A Machine-Checked Constructive Metatheory of Computation Tree Logic

Christian Doczkal

Dissertation zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Saarbrücken, 2015

Dissertation zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

eingereicht von Christian Doczkal, M.Sc.

Berichterstatter:
Prof. Dr. Gert Smolka, Universität des Saarlandes
Prof. Dr. Martin Lange, Universität Kassel

Dekan:
Prof. Dr. Markus Bläser

Prüfungsausschuss:
Prof. Dr. Christoph Weidenbach (Vorsitz)
Prof. Dr. Gert Smolka
Prof. Dr. Martin Lange
Dr. Swen Jacobs

Tag des Kolloquiums:
21. März 2016

Textfassung vom 31. März 2016

Abstract

This thesis presents a machine-checked constructive metatheory of computation tree logic (CTL) and its sublogics K and K* based on results from the literature. We consider models, Hilbert systems, and history-based Gentzen systems and show that for every logic and every formula s the following statements are decidable and equivalent: s is true in all models, s is provable in the Hilbert system, and s is provable in the Gentzen system. We base our proofs on pruning systems constructing finite models for satisfiable formulas and abstract refutations for unsatisfiable formulas. The pruning systems are devised such that abstract refutations can be translated to derivations in the Hilbert system and the Gentzen system, thus establishing completeness of both systems with a single model construction.

All results of this thesis are formalized and machine-checked with the Coq interactive theorem prover. Given the level of detail involved and the informal presentation in much of the original work, the gap between the original paper proofs and constructive machine-checkable proofs is considerable. The mathematical proofs presented in this thesis provide for elegant formalizations and often differ significantly from the proofs in the literature.

Kurzzusammenfassung

Diese Dissertation beschreibt eine maschinell verifizierte konstruktive Metatheorie von computation tree logic (CTL) und deren Teillogiken K und K*. Wir betrachten Modelle, Hilbert-Kalküle und History-basierte Gentzen-Kalküle und zeigen, für jede betrachtete Logik und jede Formel s, Entscheidbarkeit und Äquivalenz der folgenden Aussagen: s gilt in allen Modellen, s ist im Hilbert-Kalkül ableitbar und s ist im Gentzen-Kalkül ableitbar. Die Beweise bauen auf Pruningsystemen auf, welche für erfüllbare Formeln endliche Modelle und für unerfüllbare Formeln abstrakte Widerlegungen konstruieren. Die Pruningsysteme sind so konstruiert, dass abstrakte Widerlegungen zu Widerlegungen sowohl im Hilbert- als auch im Gentzen-Kalkül übersetzt werden können. Dadurch wird es möglich, die Vollständigkeit beider Systeme mit nur einer Modellkonstruktion zu zeigen.

Alle Ergebnisse dieser Dissertation sind formalisiert und maschinell verifiziert mit Hilfe des Beweisassistenten Coq. In Anbetracht der Fülle an Details und der informellen Beweisführung in großen Teilen der Originalliteratur, erfordert dies teilweise tiefgreifende Veränderungen an den Beweisen aus der Literatur. Die Beweise in der vorliegenden Arbeit sind so aufgebaut, dass sie zu eleganten Formalisierungen führen.

Acknowledgments

This thesis presents the results of several years of research. Many people have contributed to this research in different ways. I take this opportunity to thank those to whom I owe a particularly large debt of gratitude.

First and foremost I want to thank my supervisor Gert Smolka. He granted me the freedom to develop my own ideas and encouraged me to strive for simplicity in presenting them. In countless discussions, he provided guidance and invaluable advice on nearly every aspect of my work. I also thank Martin Lange for giving his expert opinion on this thesis.

During my work on this thesis I was employed at the Programming Systems Lab at Saarland University. I want to thank my current and former colleagues: Gert, Chad E. Brown, Jonas Kaiser, Mark Kaminski, Steven Schäfer, Thomas Schneider, Jan Schwinghammer, and Tobias Tebbi for the many interesting discussions that made working at the lab a great experience. Special thanks go to Chad, from whom I know most of what I know about constructive proofs, and Mark, whose work on modal logic had great influence on my own.

As part of my work I had the privilege to advise two very good students: Jan-Oliver Kaiser and Alexander Anisimov. Working with them, learning from them, and passing on some of the knowledge I acquired during my research was a truly gratifying experience. Further, I want to thank our student assistant Tobias Blass for helping me with system administration, thus making it much less of a burden. I am also grateful to Jonas for taking over system administration while I was finishing this thesis.

Last but not least I want to thank my family for their patience and continued support and my friends who bore with me in the final stages of my writing.

Contents

1 Introduction

This thesis presents a machine-checked constructive metatheory of computation tree logic (CTL) and its sublogics K and K* based on results from the literature. We prove soundness and completeness of Hilbert systems and history-based Gentzen systems, small-model theorems, and decidability results. All results are formalized and machine-checked with the Coq interactive theorem prover. Given the level of detail involved and the informal presentation in much of the original work, the gap between the original paper proofs and constructive machine-checkable proofs is considerable. The mathematical proofs presented in this thesis provide for elegant formalizations and often differ significantly from the proofs in the literature.

1.1 Related Work

This thesis builds on related work from two largely independent areas. We first survey related work on basic modal logics and modal logics with eventualities (e.g., PDL and CTL). Afterwards, we provide some background on constructive type theory and interactive theorem proving in Coq.

1.1.1 Basic Modal Logic

The term "modal logic" refers to a broad family of logics ranging from logics for reasoning about knowledge and beliefs to logics used for program verification. According to Blackburn et al. [BdRV01], modal logic as a mathematical discipline emerged 1918 with Lewis's *Survey of Symbolic Logic* [Lew18]. Lewis axiomatized a logic extending propositional logic with a single unary modality I ("it is impossible that"). Viewed from todays perspective, the axiom systems of Lewis and Langford [Lew18, LL32] are strange in so far as there is no separation between modal and propositional axioms. The nowadays standard modular Hilbert systems for modal logics consisting of modus ponens, propositional axioms, and a collection of dedicated modal axioms and rules appeared first with Gödel's axiomatization of the modal logic S4 [Göd33].

The initial work on modal logic was purely syntactic. This changed in the late 1950s with the introduction of *relational semantics*. Relational semantics is usually attributed to Kripke [Kri59, Kri63] and therefore also known as *Kripke semantics*. See [Gol06] for a historical account. Relational models are transition systems where the states are labeled with atomic propositions. Formulas are then evaluated at individual states of a model. For instance, an atomic formula p holds at a state w of some model $\mathcal{M}$ (usually written $\mathcal{M}, w \models p$) if w is labeled with p. Relational models allow for simple semantic characterizations of many modal logics. The modalities $\Box$ and $\Diamond$ of basic modal logic K extend propositional logic with quantification over direct successor states. The formula $\Box s$ holds at some state w if s holds at at all immediate successors of w and $\Diamond s$ holds at w if s holds at some immediate successor of w.

With relational semantics come the notions of *satisfiability* and *validity*. A formula is satisfiable if it holds at some state of some model and valid if it holds at all states of all models. This naturally leads to the question of *completeness*, i.e., the question whether a given proof system can prove all valid formulas of a logic. Completeness results dominated the technical work on modal logic for over a decade [Seg71, LS77]. One of the main techniques for establishing completeness is the construction of *canonical models*, i.e., models whose states consist of maximally consistent sets of formulas and where every state satisfies all formulas it contains (cf. [Fit07]).

For many modal logics, satisfiability of formulas is decidable. One way of establishing decidability is by showing that the logic has the *small-model property*. A logic has the small model property if every satisfiable formula has a model whose size (in terms of the number of states) can be bounded by some function in the size of the formula. This can often be established via *filtration* [LS77]. The filtration of a model $\mathcal{M}$ with respect to the subformulas of some formula s is a quotient of $\mathcal{M}$ where all states that agree on the finitely many subformulas of s are identified. The size of such a quotient is at most exponential in the size of s and for many logics (e.g., K) will satisfy s whenever $\mathcal{M}$ satisfies s. Filtration, if applicable, also establishes a form of *subformula property*, i.e., that in order to determine satisfiability of some formula s it suffices to consider finite syntactic models whose states are sets of subformulas of s and where every state satisfies all formulas it contains.

