten zugeordnete Tabelle bedingter Wahrscheinlichkeiten modelliert dabei die Zuverlässigkeit des Sensors, die u. a. von der Robustheit, der Anfälligkeit für Bewegungsartefakte oder der Messgenauigkeit abhängt. Ebenso können spontane Schwankungen (die auch als *Rauschen* bezeichnet werden) das Nutzsignal überlagern, an dem man eigentlich interessiert ist, wie z. B. Hautleitwertsfluktuationen beim Hautleitwert, die durch äußere Ereignisse ausgelöst werden (es sind also nicht die spontanen Hautleitwertsfluktuationen gemeint) oder spontane Schwankungen der Pupillenweite beim Pupillendurchmesser. Dabei ist es oft vom Kontext abhängig, wie zuverlässig ein Sensor tatsächlich ist (z. B. ob er mobil oder stationär eingesetzt wird),




(a) Bayessches Netz für absolute Daten. (b) Bayessches Netz für relative Daten.

Abbildung C.11: Bayessche Netze für absolute bzw. relative Daten im Vergleich.

so dass es sinnvoll sein kann, einen Kontextsensor für die Sensoreigenschaften einzuführen.

**Wahl der Hypothesen** Die Wahl der Hypothesen ist entscheidend davon abhängig, welche Art an Daten der Sensor oder der Vorverarbeitungsalgorithmus bzw. das Klassifikationsverfahren liefert. Dabei beeinflussen die Eigenschaften der physiologischen Größe wie der gesamte Messbereich, das Minimum, das Maximum und der Durchschnittswert (bzw. Durchschnittsbereich), der Schwankungsbereich (in Abhängigkeit des aktuellen Wertes) die Wahl der Hypothesen und ihren Wertebereich des entsprechenden Sensorknotens. In Abhängigkeit der Anzahl der Hypothesen und die Einteilung des Wertebereichs der Hypothesen bestimmen sich die bedingten Wahrscheinlichkeiten, die dem Knoten zugeordnet werden. Ebenso sind die bedingten Wahrscheinlichkeiten abhängig davon, ob absolute oder relative Sensordaten verwendet werden (beispielsweise “Herzschlagrate: 80 – 90” oder “leicht erhöhter Herzschlag”). Die absoluten Sensordaten müssten dann vom dynamischen Bayesschen Netz relativ zum Anwender eingeschätzt werden (ob beispielsweise der Herzschlag erhöht oder normal ist). In Abbildung C.11 sind zwei Bayessche Netze dargestellt, die die Verarbeitung von relativen und absoluten Sensordaten gegenüberstellen. Das Bayessche Netz in Abbildung C.11(a) verarbeitet absolute Sensordaten. Man gibt die absoluten Sensordaten (hier die Herzschlagrate) an und ob die Person sportlich oder eher unsportlich ist. Das Netz schätzt dann ein, ob die Herzschlagrate erhöht oder normal ist. In unserem Bei-

spiel ist sie erhöht. Dass die Herzschlagrate der Person erhöht ist, muss man im anderen Bayesschen Netz, das in Abbildung C.11(b) dargestellt ist, vom Vorverarbeitungsverfahren oder dem Klassifizierer berechnen und ins Bayessche Netz eintragen lassen.

Die Tabellen bedingter Wahrscheinlichkeiten der Knoten der beiden Bayesschen Netze sehen wie folgt aus. Bei den Knoten, die die beiden Bayesschen Netze gemeinsam haben, sind auch die Tabellen bedingter Wahrscheinlichkeiten identisch, d. h. die Tabelle bedingter Wahrscheinlichkeiten der Knoten Userstate1 und Userstate2 sieht wie folgt aus:

$$\mathrm{CPT}(\mathsf{Userstate\{1/2\}}) = \left[\begin{array}{r|l} \text{bad} & 0.5 \\ \text{good} & 0.5 \end{array}\right],$$

und die Tabelle bedingter Wahrscheinlichkeiten der Knoten Relative1 und Relative2 sieht wie folgt aus:

$$\mathrm{CPT}(\mathsf{Relative\{1/2\}}) = \left[\begin{array}{r|ll} \mathsf{Userstate\{1/2\}} & \text{bad} & \text{good} \\ \hline \text{low} & 0.0 & 0.8 \\ \text{middle} & 0.2 & 0.2 \\ \text{high} & 0.8 & 0.0 \end{array}\right].$$