While filtration yields decidability, it does not yield reasonable decision methods. One of the most successful methods both for theoretical analysis and practical decision procedures is the *tableau method*. In its original form, the tableau method was developed by Beth [Bet55]. The first application to modal logic is due to Kripke [Kri59]. In the case of modal logics with the subformula property, tableau proofs can be seen as the search for a syntactic model. If this search fails, the search tree can be seen as a proof that the starting formula is unsatisfiable. That is, tableau systems are at the same time refutation calculi and devices for the construction of models. Many tableau systems can also be formulated as Gentzen systems. Fitting [Fit83] gives Gentzen systems for a variety of modal logics.

1.1.2 Modal logic with Eventualities

Modal logics for reasoning about programs like PDL [FL79, Pra79], UB [Pnu77, BAPM83] and CTL [EC82, EH85, Eme90] extend basic modal logic with various eventualities. The simplest logic with eventualities is the logic K^* [KS10] (introduced as “nexttime” logic in [MP79] and referred to as UB^- in [EH85]). K^* extends K with quantification over transitively reachable states. The formula $\Diamond^* s$ holds at a state w if s holds at some state reachable from w. Following [Pnu77], we call $\Diamond^* s$ an *eventuality* since after finitely many transitions s must eventually hold. The state satisfying s is said to *fulfill* the eventuality. Dually, the formula $\Box^* s$ holds at a state if s holds at every reachable state. We call formulas of the form $\Box^* s$ *invariants* since $\Box^* s$ must hold again after taking a transition.

K^* is a subsystem of PDL and shares many of its metatheoretic properties. While satisfiability for K is PSPACE-complete [Lad77], the satisfiability problem for K^* is EXPTIME-complete [FL79, Pra79, BdRV01]. Moreover, eventualities cause the logic to be non-compact (consider the unsatisfiable set $\{\Diamond^* \neg p, \Box p, \Box\Box p, \dots\}$ of which every strict subset is satisfiable). That is, although K^* is decidable, it is not subsumed by first-order logic. Segerberg [Seg77] gives a Hilbert system for PDL that can easily be adapted to K^*. Due to the non-compactness of K^*/PDL, the standard construction for canonical models does not apply. Completeness of Segerberg’s axiomatization was shown independently by Gabbay [Gab77] and Parikh [Par78](cf. [HKT00]). There are a number of alternative completeness proofs for PDL in the literature. These proofs either construct non-standard canonical models and use filtration to obtain finite counter-models for unprovable formulas [Ber79, HKT00] or directly construct finite models [KP81].

CTL is a temporal logic used for model checking [CES83, EL86, Eme08, BK08]. The logic is interpreted over Kripke models where every state has at least one successor. Syntactically, CTL extends K^* in two directions. It generalizes the eventuality $\Diamond^* s$ to an eventuality $\mathsf{E}(s \,\mathsf{U}\, t)$ (read *exists s until t*) which holds at some state w if there is some (infinite) path starting at w where s holds at every state until a state satisfying t is reached. In addition to these existential eventualities, CTL also features universal eventualities $\mathsf{A}(s \,\mathsf{U}\, t)$ that hold at some state w if for every infinite path starting at w the formula s holds at every state until a state satisfying t is reached.

While filtration preserves satisfaction for existential eventualities, this is not the case for the universal eventualities of CTL. The quotient construction may introduce cycles in the model along which the eventuality is never fulfilled. Nevertheless, filtration for CTL yields pseudo-models than can be unfolded into proper finite models [EH85, Eme90]. That is, CTL has the small model property and also a form of subformula property.

Arguably the simplest algorithm for deciding satisfiability in the presence of eventualities is *pruning* [Pra79]. Originally developed to establish the EXPTIME upper bound for PDL, pruning has also been used to show EXPTIME decidability for CTL [EH85] and hybrid PDL [KSS11]. Given some input formula s, pruning starts with the collection of all Hintikka sets built from subformulas of s. Here, Hintikka sets are sets of formulas satisfying closure conditions ensuring propositional consistency. Every Hintikka set can be seen as a potential state of a model. Pruning then successively removes Hintikka sets containing $\Diamond$-formulas for which there is no successor or eventualities that cannot be fulfilled. The process terminates with exactly the satisfiable Hintikka sets over the subformulas of s.

Beyond showing decidability of satisfiability, pruning can also serve as a basis for completeness proofs. Emerson and Halpern [EH85, Eme90] show completeness of a Hilbert system for CTL by showing that every Hintikka set that is removed during pruning can be refuted. This sidesteps the issues arising from the non-compactness of the logic.

Other decision methods for logics with eventualities are based on tableau methods [BAPM83, EH85, Wid10, Kam12]. Many of these methods can be seen as optimized incremental versions of pruning, i.e., they combine local expansion rules with global rules for eventuality checking. As a consequence, these methods cannot easily be formulated as Gentzen systems. Brünnler and Lange [BL08] present a Gentzen system for CTL based on a game-theoretic interpretation of the logic [LS01]. The system is non-standard in that eventualities are annotated with histories, which are sets of sets of formulas. Histories are needed to handle eventualities with local rules.

1.1.3 Interactive Theorem Proving

Interactive theorem provers based on type theory are being used to establish theorems with a degree of certainty well beyond paper proofs. This ranges from machine-checked proofs for mathematical theories such as Landau's *Grundlagen der Arithmetik* [vBJ77], the Four-Color Theorem [Gon08], the Feit-Thompson Odd Order Theorem [GAA$^+$13] or the Kepler Conjecture [HAB$^+$15] to certified software such the CompCert C compiler [Ler06, BDL06] or the seL4 microkernel [KDE09]. Beyond these "landmark projects", the last decades have produced a large body of formalizations on a wide variety of topics.

While for long and complex proofs the attained degree of certainty is often the most important reason for formalization, there are other motivations to formalize mathematics. In particular for small and medium size developments, there is often little doubt in the correctness of the formalized result. For these developments the main motivation is usually the desire to obtain a deeper understanding of the results and the techniques required to obtain elegant formalizations in a given system. One example is the theory of regular languages, parts of which have been formalized in a variety of proof assistants. This includes both purely mathematical developments [CJNU00, WZU14, DKS13, Pau15] as well as executable certified decision methods [BR09, CS11, BP12].

In this thesis, we use the proof assistant Coq [Coq15] with the Ssreflect [GMT08] extension. The Coq system is based on the *propositions-as-types* principle which has its roots in the Brouwer, Heyting, Kolmogorov (BHK) interpretation of intuitionistic logic (See [Wad15] for a historical overview). They key idea of propositions-as-types is that propositions can be seen as the types of some λ-calculus with terms of those types interpreted as proofs. The first implemented system employing propositions-as-types was the Automath system [dB68].

Coq implements a type theory called the predicative Calculus of Inductive Constructions (pCIC) [Coq15]. It combines dependent types and the impredicative universe of propositions from the Calculus of Constructions [CH88] with an infinite hierarchy of predicative type universes [ML84, Luo89, Luo94] as well as primitive inductive [Wer94] and coinductive types [Gim94].

The logic of Coq is constructive by default and designed to be consistent with classical assumptions such as *excluded middle* and various choice principles. For constructive proofs, decidability properties are of great importance. While case distinctions on arbitrary properties are not permitted in constructive proofs, case distinctions on decidable properties are always permitted. Consequently, constructive proofs often involve establishing decidability properties. Originally developed for the formalization of the Four-Color Theorem, Ssreflect [GMT08] extends Coq with a tactic language designed to simplify reasoning about decidable properties. In addition to the tactic language, the Ssreflect extension comes with a comprehensive library for reasoning about discrete structures [GMR$^+$07, GGMR09, BGBP08].

1.2 Motivation

The original proofs of the metatheoretic results for CTL are of considerable complexity and presented in a fairly informal manner. This applies to the proofs of the small-model property and the completeness of Hilbert axiomatizations [EH85, Eme90, LS01] as well as the completeness proof for the Gentzen system given by Brünnler and Lange [BL08]. Given the practical importance of CTL and the complexity of the proofs of the metatheoretic results, CTL yields an interesting and rewarding candidate for formalization.

We formalize our results using the Coq interactive theorem prover. The expressive type theory implemented by Coq allows for natural representations of all required concepts (e.g., formulas, models, and proof systems). In particular, our representation of models makes use of the fact that Coq, unlike HOL-based systems like Isabelle/HOL [NPW02], treats types as first-class objects.

The proof assistant Coq is constructive by default and there is a strong tradition in the type theory community to not assume unnecessary axioms. Given that CTL is decidable and has the small-model property, it is to be expected that much of the metatheory of CTL and its sublogics can be obtained constructively. In fact, some of the completeness proofs for temporal logics in the literature are constructive. A constructive completeness proof for a Hilbert system for UB (a subsystem of CTL) appears in the work of Ben-Ari et al. [BAPM83]. The proof is based on a tableau procedure deciding satisfiability and a construction of Hilbert refutations for the case where the tableau procedure fails to find a model. Also, Brünnler and Lange [BL08] give a constructive completeness proof for a Gentzen system for CTL using a variant of the Gentzen system itself as the underlying decision method.

While most of our results can be established constructively, there are a few results that are inherently classical. This mainly applies to results about infinite models (e.g., soundness and certain formulations of the small-model theorem). In a constructive setting, we can make the need for classical assumptions formal by showing that the respective result not only follows from some classical axiom, but is in fact equivalent to it. This allows us to provide a fine-grained analysis showing which results can be obtained constructively and where classical assumptions are essential.

For our constructive completeness results we establish a number of decidability properties that are of independent interest (e.g., decidability of model checking and satisfiability). Here we profit from the fact that constructive type theory comes with a built-in notion of decidability. This allows us to establish decidability results without first formalizing a model of computation.

1.3 Overview of this Thesis

Rather than heading directly to CTL and proving completeness of a Hilbert system and a history-based Gentzen system in one monolithic development, we present our results in stages. We start of with basic modal logic K and subsequently extend this development to modal logic with transitive closure (K*) and CTL. The motivation for this is twofold. First, the presentation in stages allows us to discuss each of the issues arising during the development in the simplest possible context. Moreover, the accompanying formal developments share a lot of structure, thus prompting the development of generic lemmas and reusable libraries.

1.3.1 Informative Decision Methods

We prove our completeness results in the form of *informative decision methods*. For each proof system, we define a function[1] in Coq which for a given input formula s either returns a finite model satisfying s or a proof of $\neg s$. Our proofs extend and refine the proofs from [Eme90] where completeness of a Hilbert system for CTL is shown based on a pruning procedure deciding satisfiability.

The central notion in the design of our informative decision methods is the notion of a *demo*. Demos are finite pseudo-models built from finite sets of formulas we call *clauses*. More precisely, demos are sets of clauses satisfying a number of closure conditions. These closure conditions ensure the existence of a model whose states are labeled with clauses from the demo such that every state satisfies all formulas in its labeling clause.

[1] While all functions one can construct in Coq are computable, the functions we construct will not be practically executable.

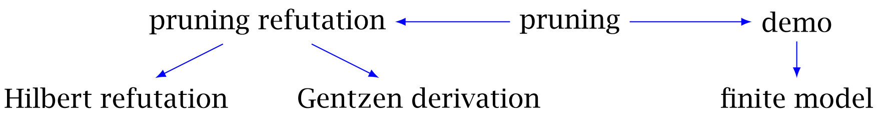


Figure 1.1: Structure of completeness proofs

Demos sit at the heart of our informative decision methods. The demo conditions determine a system of pruning rules. Starting from the clauses built over the subformulas of the input formula, pruning successively removes clauses violating the demo conditions until what remains is a demo. Consequently, the demo conditions also give rise to a refutation calculus that complements pruning, i.e., that derives all clauses removed during pruning. To establish completeness of some proof system based on pruning there are two subtasks:

- ▷ The construction of finite models from demos
- ▷ The translation of pruning refutations to derivations of the proof system.

In the case of K and K*, demos are essentially models. Due to the failure of filtration for CTL, demos for CTL are merely pseudo-models and need to be unfolded into proper models [EH85, Eme90].

For each of the logics K, K*, and CTL we translate pruning refutations to Hilbert refutations and to derivations of a Gentzen system for the respective logic. Since the construction of models from demos is independent from the translation arguments, we obtain for each logic two completeness results with a single model construction (cf. Figure 1.1).

For K we employ the standard Hilbert system from the literature [Fit07]. For K* and CTL we give Hilbert systems motivated by an inductive interpretation of eventualities. We also establish completeness of other Hilbert systems for CTL [Eme90, LS01] by proving them equivalent to our Hilbert system. The Gentzen system for K is essentially a clausal presentation of Fitting's destructive tableaux [Fit07]. For K* and CTL, we employ variants of the history-based Gentzen system for CTL developed by Brünnler and Lange [BL08].

In particular for CTL, the construction of models from demos and the translations to Hilbert and Gentzen proofs are of considerable complexity. We formulate the closure conditions for demos to give a good compromise between the complexity of the model construction and the translation arguments. For our demos we use literal clauses and the notion of support [KS10, KS14] instead of the more traditional notion of Hintikka sets [Pra79, EH85, KSS11]. (The term demo appears first in [KSS11], but clausal demos for K* already appear in [KS10] as evident branches.) While the Hintikka sets employed for modal logics are finite sets that are downward saturated with respect to certain propositional decomposition rules, the collection of formulas supported by a literal clause corresponds to an infinite (but decidable) Hintikka set that is both downward and upward closed. The notion of support can be defined using a simple recursive function which is convenient for the formal proofs. Moreover, the notion of support provides a natural fit for the destructive (i.e., non-cumulative) reading of the Gentzen systems from [BL08].

The crucial part in the definition of demos is the treatment of eventualities. We handle eventualities using inductively defined fulfillment predicates. In the case of CTL, fulfillment predicates replace the test for fragments embedded in the demo employed by Emerson and Halpern [EH85, Eme90]. Pruning refutations employ inductive fulfillment in negated form and non-fulfillment has exactly the closure properties required for the translation to Hilbert refutations. We give a simple bottom-up construction of fragments from inductive fulfillment. For the construction of models from fragments we adapt the declarative construction from [Eme90] rather than the iterative construction from [EH85].

The translation from pruning refutations to Hilbert refutations turns out to be fairly robust as it comes to minor variations of the pruning rules. In contrast, due to the analyticity of the Gentzen systems, the precise formulation of the pruning rules is crucial for the translation of pruning refutations to Gentzen derivations to succeed. In the case of K, the connection between pruning refutations and Gentzen derivations is immediate. In the presence of eventualities, the situation becomes more complex. The history-based systems introduced by Brünnler and Lange [BL08] employ rules allowing to put a focus on some eventuality. Once an eventuality is put in focus, no other eventuality may be focused on. This conflicts with pruning. Pruning constructs derivations in a bottom-up manner and the Gentzen derivations for previously pruned clauses may employ the focusing rules in conflicting ways.

We resolve this mismatch by relaxing the fulfillment conditions for eventualities. Relaxed fulfillment arises naturally from a detailed analysis of the fragment-based model construction for CTL [Eme90]. By including in the definition of demo only those conditions that are required to verify the construction of models from demos, we obtain a notion of relaxed demo. The pruning system arising with relaxed demos generates refutations that can be translated in a natural way to derivations in the history-based Gentzen system.

Relaxed fulfillment adapts naturally to K^* and yields pruning refutations that can be translated to derivations of the Gentzen system for K^*. We show that pruning with respect to relaxed fulfillment yields the same demo as pruning with respect to normal fulfillment. This allows us to prove completeness of the Gentzen system for K^* without resorting to a fragment-based model construction.

1.3.2 Classical Models

We are developing the metatheory of classical modal logics in the context of constructive type theory. It should not come as a surprise that the mismatch between the constructive metatheory (i.e., the theory we do our proofs in) and the classical object logics (i.e., the logics we prove results for) will cause some complications.