Im Knoten Absolute1 sind nur Herzschlagraten von 40 bis 160 modelliert. Es müssten der Vollständigkeit halber auch Hypothesen modelliert sein, die Herzschlagraten sowohl unter 40 als auch Herzschlagraten über 160 als Eingabe akzeptieren. In unserem Modell wollen wir jedoch davon ausgehen, dass die Wahrscheinlichkeit für das Eintreten dieser Herzschlagraten bei 0 liegt. So können wir diese Hypothesen wegfallen lassen und die Komplexität des Netzes vereinfachen. Die Tabelle bedingter Wahrscheinlichkeiten des Knotens Absolute1 ist nun wie folgt:

$$\mathrm{CPT}(\mathsf{Absolute1}) = \left[\begin{array}{r|ccc|ccc} \mathsf{Person1} & \multicolumn{3}{c|}{\text{unathletic}} & \multicolumn{3}{c}{\text{sporty}} \\ \mathsf{Relative1} & \text{low} & \text{middle} & \text{high} & \text{low} & \text{middle} & \text{high} \\ \hline \text{40–60} & 0.5 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ \text{61–80} & 0.5 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ \text{81–100} & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.5 \\ \text{101–160} & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.5 \end{array}\right].$$

Der Knoten Absolute1 ist neben dem Knoten Relative1 auch vom Knoten Person1 abhängig, dessen Tabelle bedingter Wahrscheinlichkeiten wie folgt aussieht:

$$\mathrm{CPT}(\mathsf{Person1}) = \left[\begin{array}{r|l} \text{unathletic} & 0.5 \\ \text{sporty} & 0.5 \end{array}\right].$$

Ähnlich diesem Knoten könnte zusätzlich ein Knoten existieren, der noch den Kontext modelliert. Die Anzahl der Tabelleneinträge im Knoten Absolute1 würde entsprechend der Anzahl der Hypothesen des Kontext-Knotens multiplikativ ansteigen.

# Literaturverzeichnis

[Amemiya et al. 04] Tomohiro Amemiya, Jun Yamashita, Koichi Hirota and Michitaka Hirose. *Virtual Leading Blocks for the Deaf-Blind: A Real-Time Way-Finder by Verbal-Nonverbal Hybrid Interface and High-Density RFID Tag Space*. In *Proceedings of the 2004 Conference on Virtual Reality (VR'04)*, pp. 165–172. IEEE Virtual Reality, March 2004.

[Andersen et al. 89] S. K. Andersen, K. G. Olesen, F. V. Jensen and F. Jensen. *A Shell for Building Bayesian Belief Universes for Expert Systems*. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 1080–1085, 1989.

[Baldes et al. 05] Stephan Baldes, Alexander Kröner und Mathias Bauer. *Configuration and Introspection of Situated User Support*. In Bauer et al. [Bauer et al. 05], pp. 3–7.

[Bartsch 93] Hans-Jochen Bartsch. *Taschenbuch mathematischer Formeln*. Leipzig: Fachbuchverlag, 15., neubearbeitete Auflage, 1993.

[Bauer et al. 05] Mathias Bauer, Boris Brandherm, Johannes Fürnkranz, Gunter Grieser, Andreas Hotho, Andreas Jedlitschka und Alexander Kröner (Hrsg.). *Lernen, Wissensentdeckung und Adaptivität (LWA) 2005, GI Workshops, Saarbrücken, 10.–12. Oktober, 2005*. Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 2005.

[Becker & Geiger 96] Ann Becker and Dan Geiger. *A Sufficiently Fast Algorithm for Finding Close to Optimal Junction Trees*. In Horvitz and Jensen [Horvitz & Jensen 96], pp. 81–89.

[Besnard & Hanks 95] Philippe Besnard and Steve Hanks (eds.). *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI–95)*, Montréal, Québec, Canada, August 18–20, 1995. Morgan Kaufmann.

[Bohnenberger et al. 02] Thorsten Bohnenberger, Boris Brandherm, Barbara Großmann-Hutter, Dominik Heckmann und Frank Wittig. *Empirically Grounded Decision-Theoretic Adaptation to Situation-Dependent Resource Limitations*. Künstliche Intelligenz, 3:10–16, 2002.

[Bohnenberger 05] Thorsten Bohnenberger. *Decision-theoretic planning for user-adaptive systems: dealing with multiple goals and resource limitations*. DISKI: Dissertationen zur Künstlichen Intelligenz, Band 289. Berlin, Akademische Verlagsgesellschaft AKA, 2005.