Difficulties arise with soundness rather than completeness. Satisfaction of formulas is usually defined with respect to the class of *all* Kripke structures. This includes infinite structures for which it may not be decidable whether some formula holds at a particular state or not. The semantics with respect to all Kripke structures essentially corresponds to a shallow embedding of the object theory into the metatheory. Consequently, it is impossible to show soundness of classical proof systems with respect to this semantics. One way to sidestep these soundness issues would be to only consider finite models. Instead, we consider as models those Kripke structures for which the satisfaction relation is logically decidable, i.e., models $\mathcal{M}$ where $\mathcal{M}, w \vDash s \vee \mathcal{M}, w \nvDash s$ holds for all formulas s and states w of $\mathcal{M}$. We will refer to these models as *classical models*. Classical models include all finite models as well as some infinite ones. For K, we exhibit a (necessarily infinite) classical model satisfying all satisfiable formulas. One motivation for working with classical models is that for a classically minded reader the restriction to classical models is of no concern since every model is a classical model in the presence of excluded middle. Moreover, using classical models (rather than finite models) allows us to constructively prove small-model theorems without a priori assuming the finite-model property.

For our completeness results, the restriction to classical models is irrelevant since all the logics considered in this thesis have the small-model property. Therefore, we only need to construct finite models for our completeness proofs. However, this does not mean that completeness proofs for axiomatizations of these logics are automatically constructive. Some of the proofs for PDL [Ber79, HKT00] employ filtration which is not constructive. Also, the proofs of Kozen and Parikh [KP81] and Emerson and Halpern [EH85, Eme90] are non-constructive in that they assume logical decidability of Hilbert provability. While Hilbert provability for PDL and CTL is computationally (and hence also logically) decidable, the easiest way to show this is through completeness.

1.3.3 Inductive Interpretation of Eventualities

For CTL we employ an alternative semantics where eventualities are characterized inductively and invariants are characterized coinductively following the embedding [EC80, EL86] of CTL in the propositional μ-calculus [Koz83]. We favor the inductive semantics over the usual path semantics [Eme90] since infinite paths are difficult to work with in a constructive setting. We show that the inductive semantics constructively agrees with the path semantics on finite models. For infinite models, establishing the equivalence of the inductive and the path semantics requires excluded middle and a weak form of choice.

The inductive interpretation of eventualities and invariants gives rise to a natural Hilbert system for CTL. We characterize eventualities using axioms corresponding to the introduction rules of the inductive characterization and a rule corresponding to the induction scheme. Dually, invariants are characterized using axioms corresponding to inversion and a rule for coinduction.

The Hilbert system obtained this way allows for proofs that follow the mathematical intuitions fairly closely. This is convenient since we construct a fair amount of Hilbert proofs. In addition to the translation from pruning refutations to Hilbert refutations for the completeness proof, we also prove soundness of the Gentzen system using a translation to Hilbert proofs. Together with the translation from pruning refutations to Gentzen derivations, this yields an alternative completeness proof for the Hilbert system.

1.3.4 Formalization in Coq

All results presented in this thesis are formalized in the proof assistant Coq [Coq15] with the Ssreflect extension [GMT08]. In the context of this thesis *formal* always means machine-checked.

The mathematical development presented in this document and the accompanying Coq formalization [ACF] complement each other. The mathematical presentation is written using the language of type theory and with a particular emphasis on giving precise definitions. This allows us to maintain a close correspondence between the definitions used in this thesis and the definitions used in the formalization. This is important since the definitions underlying the formalization need to be understandable for the formalization to provide additional value. For the proofs themselves, we often just mention the main ideas since additional detail can always be found in the formalization. In particular, we largely ignore the technical details of the realization in Coq and only mention important design decisions.

One of the most technical parts of the formal development is the construction of models from demos for CTL. Based on the declarative model construction in [Eme90], we give a model construction for CTL that can be formalized with reasonable effort and is general enough to obtain the completeness results for the Hilbert system and the history-based Gentzen system as well as the usual upper-bound for the small-model property [EH85].

Another fairly technical part of the formal development is the generation of Hilbert proofs. In the literature [Fit07, BAPM83, EH85, Eme90] this is handled at a fairly informal level. For the formalization, we exploit that Hilbert systems for modal logics are modular, i.e., the Hilbert system for K extends a Hilbert system for propositional logic and is itself extended by the Hilbert systems for K* and CTL. We build a hierarchical library of over 100 modal logic facts that allows seamless reuse of facts established for subsystems (e.g., theorems of K in the development for CTL). In order to build Hilbert proofs, we use Coq's tactic language to provide goal management, rewriting, and assumption management for the construction of Hilbert derivations.

The formal development also makes extensive use of finite sets of formulas (e.g., clauses) and finite sets of finite sets of formulas (e.g., demos or histories). We develop a library for finite sets over countable types (e.g., formulas) and use it to formalize demos, pruning, and Gentzen systems. Beyond the usual operations (e.g., separation, powerset, and replacement) the library includes fixpoint operators for bounded monotone functions which we use to show decidability of inductive definitions over finite domains (e.g., fulfillment). Altogether the set library contains well over 150 lemmas.

1.3.5 Chapter Breakdown

Chapter 2 gives an overview of the logic of Coq and some of the constructions we use in our proofs.

Chapter 3 exemplifies the structure of our constructive completeness results through basic modal logic K.

Chapter 4 introduces inductive fulfillment to treat eventualities and presents the completeness proof for the Hilbert system for K^*.

Chapter 5 presents the construction of models from relaxed demos and the completeness proof for the Hilbert system for CTL.

Chapter 6 presents completeness proofs for history-based Gentzen systems for CTL and K^* by translating the pruning refutations arising with relaxed demos.

Chapter 7 presents the finite set library and the infrastructure for generating Hilbert refutations and gives an overview of the formalization.

Chapter 8 concludes the thesis with a discussion of the results and directions for future work.

1.4 Contributions

The main contributions of this thesis are as follows:

1. We are the first to prove small-model theorems and completeness of Hilbert systems for modal logics with eventualities in a proof assistant. Given the level of detail involved and the informal presentation in much of the original work [Pra79, BAPM83, EH85, Eme90], the gap between the original paper proofs and our formal proofs is considerable.
2. For each logic considered in this thesis, we introduce a notion of pruning refutation abstracting away the algorithmic details of pruning. Admissibility of the refutation rules yields a sufficient criterion for completeness and allows us to show completeness of Hilbert and Gentzen systems with one model construction per logic instead of one model construction per calculus.
3. We provide a detailed analysis of history-based Gentzen systems as introduced by Brünnler and Lange [BL08]. We handle the nondeterminism introduced by

the focusing mechanism using relaxed fulfillment conditions for eventualities. This allows us to give pruning-based completeness proofs for history-based Gentzen systems that are, at least in terms of the required formalization effort, simpler than the original completeness proof [BL08].

4. We prove all our results in the constructive logic of the proof assistant Coq without assuming global axioms. We provide a fine-grained analysis showing which results can be obtained constructively and where classical assumptions are necessary. For CTL we introduce an inductive semantics that is classically equivalent to the usual path semantics but avoids the need to reason about infinite paths and therefore works better in a constructive setting.
5. For the formal development we identify reoccurring patterns and turn them into reusable components where possible. This leads to the development of two reusable libraries.
   a) A library for finite sets over countable types featuring the usual set theoretic operations as well as fixpoint operators for bounded monotone functions.
   b) A hierarchical library for generating Hilbert derivations for propositional and modal logics providing for natural-deduction style assumption management and rewriting.

1.4.1 Published Results

Parts of this thesis extend and revise material that has already been published.

- A precursor to the notion of classical model employed throughout this thesis and a first constructive and formal decidability proof for K^* appears in:

  Christian Doczkal and Gert Smolka. Constructive formalization of hybrid logic with eventualities. In Zhong Shao Jean-Pierre Jouannaud, editor, *Certified Programs and Proofs (CPP 2011)*, volume 7086 of *LNCS*, pages 5–20. Springer, 2011.
- A first formal and constructive completeness proof for a Hilbert system for K^* (Chapter 4) and a “Gentzen system” whose rules are fairly close to the rules for pruning refutations for K^* appears in:

  Christian Doczkal and Gert Smolka. Constructive completeness for modal logic with transitive closure. In Chris Hawblitzel and Dale Miller, editors, *Certified Programs and Proofs (CPP 2012)*, volume 7679 of *LNCS*, pages 224–239. Springer, 2012.
- The finite set library (Chapter 7) and first formal and constructive completeness proofs for the Hilbert system and the history-based Gentzen system for CTL (Chapter 6) appears in:

  Christian Doczkal and Gert Smolka. Completeness and decidability results for CTL in Coq. In G. Klein and R. Gamboa, editors, *Interactive Theorem Proving (ITP 2014)*, volume 8558 of *LNCS (LNAI)*, pages 226–241. Springer, 2014.