[Boutilier & Goldszmidt 00] Craig Boutilier and Moisés Goldszmidt (eds.). *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–00)*, San Francisco, 2000. Morgan Kaufmann.

[Boyen & Koller 98] Xavier Boyen und Daphne Koller. *Tractable Inference for Complex Stochastic Processes*. In Cooper und Moral [Cooper & Moral 98], pp. 33–42.

[Boyen & Koller 99] Xavier Boyen und Daphne Koller. *Exploiting the Architecture of Dynamic Systems*. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI–99)*, pp. 313–320, Orlando, Florida, July 18–22 1999. Menlo Park: AAAI Press.

[Brandherm & Jameson 04] Boris Brandherm und Anthony Jameson. *An Extension of the Differential Approach for Bayesian Network Inference to Dynamic Bayesian Networks*. International Journal of Intelligent Systems, 19(8):727–748, 2004.

[Brandherm & Schwartz 05] Boris Brandherm und Tim Schwartz. *Geo Referenced Dynamic Bayesian Networks for User Positioning on Mobile Systems*. In Thomas Strang und Claudia Linnhoff-Popien (Hrsg.), *Proceedings of the International Workshop on Location- and Context-Awareness (LoCA), LNCS 3479*, Band 3479 / 2005: Lecture Notes in Computer Science, pp. 223–234, Munich, Germany, 2005. Springer-Verlag Berlin Heidelberg.

[Brandherm et al. 97] Boris Brandherm, Dirk Wagner und Frank Wittig. *Lösungsalgorithmen in mehrfach verbundenen Bayes'schen Netzen*. Fortgeschrittenenpraktikum, Lehrstuhl Prof. Wahlster, Fachrichtung 6.2 - Informatik, Universität des Saarlandes, Deutschland, 1997.

[Brandherm et al. 05] Boris Brandherm, Holger Schultheis, Margeritta von Wilamowitz-Moellendorff, Tim Schwartz und Michael Schmitz. *Using*

*Physiological Signals in a User-Adaptive Personal Assistant*. In *Proceedings of the 11th International Conference on Human-Computer Interaction (HCII-2005)*, Las Vegas, Nevada, USA, 2005.

[Brandherm 00] Boris Brandherm. *Rollup-Verfahren für komplexe dynamische Bayessche Netze*. Diplomarbeit, Lehrstuhl Prof. Wahlster, Fachrichtung 6.2 – Informatik, Universität des Saarlandes, Deutschland, 2000.

[Brandherm 01] Boris Brandherm. *Verarbeitung von Sensordaten in Dynamischen Bayesschen Netzen*. In Nicola Henze (Hrsg.), *ABIS-01, Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, pp. 288–293, Dortmund, Deutschland, 2001. Universität Dortmund.

[Brandherm 03] Boris Brandherm. *An Extension of the Differential Approach for Bayesian Network Inference to Dynamic Bayesian Networks*. In Ingrid Russell and Susan Haller (eds.), *Proceedings of the Sixteenth International FLAIRS Conference*, pp. 486–490, Menlo Park, California, 2003. AAAI Press.

[Bronstein & Semendjajew 89] Ilja N. Bronstein und Konstantin A. Semendjajew. *Taschenbuch der Mathematik*. Thun und Frankfurt/Main: Verlag Harri Deutsch, 24. Auflage, 1989.

[Butz & Krüger 06] Andreas Butz und Antonio Krüger. *Applying the Peephole Metaphor in a Mixed-Reality Room*. IEEE Computer Graphics and Applications, 26(1):56–63, 2006.

[Cooper & Moral 98] Gregory F. Cooper and Serafín Moral (eds.). *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–98)*, University of Wisconsin, Madison, Wisconsin, USA, July 24–26, 1998. Morgan Kaufmann Publishers.

[Cooper 84] Gregory F. Cooper. *NESTOR: A Computer-Based Medical Diagnostic Aid that integrated Causal and Probabilistic Knowledge*. PhD thesis, Department of Computer Science, Stanford University, California, 1984.

[Cooper 87] Gregory F. Cooper. *Probabilistic Inference using Belief Networks is NP-hard*. Technical Report KSL-87-27, Medical Computer Science Group, Stanford University, California, 1987.

[Cormen et al. 92] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Electrical Engineerinng and Computer Science Series. Cambridge, MA: The MIT Press, 1992.