- A formal proof of the small-model property of CTL and a revised version of the completeness result for the Hilbert system based on pruning (Chapter 5) appear in:

  Christian Doczkal and Gert Smolka. Completeness and decidability results for CTL in constructive type theory, 2015. Accepted for publication in *J. Autom. Reason.*

2 Type Theory Preliminaries

All results in this thesis have been formalized in the Coq interactive theorem prover Coq [Coq15] using the Ssreflect extension [GMT08]. In this chapter we briefly describe the type theory underlying Coq and some of the standard constructions used in our formal development. For a high-level introduction to type theory we refer to the fist chapter of the Homotopy Type Theory book [Uni13]. A gentle introduction to constructive type theory and interactive theorem proving with Coq can be found in [SB14].

2.1 The Type Theory of Coq

The proof assistant Coq implements a type theory called the predicative Calculus of (Co)Inductive Constructions (pCIC) [Coq15]. The logic pCIC can be seen as a combination of several extensions to the Calculus of Constructions [CH88]. It combines the infinite hierarchy of predicative type universes from the Extended Calculus of Constructions (ECC) [Luo89, Luo94] with the general inductive types of the Calculus of Inductive Constructions [Wer94] and coinductive types [Gim94].

The core of pCIC is a type theory with a universe Prop of propositions and an infinite hierarchy of type universes Type_i for $i \in \mathbb{N}$. The type theory is single sorted, i.e., there is only one syntactic category for terms and types. The term language can be defined roughly as follows:

$$M, N := \ x \mid \mathsf{Prop} \mid \mathsf{Type}_i \mid \forall x : M.N \mid \lambda x : M.N \mid M\,N$$

We write $M : N$ to denote that M has type N. The type $\forall x : M.N$ is a dependent function type. A function $f : (\forall x : M.N)$ returns for every argument $A : M$ some result of type $N[A/x]$, i.e., the result of substituting A for x in N. The usual non-dependent function type $M \to N$ is accommodated as the type $\forall x : M.N$ where x does not occur in N.

Every object in pCIC has a type. The type of a function type is always a universe. The universe Prop is to be seen as the universe of propositions or logical statements. Terms whose types are propositions are called proofs. This is often referred to as the propositions-as-types or Curry-Howard correspondence (see [Wad15] for a historical account).

In Coq the base theory can be extended with new inductive definitions at any time. Every inductive definition defines a type or a type constructor (e.g. list). For each inductive definition, the syntax is extended with the type constructor, a number of value constructors, a new binder $\mathsf{fix}\, f\ (x : M).\ s$, and a dependent match-construct. Together, fix and match allow for the definition of functions by (higher-order) primitive recursion.

Depending on the arity, we refer to inductively defined types in the universe Prop as inductive propositions, predicates, or relations. Types defined inductively in the universes Type_i are referred to as inductively defined data types. Examples are the type of boolean values $\mathbb{B} : \mathsf{Type}_0$ and the type of natural numbers $\mathbb{N} : \mathsf{Type}_0$. We remark that most of the usual data types have type Type_0.

The type theory of Coq can be seen both as a logic and as an effect-free terminating programming language. That is, the calculus comes with a built-in notion of β-reduction.[1]

There are three fundamental properties that are usually assumed about pCIC. These are:

- **Normalization:** If $N : M$ then one can compute some normal (i.e., irreducible) term N' such that $N \Rightarrow_\beta^* N'$.
- **Preservation:** If $N : M$ and $N \Rightarrow_\beta N'$, then $N' : M$.
- **Canonical Values:** If I is an inductive type and $M : I$ is closed and irreducible, then M is of the form $CM_0 \dots M_n$ where C is some constructor of I.

Taken together, normalization, preservation, and the canonical value property establish consistency of the type theory, i.e., the existence of unprovable propositions. Consider the inductive proposition $\bot$ (falsity) that has no constructor. The type behaves as expected, i.e., a case analysis on an assumption of type $\bot$ finishes any proof since there are no cases to consider. More importantly, $\bot$ has no closed proof. To see this, assume p is some closed proof of $\bot$. Then p has a normal form that still has type $\bot$ and starts with a constructor. This is clearly impossible.

We remark that the reasoning above only requires weak normalization, i.e. that every term can be normalized, and not that every reduction sequence eventually yields a normal form. Even though the reference manual [Coq15] claims that pCIC is strongly normalizing the implementation at the time of writing is not. An example is included in the formalization accompanying this thesis.

Normalization for terms with fix-expressions requires an elaborate guard condition ensuring that recursive calls are only made on structurally smaller arguments. Originally, this guard condition was justified by a translation to recursion schemes [Gim94]. However, the guard condition has been altered several times since then. Luo [Luo94] gives a normalization proof for ECC (essentially pCIC without inductive definitions). While there are normalization proofs for weaker type theories with guarded recursion [Nak00, AV14], there appears to be no normalization proof for the full system implemented by Coq.

For the rest of this thesis, we take a somewhat idealized view on Coq and assume that normalization, preservation and the canonical value property hold,

[1] The presentation of pCIC in the reference manual [Coq15] also features let-bindings and definitions which each come with their own β-like reduction rules.

$$\begin{aligned}
\mathsf{XM} &:= \forall P : \mathsf{Prop}.\ P \vee \neg P\\
\mathsf{IXM} &:= \forall P : \mathsf{Prop}.\ P + \neg P\\
\mathsf{DC} &:= \forall X : \mathsf{Type}\ \forall R : X \to X \to \mathsf{Prop}.\\
&\qquad (\forall x \exists y.\ R\, x\, y) \to \forall x \exists f : \mathbb{N} \to X.\ (f\,0 = x) \wedge (\forall n.\ R\,(f\,n)\,(f(n+1)))
\end{aligned}$$

Figure 2.1: Independent statements

even though this may not be true for the current implementation. We firmly believe that we do not accidentally exploit any inconsistency of the type theory or its implementation. Experience with previously discovered soundness bugs in Coq and other systems shows that fixing these bugs usually affects very few proofs.[2]

Besides the logical operations, which are all defined inductively in Coq, we employ a number of standard inductive types throughout this thesis. We write $M + N$ for the sum type (i.e., tagged disjoint union) of the types M and N. Its value constructors are $\mathsf{inl}\,A$ for $A : M$ and $\mathsf{inr}\,B$ for $B : N$. Further, we write $\Sigma x : M.\ N$ for the type of dependent pairs $\langle A, B\rangle$ where $A : M$ and $B : N[A/x]$. Coq also features dependent record types which can be seen as a generalization of nested Σ-types with primitive projections.

2.2 Independent Statements

There are a number of classical statements that are generally accepted as being independent of pCIC, i.e., they are not provable but can consistently be assumed. Since we are interested in constructive proofs, we do not assume any global axioms. Nevertheless, we sometimes use local classical assumptions to establish results that cannot be obtained constructively. Moreover, we will sometimes argue that certain classically provable statements are not provable constructively by showing that they entail some independent statement.

We will refer to the following independent statements: *excluded middle* (XM), *informative excluded middle* (IXM), and *dependent choice* [Her12] (DC). The precise statements are given in Figure 2.1.

The axiom of excluded middle is the defining feature of classical logic. Technically, it allows case distinctions on arbitrary propositions when constructing proofs. Informative excluded middle extends this to the construction of objects of arbitrary types. In particular, IXM allows arbitrary case distinctions in function definitions and therefore the definition of non-computable functions. The axiom

[2] The version of Coq employed here (i.e., version 8.4) is inconsistent with *propositional extensionality* (i.e., the assertion that equivalent propositions are equal). Coq 8.5 beta 2 reestablishes consistency with propositional extensionality through a change in the guard condition. This affects none of the proofs in our formalization.

of dependent choice is a weak choice principle. Interpreting the relation R as a graph without terminal nodes, the axiom DC provides for infinite paths starting at every node. Dependent choice will play a role for the path semantics of CTL (Chapter 5).

Consistency of all axioms relies on the assumption that pCIC has a proof-irrelevant set-theoretic model. Such a model would validate all axioms in Figure 2.1. Lee and Werner [LW11] give such a model for a calculus that is very similar to pCIC but uses judgmental equality instead of untyped conversion. It is generally assumed that the type theories with conversion or judgmental equality are equivalent, but so far this has only been proved for subsystems of pCIC [SH12].