[Cozman 01] Fabio Gagliardi Cozman. *JavaBayes*. http://www.cs.cmu.edu/ javabayes/, 2001.

[Dagum et al. 92] P. Dagum, A. Galper and E. Horvitz. *Dynamic Network Models for Forecasting*. In Dubois et al. [Dubois et al. 92], pp. 41–48.

[Darwiche 99] Adnan Darwiche. *A Differential Approach to Inference in Bayesian Networks*. Technical Report D–108, Computer Science Department, UCLA, Los Angeles, Ca 90095, 1999.

[Darwiche 00] Adnan Darwiche. *A Differential Approach to Inference in Bayesian Networks*. In Boutilier and Goldszmidt [Boutilier & Goldszmidt 00], pp. 123–132.

[Darwiche 01] Adnan Darwiche. *Constant-Space Reasoning in Dynamic Bayesian Networks*. International Journal of Approximate Reasoning, 26(3):161–178, 2001.

[Darwiche 03] Adnan Darwiche. *A Differential Approach to Inference in Bayesian Networks*. Journal of the Association for Computing Machinery, 50(3):280–305, 2003.

[Dubois et al. 92] Didier Dubois, Michael P. Wellman, Bruce D'Ambrosio and Phillipe Smets (eds.). *Proceedings of the Eighth Annual Conference on Uncertainty in Artificial Intelligence (UAI–92)*, Stanford University, Stanford, California, USA, July 17–19, 1992. Morgan Kaufmann Publishers.

[Duda et al. 76] R. O. Duda, P. E. Hart and N. J. Nilsson. *Subjective Bayesian Methods for Rule-based Inference Systems*. In *AFIPS Conference Proceedings of the 1976 National Computer Conference*, volume 45, pp. 1075–1082, 1976.

[Eyeled 06] Eyeled. *eyeled – mobile competence*. http://www.eyeled.de, 2006.

[Feld 06] Michael Feld. *Erzeugung von Sprecherklassifikationsmodulen für multiple Plattformen*. Diplomarbeit, Lehrstuhl Prof. Wahlster, Fachrichtung 6.2 - Informatik, Universität des Saarlandes, Deutschland, 2006.

[Fluidum ] Fluidum. *Flexible User Interfaces for Distributed Ubiquitious Machinery*. `http://www.fluidum.org`.

[Forbes et al. 95] Jeff Forbes, Tim Huang, Keiji Kanazawa and Stuart Russell. *The BATmobile: Towards a Bayesian Automated Taxi*. In Chris S. Mellish (ed.), *Proceedings of the Fourteenth International Joint Conference*

*on Artificial Intelligence*, pp. 1878–1885, San Mateo, CA, 1995. Morgan Kaufmann Publishers.

[Fox et al. 02] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz and Gaetano Borriello. *Bayesian Filtering for Location Estimation*. IEEE Pervasive Computing, 2(3):24–33, 2002.

[Friedman 04] Nir Friedman. *Inferring Cellular Networks Using Probabilistic Graphical Models*. Science, 303(5659):799–805, February 2004.

[Gong & Xiang 03] Shaogang Gong and Tao Xiang. *Recognition of Group Activities Using Dynamic Probabilistic Networks*. In *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003)*, volume 2, pp. 742–749, 2003.

[Goossens et al. 94] Michel Goossens, Frank Mittelbach und Alexander Samarin. *Der LaTeX-Begleiter*. Bonn; Paris; Reading, Mass. [u. a.]: Addison-Wesley, 1. Auflage, 1994. 1., korrigierter Nachdruck 1996.

[Harter et al. 02] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward und Paul Webster. *The Anatomy of a Context-Aware Application*. Wireless Networks, 8(2-3):187–197, 2002.

[Healey & Picard 00] Jennifer Healey und Rosalind Picard. *SmartCar: Detecting Driver Stress*. In *Proceedings of the 15th International Conference on Pattern Recognition ICPR'00*, Band 4, pp. 218–221, 2000.

[Healey & Picard 05] Jennifer A. Healey und Rosalind W. Picard. *Detecting Stress During Real-World Driving Tasks Using Physiological Sensors*. IEEE Transactions on Intelligent Transportation Systems, 6(2):156–166, 2005.

[Healey 00] Jennifer Healey. *Wearable and Automotive Systems for the Recognition of Affect from Physiology*. PhD thesis, MIT, Cambridge, MA, 2000.