2.3 Decidability

For constructive proofs, decidability plays an important role since decidable propositions behave classically. More precisely, case distinctions on decidable properties are always permitted.

We say that a property $P : \mathsf{Prop}$ is **decidable**, if there exists a boolean expression $p : \mathbb{B}$ such that $p = \mathsf{true} \leftrightarrow P$. Following the Ssreflect [GMT08] terminology, we call p a **boolean reflection** of P and say that p **decides** P. Similarly, we call a predicate $P : X \to \mathsf{Prop}$ decidable if there exists a predicate $p : X \to \mathbb{B}$ such that $p\,x$ decides $P\,x$ for all $x : X$. We refer to boolean predicates (i.e., functions of type $X \to \mathbb{B}$) as decidable predicates since they can be seen as their own boolean reflection. If a boolean p appears in the place of a proposition it is to be read as $p = \mathsf{true}$ and similarly for boolean predicates.

Calling functions into bool decision methods is justified by the three basic assumptions about pCIC in Section 2.1. That is, if p is some closed boolean reflection of some closed proposition P, then the normal form of p is computable and one of the two constructors true or false. Hence, the internal notion of decidability described above is a sufficient criterion for computational decidability. We remark that this argument depends on the canonical value property of pCIC which ceases to hold in the presence of axioms.

Many logical operations, e.g. the propositional logical operations and quantification over lists, have boolean reflections. For the mathematical presentation we will use the standard mathematical symbols, both for the respective proposition and its boolean reflection (e.g., we will write $\bot$ both for false and for the inductive proposition without constructor).

We remark that the notion of decidability employed here is a very shallow notion. In particular, it is not possible to prove any undecidability results. The reason for this is that pCIC is consistent with IXM, and IXM can be used to provide boolean reflections for arbitrary propositions. Hence, it is impossible to show that some proposition has no boolean reflection.

2.4 Type Classes and Canonical Structures

The type theory of Coq accommodates parametric polymorphism through functions from types to values. In addition to parametric polymorphism, Coq also implements overloading (i.e., ad-hoc polymorphism). Overloading allows certain operators and lemmas to be applied uniformly to a variety of types.

There are two implementations of overloading in Coq: canonical structures [Saï99, MT13] and type classes [SO08]. Canonical structures have been implemented in Coq long before type classes were added. However, they have been poorly documented for a long time and have not seen widespread use outside the Ssreflect libraries [GMR$^+$07, GGMR09].

Regardless whether one works with type classes or canonical structures, classes of types are represented as dependent record types that bundle types, operators over these types, and correctness properties. For example the class of types where equality is decidable corresponds to the following record type:

$$\mathsf{eqType} := \{\mathsf{sort} : \mathsf{Type}, \mathsf{eqb} : \mathsf{sort} \to \mathsf{sort} \to \mathbb{B}, \mathsf{eqP} : \forall x\, y. \mathsf{eqb}\, x\, y \leftrightarrow x = y\}$$

The projection $\mathsf{eqb} : \forall X : \mathsf{eqType}.\ \mathsf{sort}_X \to \mathsf{sort}_X \to \mathbb{B}$ serves as a generic boolean equality operator. Given some record $N : \mathsf{eqType}$ where sort_N is $\mathbb{N}$, the term $\mathsf{eqb}\, N\, 2\, 2$ is a boolean reflection of $2 = 2$. The record N (and other eqType records) can be registered with Coq and inferred automatically.

In the following we refer to types with a decidable equality (i.e, types with an associated eqType record) as **discrete types**. In addition to discrete types, the Ssreflect libraries define a number of other type classes. For our purposes, countable types and finite types are the most important.

A **countable type** is a discrete type that can be enumerated (e.g., numbers or formulas). For countable types, a choice function for decidable properties can be constructed. That is, for every countable type X there is a function

$$\mathsf{choose}_X : \forall p : X \to \mathbb{B}.\ (\exists x.\ p\, x) \to X$$

such that for every proof $E : (\exists x.\ p\, x)$ we have $p\, (\mathsf{choose}_X\, p\, E)$. Moreover, the result of $\mathsf{choose}_X\, p\, E$ does not depend on the proof E. In the mathematical presentation, we will suppress the proof argument E and abbreviate $\mathsf{choose}_X\, p\, E$ as $\varepsilon\, p$.

A **finite type** is a countable type whose elements can be given with a list. Many properties that are not decidable in general are decidable over finite types. In particular, quantification over finite types preserves decidability.

2.5 Finite Sets

For our completeness results, we make extensive use of finite sets over countable base types. For our purposes, finite sets are data types. In particular, we only consider finite sets with decidable membership.

The Ssreflect libraries [GMR$^+$07] provide finite sets but only over finite base types. For some parts of our development this is to restrictive, since we will work extensively with sets of formulas as well as sets of sets of formulas. For this we require a library providing finite sets over countable types providing the usual operations including separation $\{\, x \in A \mid p\, x \,\}$, replacement $\{\, f\, x \mid x \in A \,\}$, powerset 2^A, and a choice operator εA for non-empty sets. In the following, we write $\mathsf{set}\, X$ for the type of finite sets over a countable or finite type X. Note that we are working with a typed set theory. That is, if A has type $\mathsf{set}\, X$, then 2^A has type $\mathsf{set}(\mathsf{set}\, X)$. For this to be well-typed, $\mathsf{set}\, X$ has to be a countable type whenever X is a countable type.

Since we could not find a library satisfying all our needs, we developed our own. The type $\mathsf{set}\, X$ is realized as a constructive quotient of the type of lists over X. This yields an extensional representation, i.e., we have

$$X \subseteq Y \to Y \subseteq X \to X = Y \tag{2.1}$$

Extensionality ensures that set membership on all levels (sets, sets of sets, etc.) is just membership in the list representing the set. We defer the details of the construction to Chapter 7.

In addition to the standard set operations, we also define fixpoint operators for functions from finite sets to finite sets. We will use these fixpoint operators to show that certain inductive definitions over finite sets are decidable.

Let X be a countable type, let $U : \mathsf{set}\, X$, and let $F : \mathsf{set}\, X \to \mathsf{set}\, X$ be a function. We call F

- **monotone** if $\forall A\, B.\ A \subseteq B \to F\, A \subseteq F\, B$
- and **bounded by** U if $\forall A.\ A \subseteq U \to F\, A \subseteq U$

If F is monotone and bounded by U, its least fixpoint and its greatest fixpoint contained in U can be computed as follows [Cou81]:

$$\begin{aligned} \mathrm{lfp}\, F &:= F^{|U|}\, \emptyset \\ \mathrm{gfp}\, F &:= C_U(\mathrm{lfp}(C_U \circ F \circ C_U)) \end{aligned}$$

Here $F^{|U|}\, \emptyset$ is F iterated $|U|$-times on the empty set and C_U is the complement in U. We then have the usual fixpoint equations.

Lemma 2.5.1 Let $F : \mathsf{set}\, X \to \mathsf{set}\, X$ be monotone and bounded by U. Then $F(\mathrm{lfp}\, F) = \mathrm{lfp}\, F$ and $F(\mathrm{gfp}\, F) = \mathrm{gfp}\, F$

Moreover, the iterative construction of the fixpoint gives rise to the following "induction" principles:

Lemma 2.5.2 Let $P : \mathsf{set}\, X \to \mathsf{Prop}$ and let $F : \mathsf{set}\, X \to \mathsf{set}\, X$ be bounded by U.

1. $\forall P. P\, \emptyset \to (\forall A. P\, A \to P\, (F\, A)) \to P(\mathrm{lfp}\, F)$
2. $\forall P. P\, U \to (\forall A. P\, A \to P\, (F\, A)) \to P(\mathrm{gfp}\, F)$

We will make use of the fixpoint operators to show that certain inductive or coinductive predicates are decidable over finite domains.

3 Basic Modal Logic

In this chapter we develop a constructive metatheory of basic modal logic K. We give soundness and completeness proofs for a Hilbert system and a Gentzen system and construct a model satisfying all satisfiable formulas.