[Heckmann et al. 05a] Dominik Heckmann, Tim Schwartz, Boris Brandherm and Alexander Kröner. *Decentralized User Modeling with UserML and GUMO*. In Peter Dolog and Julita Vassileva (eds.), *Decentralized, Agent Based and Social Approaches to User Modeling, Workshop DASUM-05 at 9th International Conference on User Modelling, UM2005*, pp. 61–66, Edinburgh, Scotland, Jul 2005.

[Heckmann et al. 05b] Dominik Heckmann, Tim Schwartz, Boris Brandherm, Michael Schmitz und Margeritta von Wilamowitz-Moellendorff. *GUMO*

*- the General User Model Ontology*. In *Proceedings of the 10th International Conference on User Modeling*, pp. 428–432, Edinburgh, UK, 2005. LNAI 3538: Springer, Berlin Heidelberg.

[Heckmann 06a] Dominik Heckmann. *Situation Modeling and Smart Context Retrieval with Semantic Web Technology and Conflict Resolution*, pp. 34–47. LNAI3946. Springer-Verlag, Berlin Heidelberg, 2006.

[Heckmann 06b] Dominik Heckmann. *Ubiquitous User Modeling*. DISKI: Dissertationen zur Künstlichen Intelligenz, Band 297. Berlin, Akademische Verlagsgesellschaft AKA, 2006. ISBN 3-89838-297-4.

[Horvitz & Barry 95] Eric Horvitz and Matthew Barry. *Display of Information for Time-Critical Decision Making*. In Besnard and Hanks [Besnard & Hanks 95], pp. 296–305.

[Horvitz & Jensen 96] Eric Horvitz and Finn Jensen (eds.). *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, Reed College, Portland, Oregon, USA, August 1–4, 1996. Morgan Kaufmann Publishers.

[Hotz 90] Günter Hotz. *Einführung in die Informatik*. Stuttgart: Teubner, 1990.

[Huang & Darwiche 96] Cecil Huang and Adnan Darwiche. *Inference in Belief Networks: A Procedural Guide*. International Journal of Approximate Reasoning, 15 (3):225–263, 1996.

[Hugin 04] Hugin. *Hugin Expert*. http://www.hugin.com, 2004.

[Identec 05] Identec. *Identec Solutions AG*. www.identecsolutions.com, 2005.

[Jain et al. 03] A. Jain, E. Gehde und D. Alfer. *Bedienungshandbuch Variograf*. Becker Meditec, Karlsruhe, Deutschland, 2003.

[Jameson et al. 06] Anthony Jameson, Jürgen Kiefer, Christian Müller, Barbara Großmann-Hutter, Frank Wittig und Ralf Rummer. *Assessment of a User's Time Pressure and Cognitive Load on the Basis of Features of Speech*. Manuscript submitted for publication, 2006.

[Jameson 90] Anthony Jameson. *Knowing What Others Know: Studies in Intuitive Psychometrics*. PhD thesis, University of Amsterdam, The Netherlands, 1990.

[Jensen & Jensen 94] Finn V. Jensen and Frank Jensen. *Optimal Junction Trees*. In Lopez de Mantaras and Poole [Lopez de Mantaras & Poole 94], pp. 360–366.

[Jensen 96] Finn V. Jensen. *An Introduction to Bayesian Networks*. New York: Springer, 1996.

[Johanson & Fox 02] Brad Johanson und Armando Fox. *The Event Heap: A Coordination Infrastructure for Interactive Workspaces*. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, pp. 83–93, 2002.

[Kanazawa et al. 95] Keiji Kanazawa, Daphne Koller and Stuart Russell. *Stochastic Simulation Algorithm for Dynamic Probabilistic Networks*. In Besnard and Hanks [Besnard & Hanks 95], pp. 346–351.

[Kiefer 02] Jürgen Kiefer. *Auswirkungen von Ablenkung durch gehörte Sprache und eigene Handlungen auf die Sprachproduktion*. Diplomarbeit, Fachrichtung 5.3 – Psychology, Universität des Saarlandes, Deutschland, 2002.

[Kipper et al. 95] Bernhard Kipper, Thorsten Brants, Marcus Plach und Ralph Schäfer. *Bayessche Netze: Ein einführendes Beispiel*. Technical Report 4, Graduiertenkolleg Kognitionswissenschaft, Saarbrücken, 1995.

[Kjærulff & van der Gaag 00] Uffe Kjærulff and Linda C. van der Gaag. *Making Sensitivity Analysis Computationally Efficient*. In Boutilier and Goldszmidt [Boutilier & Goldszmidt 00], pp. 317–325.