Our completeness results are based on a decision procedure for satisfiability. The decision procedure employs pruning [Pra79, KSS11]. Pruning starts with the finite collection of clauses (i.e., finite sets of formulas) over some finite subformula universe and removes unsatisfiable clauses until only satisfiable clauses remain. We complement pruning with a refutation calculus designed such that it can derive all clauses removed during pruning. This yields a sufficient criterion for completeness: a proof system is complete if pruning refutations can be translated to refutations in the respective system. We give translations from pruning refutations to Hilbert refutations and to derivations in the Gentzen system. The completeness results for both systems then come in the form of informative decision methods, i.e., functions returning for every input formula s either a finite model satisfying s or a proof of $\neg s$. The usual completeness result (i.e., provability of all valid formulas) follows as a corollary. Together with soundness, we obtain the small-model property and the decidability of satisfiability and provability as additional corollaries.

Soundness of the Hilbert system and the Gentzen system is shown with respect to a class of models we call classical models. Classical models internalize the instances of excluded middle needed to give constructive soundness proofs. In the presence of excluded middle, classical models are equivalent to Kripke models [Kri63]. Constructively, the internalized assumptions can always be fulfilled for finite models. Moreover, we construct a (necessarily infinite) universal classical model satisfying all satisfiable formulas.

Pruning [Pra79] was originally developed to establish the EXPTIME decidability of the satisfiability problem for PDL [FL79]. The technique has been adapted to a variety of other modal logics [EH85, KSS11]. By using pruning for K, we obtain proofs that extend in a uniform way to modal logic with transitive closure (Chapter 4) and CTL [EC82] (Chapter 5). For K all proofs are fairly simple. This allows us to focus on the design decisions underlying the formalization in Coq.

$$\begin{array}{rl}
\mathsf{S} & \vdash (u \to s \to t) \to (u \to s) \to u \to t \\
\mathsf{K} & \vdash s \to t \to s \\
\mathsf{DN} & \vdash ((s \to \bot) \to \bot) \to s \\
\mathsf{N} & \vdash \Box(s \to t) \to \Box s \to \Box t
\end{array}$$

$$\frac{\vdash s \qquad \vdash s \to t}{\vdash t}\ \mathsf{MP} \qquad\qquad \frac{\vdash s}{\vdash \Box s}\ \mathsf{Nec}$$

Figure 3.1: Hilbert system for K

3.1 Hilbert System and Models

We fix a countably infinite alphabet $\mathcal{A}$ of atomic propositions p and define the **formulas** of K as follows:

$$s, t \ := p \mid \bot \mid s \to t \mid \Box s$$

We choose to work with a minimal set of logical operations for technical convenience. We make use of the following abbreviations for formulas:

$$\begin{array}{rlrl}
\neg s & := s \to \bot & s \leftrightarrow t & := (s \to t) \wedge (t \to s) \\
s \vee t & := \neg s \to t & \Diamond s & := \neg \Box \neg s \\
s \wedge t & := \neg(s \to \neg t) & &
\end{array}$$

We axiomatize provability of formulas with a Hilbert system. The rules and axioms are given Figure 3.1. The system consists of a minimal Hilbert system for classical propositional logic (S, K, DN, and MP) extended with the *normality scheme* (N) and the *necessitation rule* (Nec). The Hilbert system in Figure 3.1 corresponds to the system in [Fit07] with an explicit choice for the propositional axioms. If $\vdash \neg s$, we say that s is (Hilbert) **refutable** and call the proof of $\neg s$ a (Hilbert) **refutation** of s.

We now define models for K. Since we are working in a constructive setting we have to be careful as to what is the right class of models to consider.

In classical mathematics (i.e., set theory), modal logic is interpreted over transition systems, also called Kripke structures [Kri63], where the states are labeled with atomic propositions. A **model** $\mathcal{M}$ for K consists of the following components:

- A set $|\mathcal{M}|$ of **states**.
- A **transition relation** $\Rightarrow_{\mathcal{M}} \subseteq |\mathcal{M}| \times |\mathcal{M}|$.
- A **labeling relation** $\Lambda_{\mathcal{M}} \subseteq \mathcal{A} \times |\mathcal{M}|$.

Let $\mathcal{M}$ be a model. If $w \Rightarrow_{\mathcal{M}} v$ for states w and v of $\mathcal{M}$, we call v a **successor** of w. The **satisfaction relation** $w \vDash s$ between states w of $\mathcal{M}$, and formulas s is defined by recursion on formulas:

$$\begin{aligned} w \vDash p &:= \Lambda_{\mathcal{M}}\, p\, w \\ w \vDash \bot &:= \bot \\ w \vDash s \to t &:= w \vDash s \to w \vDash t \\ w \vDash \Box s &:= \forall v \in \mathcal{M}.\ (w \Rightarrow_{\mathcal{M}} v) \to v \vDash w \end{aligned}$$

If the model $\mathcal{M}$ cannot be determined from the state w, we also write $\mathcal{M}, w \vDash s$ to disambiguate. A formula s is **satisfied** by $\mathcal{M}$ if $w \vDash s$ for some state w of $\mathcal{M}$.

A formula is **satisfiable** if it is satisfied by some model and **valid** if it is satisfied at every state of every model. Some authors require $|\mathcal{M}|$ to be nonempty (cf. [BdRV01]). Admitting the empty model does not change the notions of satisfiability and validity. However, it allows us to drop some side conditions and avoid certain corner cases.

In Coq, we represent models as dependent records comprised of a type of states, a transition relation and a labeling:

$$\{|\mathcal{M}| : \mathsf{Type}, \Rightarrow_{\mathcal{M}} : |\mathcal{M}| \to |\mathcal{M}| \to \mathsf{Prop}, \Lambda_{\mathcal{M}} : \mathcal{A} \to |\mathcal{M}| \to \mathsf{Prop}\}$$

We will use the letter $\mathcal{M}$ to refer both to the model as a whole as well as the underlying type of states. Further, we write $w \in \mathcal{M}$ if w is a state of $\mathcal{M}$.

3.2 Soundness and Classical Models

The satisfaction relation together with the representation of models essentially yields a shallow embedding of K into Coq. Consequently, this representation does not provide a reasonable semantics for classical modal logic in a constructive setting. In fact, soundness of the Hilbert system for the class of all models is equivalent to excluded middle. We introduce a class of classical models that internalizes enough instances of excluded middle to allow for constructive soundness proofs. Classical models are designed such that all finite models are classical models.

Fact 3.2.1 If $\neg\neg p \to p$ is valid, then XM is provable.

Proof Assume $\neg\neg p \to p$ is valid. To show XM, it suffices to show $\neg\neg P \to P$ for some arbitrary proposition $P : \mathsf{Prop}$. We define a model $\mathcal{M}$ with a single state w_0 where $\Lambda\, p\, w_0 := P$. By assumption we have $w_0 \vDash \neg\neg p \to p$ which is, by definition, equal to $\neg\neg P \to P$. ∎

We clearly have $\vdash \neg\neg p \to p$. Since XM is not provable constructively, soundness for the class of all models is also not provable constructively.

Since we cannot prove soundness with respect to all models, we refine the class of general models to a class of models we call classical models.

Definition 3.2.2 A **classical model** is a model $\mathcal{M}$ for which the satisfaction relation is logically decidable, i.e., a model $\mathcal{M}$ such that

$$\forall w \in \mathcal{M}\ \forall s.\ w \vDash s \vee w \nvDash s \tag{3.1}$$

If one assumes XM, then every model can be turned into a classical model. Without XM, having to prove (3.1) severely restricts our ability to construct infinite models. Note that the notion of a classical model refers to the satisfaction relation and therefore depends on the logic under consideration.

It is straightforward to show that the Hilbert system is sound for classical models.

Theorem 3.2.3 (Soundness) Let $\vdash s$. Then $\mathcal{M}, w \vDash s$ for all classical models $\mathcal{M}$ and all states w of $\mathcal{M}$.

Proof By induction on $\vdash s$ using (3.1) for the case corresponding to DN. ∎

Finite models are a particularly important class of classical models. For finite models the satisfaction relation is computationally (and hence also logically) decidable.

Definition 3.2.4 A **finite model** is a model where the type of states is a finite type and the transition relation and the labeling relation are decidable.

Lemma 3.2.5 Every finite model is a classical model.