[Kjærulff 90] Uffe Kjærulff. *Triangulation of graphs – algorithms giving small total state space*. Research Report R 90-09, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1990.

[Kjærulff 92] Uffe Kjærulff. *A Computational Scheme for Reasoning in Dynamic Probabilistic Networks*. In Dubois et al. [Dubois et al. 92], pp. 121–129.

[Kjærulff 93] Uffe Kjærulff. *Aspects of Efficiency Improvement in Bayesian Networks*. PhD thesis, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1993.

[Kjærulff 95] Uffe Kjærulff. *dHugin: A Computational System for Dynamic Time-Sliced Bayesian Networks*. International Journal of Forecasting, Special Issue on Probability Forecasting, 11:89–111, 1995.

[Knappen et al. 94] Jörg Knappen, Hubert Partl, Elisabeth Schlegl und Irene Hyna. *LaTeX2ε-Kurzbeschreibung*, 20. Oktober 1994. Version 1.1.

[Knuth 86] Donald E. Knuth. *The TEXbook*, volume A: Computers and Typesetting. Reading, Mass.: Addison Wesley, 1986.

[Krüger et al. 04] Antonio Krüger, Andreas Butz, Christian Müller, Christoph Stahl, Rainer Wasinger, Karl-Ernst Steinberg und Andreas Dirschl. *The Connected User Interface: Realizing a Personal Situated Navigation Service*. In *IUI '04: Proceedings of the 9th International Conference on Intelligent User Interfaces*, pp. 161–168, New York, NY, USA, 2004. ACM Press.

[Kruppa et al. 05] Michael Kruppa, Lübomira Spassova und Michael Schmitz. *The Virtual Room Inhabitant – Intuitive Interaction with Intelligent Environments.* In Shichao Zhang und Ray Jarvis (Hrsg.), *Australian Conference on Artificial Intelligence*, Band 3809: Lecture Notes in Computer Science, pp. 225–234. Springer, 2005.

[Kruppa 05] Michael Kruppa. *Migrating Characters Performing Explicit Physical Object References*. In Bauer et al. [Bauer et al. 05], pp. 47–49.

[Kruppa 06] Michael Kruppa. *Migrating Characters: Effective User Guidance in Instrumented Environments*. DISKI: Dissertationen zur Künstlichen Intelligenz, Band 301. Berlin, Akademische Verlagsgesellschaft AKA, 2006.

[Lamprecht 78] Ernst Lamprecht. *Einführung in die Algebra*. Basel und Stuttgart: Birkhäuser Verlag, 1. Auflage, 1978.

[Laskey & Prade 99] Kathryn Laskey and Henri Prade (eds.). *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–99)*, Stockholm, Sweden, July 30–August 1, 1999. Morgan Kaufmann Publishers.

[Lindmark 00] Kristin Lindmark. *Interpreting Symptoms of Cognitive Load and Time Pressure in Manual Input*. Diplomarbeit, Lehrstuhl Prof. Wahlster, Fachrichtung 6.2 – Informatik, Universität des Saarlandes, Deutschland, 2000.

[Lopez de Mantaras & Poole 94] Ramon Lopez de Mantaras and David Poole (eds.). *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–94)*, University of Washington, Seattle, July 29–31, 1994. Morgan Kaufmann Publishers.

[Madsen & Jensen 98] Anders L Madsen and Finn V Jensen. *Lazy Propagation in Junction Trees*. In Cooper and Moral [Cooper & Moral 98], pp. 362–369.

[MatLab 06] MatLab. *MatLab*. http://www.mathworks.com, 2006.

[Müller et al. 01] Christian Müller, Barbara Großmann-Hutter, Anthony Jameson, Ralf Rummer and Frank Wittig. *Recognizing Time Pressure and Cognitive Load on the Basis of Speech: An Experimental Study*. In Mathias Bauer, Piotr Gmytrasiewicz and Julita Vassileva (eds.), *UM2001, User Modeling: Proceedings of the Eighth International Conference*, pp. 24–33. Berlin: Springer, 2001.

[Müller 06] Christian Müller. *Zweistufige kontextsensitive Sprecherklassifikation am Beispiel von Alter und Geschlecht*. DISKI: Dissertationen zur Künstlichen Intelligenz, Band 296. Berlin, Akademische Verlagsgesellschaft AKA, 2006.

[Monstadt 06] Ralf Monstadt. *Geschwindigkeitsabhängige Adaption des Bewegungsmodells von georeferenzierten dynamischen Bayesschen Netzen in einem Indoor-Positionierungssystem*. Diplomarbeit, Lehrstuhl Prof. Wahlster, Fachrichtung 6.2 – Informatik, Universität des Saarlandes, Deutschland, 2006.

[Neapolitan 90] Richard E. Neapolitan. *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. New York: Wiley, 1990.

[Nefian et al. 02] A. Nefian, L. Liang, X. Pi, X. Liu and K. Murphy. *Dynamic Bayesian Networks for Audio-Visual Speech Recognition*. EURASIP Journal on Applied Signal Processing, 11:1–15, 2002.

[Netica 06] Netica. *Netica*. http://www.norsys.com/, 2006.

[Ni et al. 04] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau and Abhishek P. Patil. *LANDMARC: Indoor Location Sensing Using Active RFID*. Wireless Networks, 10(6):701–710, 2004.

[Nicholson & Brady 92] A. E. Nicholson and J. M. Brady. *Sensor Validation Using Dynamic Belief Networks*. In Dubois et al. [Dubois et al. 92], pp. 207–214.

[Park & Darwiche 03] James D. Park und Adnan Darwiche. *A Differential Semantics for Jointree Algorithms*. In Suzanna Becker, Sebastian Thrun und Klaus Obermayer (Hrsg.), *Advances in Neural Information Processing Systems 15*, pp. 785–784. Cambridge, MA: MIT Press, 2003.

[Park & Darwiche 04] James D. Park and Adnan Darwiche. *A differential semantics for jointree algorithms*. Artificial Intelligence, 156(2):197–216, 2004.

[Pearl 88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.

[Picard 97] Rosalind W. Picard. *Affective Computing*. Cambridge, MA: MIT Press, 1997.

[Rowe et al. 98] Dennis W. Rowe, John Sibert and Don Irwin. *Heart Rate Variability: Indicator of User State as an Aid to Human-Computer Interaction*. In Clare-Marie Karat, Arnold Lund, Joëlle Coutaz and John Karat (eds.), *Human Factors in Computing Systems: CHI '98 Conference Proceedings*, pp. 480–487. New York: ACM, 1998.

[Russell & Norvig 03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice-Hall, second edition edition, 2003.

[Schandry 98] Rainer Schandry. *Lehrbuch Psychophysiologie – Körperliche Indikatoren psychischen Geschehens*. Psychologie Verlags Union, Weinheim, 1998.

[Schäfer & Weyrath 97] Ralph Schäfer and Thomas Weyrath. *Assessing Temporally Variable User Properties With Dynamic Bayesian Networks*. In Anthony Jameson, Cécile Paris and Carlo Tasso (eds.), *User Modeling: Proceedings of the Sixth International Conference, UM97*, pp. 377–388. Wien, New York: Springer, 1997.

[Schäfer 99] Ralph Schäfer. *Benutzermodellierung mit dynamischen Bayes'schen Netzen als Grundlage adaptiver Dialogsysteme*. Dissertation, Lehrstuhl Prof. Wahlster, Fachrichtung 6.2 – Informatik, Universität des Saarlandes, Deutschland, 1999.

[Schmitz & Butz 06] Michael Schmitz and Andreas Butz. *SAFIR: Low-Cost Spatial Audio for Instrumented Environments*. In *Proceedings of the 2nd International Conference on Intelligent Environments*, pp. 427–430, Athens, Greece, July 2006.

[Schmitz 03] Michael Schmitz. *Ein Framework für räumliches Audio in instrumentierten Umgebungen*. Diplomarbeit, Lehrstuhl Prof. Wahlster, Fachrichtung 6.2 – Informatik, Universität des Saarlandes, Deutschland, 2003.

[Schultheis & Jameson 04] Holger Schultheis und Anthony Jameson. *Assessing Cognitive Load in Adaptive Hypermedia Systems: Physiological and Behavioral Methods*. In Wolfgang Nejdl und Paul De Bra (Hrsg.), *Adaptive*

*Hypermedia and Adaptive Web-Based Systems: Proceedings of AH 2004*, pp. 225–234. Berlin: Springer, 2004.

[Schwartz et al. 05] Tim Schwartz, Boris Brandherm and Dominik Heckmann. *Calculation of the User-Direction in an Always Best Positioned Mobile Localization System*. In *Proceedings of the International Workshop on Artificial Intelligence in Mobile Systems (AIMS)*, Salzburg, Austria, September 2005.

[Shafer 96] Glenn Shafer. *Probabilistic Expert Systems*. CBMS-NSF regional conference series in applied mathematics. Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics, 1996.

[Spiegelhalter 86] David J. Spiegelhalter. *Uncertainty in Artificial Intelligence*, chapter Probabilistic Reasoning in Predictive Expert Systems, pp. 47–68. Amsterdam: North Holland, 1986.

[Stahl & Heckmann 04] Christoph Stahl und Dominik Heckmann. *Using Semantic Web Technology for Ubiquitous Hybrid Location Modeling*. In *UbiGis 2004 - First International Workshop on Ubiquitous Geo Information Systems*, pp. 1–9, Gävle, Sweden, June 2004.

[Stahl et al. 05] Christoph Stahl, Jörg Baus, Boris Brandherm, Michael Schmitz und Tim Schwartz. *Navigational- and Shopping Assistance on the Basis of User Interactions in Intelligent Environments*. In *Proceedings of the IEE International Workshop on Intelligent Environments (IE)*, pp. 182–191, University of Essex, Colchester, UK, 2005. IEEE Computer Society.

[Tawfik & Neufeld 94] A. Y. Tawfik and E. Neufeld. *Temporal Bayesian Network*. In S. D. Goodwin and H. J. Hamilton (eds.), *Proceedings of TIME-94 – International Workshop on Temporal Representation and Reasoning*, p. keine Angabe der Seitenzahlen, Pensacola, FL, 1994.

[Ubisense 05] Ubisense. *Ubisense Unlimited*. http://www.ubisense.net, 2005.

[van der Gaag & Coupé 98] Linda C. van der Gaag and Veerle M. H. Coupé. *Practicable sensitivity analysis of Bayesian belief networks*. In M. Hušková, P. Lachout and J. A. Víšek (eds.), *Proceedings of the Joint Session of the 6th Prague Symposium of Asymptotic Statistics and the 13th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, pp. 81–86. Union of Czech Mathematicians and Physicists, 1998.

[von Wilamowitz-Moellendorff et al. 05] Margeritta von Wilamowitz-Moellendorff, Christian Müller, Anthony Jameson, Boris Brandherm und Tim Schwartz. *Recognition of Time Pressure via Physiological Sensors: Is the User's Motion a Help or a Hindrance?* In *Proceedings of the Workshop on Adapting the Interaction Style to Affective Factors in conjunction with the User Modeling (UM05) conference*, Edinburgh, UK, 2005.

[Wahlster & Tack 97] Wolfgang Wahlster und Werner Tack. *Ressourcenadaptive Kognitive Prozesse*. In M. Jarke, K. Pasedach und K. Pohl (Hrsg.), *Informatik 97 - Informatik als Innovationsmotor, 27. GI-Jahrestagung*, pp. 51–57. Springer, 1997.

[Want et al. 92] Roy Want, Andy Hopper, Veronica Falcao and Jonathon Gibbons. *The Active Badge Location System*. ACM Transanctions on Information Systems, 10(1):91–102, January 1992.

[Wasinger et al. 03] Rainer Wasinger, Dominika Oliver, Dominik Heckmann, Bettina Braun, Boris Brandherm und Christoph Stahl. *Adapting Spoken and Visual Output for a Pedestrian Navigation System, based on given Situational Statements*. In Andreas Hotho und Gerd Stumme (Hrsg.), *LLWA 2003, Lehren Lernen Wissen Adaptivität*, pp. 343–346, Karlsruhe, Germany, 2003.

[Wittig 03] Frank Wittig. *Maschinelles Lernen Bayes'scher Netze für benutzeradaptive Systeme*. DISKI: Dissertationen zur Künstlichen Intelligenz, Band 267. Berlin, Akademische Verlagsgesellschaft AKA, 2003. ISBN 3-89838-267-2.

[Xiang & Jensen 99] Yang Xiang and Finn Jensen. *Inference in Multiply Sectioned Bayesian Networks with Extended Shafer-Shenoy and Lazy Propagation*. In Laskey and Prade [Laskey & Prade 99], pp. 680–687.

[Zweig & Russell 98] Geoffrey Zweig und Stuart J. Russell. *Speech Recognition with Dynamic Bayesian Networks*. In Cooper und Moral [Cooper & Moral 98], pp. 173–180.