Proof It suffices to show that the satisfaction relation is decidable on finite models. This is the case since the labeling and the transition relation are decidable and all operations in the definition of the satisfaction relation preserve decidability. ∎

Remark 3.2.6 The one-state model used in the proof of Fact 3.2.1 is not a finite model according to our definition. The labeling $\Lambda_{\mathcal{M}}\, p\, w := P$ employs the abstract proposition P which is not decidable.

Classical models can be understood in two ways. On the one hand, classical models internalize the classical assumptions required for soundness into the model. This allows for constructive soundness and completeness proofs without assuming any axioms. On the other hand, classical models can be seen as an abstraction that allows establishing two soundness results with one proof. A constructive soundness proof for classical models yields a constructive soundness result for finite models and also establishes soundness for all models in contexts with excluded middle.

All logics considered in this thesis have the small model property and we only construct finite models for our completeness results. Hence, for the completeness results a finite model semantics would be sufficient. The use of classical models ensures that all our remaining results (e.g., soundness) extend to infinite models if excluded middle is assumed. Moreover, classical models are, even constructively, strictly more general than finite models. We construct a (necessarily infinite) classical model satisfying all satisfiable formulas of K at the end of this chapter.

We adopt the following convention:

Convention Unless stated otherwise, all semantic notions (model, satisfiability, validity, soundness, etc.) will always refer to the class of classical models for the respective logic. We will speak of **general models** if we explicitly do not assume logical decidability of the satisfaction relation.

3.3 Demos

We now develop the pruning system underlying our completeness results. The central notion underlying pruning is that of a demo [KSS11]. Demos are a class of finite models.

3.3.1 Clauses and Support

For our demos we use literal clauses and the notion of support [KS10, KS14] instead of the more traditional notion of Hintikka sets [Pra79, EH85]. We use signed formulas [Smu68] to express top-level negations. Signs are a formal device that leads to a simple definition of subformula universes (Section 3.4) based on our minimal syntax.

A **signed formula** is either s^+ or s^- where s is a formula. Signs bind weaker than formula constructors, so $\Box s^+$ is to be read as $(\Box s)^+$. We write σ for arbitrary signs and $\overline{\sigma}$ for the sign opposite to σ. A state satisfies a signed formula s^σ if it satisfies $\lfloor s^\sigma \rfloor$ where $\lfloor s^+ \rfloor = s$ and $\lfloor s^- \rfloor = \neg s$. Hence, negative signs can be thought of as top-level negations. In particular, we have the following equivalence:

$$\lfloor \Box s^- \rfloor \leftrightarrow \Diamond \lfloor s^- \rfloor$$

That is, negative boxes are treated as diamonds (with a negation applied to the body).

A **clause** is a finite set of signed formulas. Capital letters $C, D, \ldots$ range over clauses. A state satisfies a clause if it satisfies all its members. We abbreviate $C \cup \{s^\sigma\}$ as C, s^σ. The letters $\mathcal{S}, \mathcal{T}, \ldots$ range over finite sets of clauses.

A signed formula is a **literal** if it is of the form p^σ, $\bot^\sigma$, or $\Box s^\sigma$. A **literal clause** is a clause containing only literals. A literal clause is **locally consistent** if it contains neither $\bot^+$ nor both p^+ and p^- for any p. We refer to locally consistent literal clauses as **base clauses**.

The **support relation** $C \rhd s^\sigma$ between literal clauses C and signed formulas s^σ is defined recursively:

$$\begin{aligned} C \rhd s^\sigma &:= s^\sigma \in C && \text{if } s^\sigma \text{ is a literal} \\ C \rhd (s \to t)^+ &:= C \rhd s^- \ \vee \ C \rhd t^+ \\ C \rhd (s \to t)^- &:= C \rhd s^+ \ \wedge \ C \rhd t^- \end{aligned}$$

The support relation can be seen as a restricted form of entailment, i.e., a state satisfying a clause C also satisfies all formulas supported by C. We extend the support relation to a relation between clauses by defining

$$C \rhd D := \forall s^\sigma \in D. C \rhd s^\sigma$$

Moreover, if S is a set of base clauses, we define

$$S \rhd C := \exists D \in S. D \rhd C$$

Note that all three variants of the support relation (i.e., $C \rhd s^\sigma$, $C \rhd D$, and $S \rhd C$) are decidable since the definitions employ only structural recursion, membership in finite sets, and bounded quantification.

Lemma 3.3.1 Let C and D be literal clauses. Then $C \rhd D$ iff $D \subseteq C$.

3.3.2 Model Existence

A demo will be a finite set of base clauses S that can be seen as a model where the states are the clauses from S and every state satisfies all formulas it contains.

The **request** $\mathcal{R}\, C$ of a clause C is defined as follows:

$$\mathcal{R}\, C := \{\, s^+ \mid \Box s^+ \in C \,\}$$

For a state w to satisfy the positive box formulas in a clause C, every successor of w must satisfy the formulas in $\mathcal{R}\, C$.

Definition 3.3.2 A set of base clauses S is a **demo** if it satisfies:

(D$\Box$) If $\Box s^- \in C \in S$, then $S \rhd \mathcal{R}\, C, s^-$.

Every demo can be seen as a model. Let S be a demo. We define a finite model $\mathcal{M}_S$ where[1]

$$\begin{aligned} |\mathcal{M}_S| &:= S \\ C \Rightarrow_{\mathcal{M}_S} D &:= D \rhd \mathcal{R}\, C \\ \Lambda_{\mathcal{M}_S} p\, C &:= p^+ \in C \end{aligned}$$

[1] When appearing as a type, the set S is to be read as the finite type $(\Sigma C : \mathsf{clause}.\ C \in S)$ whose elements are in one-to-one correspondence to the elements of S.

Lemma 3.3.3 (Model Existence) Let $\mathcal{S}$ be a demo and $C \in \mathcal{S}$. Then $\mathcal{M}_{\mathcal{S}}, C \vDash t^{\sigma}$ whenever $C \rhd t^{\sigma}$.

Proof By induction on t. Let $C \in \mathcal{S}$. We consider the case where $t = \Box s^{-} \in C$. We need so show $C \vDash \Box s^{-}$. By induction hypothesis, it suffices to show $D \rhd s^{-}$ for some clause $D \in \mathcal{S}$ such that $C \Rightarrow_{\mathcal{M}_{\mathcal{S}}} D$. Since $\mathcal{S}$ is a demo, such a clause must exist. All other cases are straightforward. ∎

3.4 Pruning and Refutation Calculus

We now present the decision method for satisfiability underlying our completeness proofs. The method relies on a finiteness property we call subformula property. Given a clause C_0, we consider the finite set $\mathcal{S}$ of clauses containing only subformulas of formulas in C_0.

We then construct the canonical demo $\mathcal{D}$ containing exactly the satisfiable base clauses from $\mathcal{S}$. We compute $\mathcal{D}$ using pruning [Pra79, KSS11]. Pruning starts with the set of all base clauses in $\mathcal{S}$ and successively removes clauses violating the demo condition. This process only removes unsatisfiable clauses and terminates with the demo $\mathcal{D}$.

We complement pruning with a refutation calculus that derives unsatisfiable clauses. The calculus is designed such that it can derive all clauses from $\mathcal{S}$ that are not supported by $\mathcal{D}$. Since all clauses supported by $\mathcal{D}$ are satisfiable (Lemma 3.3.3), the refutation calculus is complete in that it can refute all unsatisfiable clauses from $\mathcal{S}$. Consequently, we can show completeness of other deductive systems (e.g, the Hilbert system) by translating derivations from the refutation calculus into the given calculus.

3.4.1 Subformula Universes

Let U be a finite set of signed formulas (i.e., a clause). We refer to U as a **subformula universe**, if it satisfies the following closure conditions:

S1. If $(s \to t)^{\sigma} \in U$, then $\{s^{\overline{\sigma}}, t^{\sigma}\} \subseteq U$.

S2. If $\Box s^{\sigma} \in U$, then $s^{\sigma} \in U$.

If U is a subformula universe, we refer to 2^U as as the **clause universe** over U. In the following U will always denote a subformula universe.

For every clause C there exists a smallest subformula universe extending C. We write $\mathsf{sfc}\,C$ for this clause and refer to it as the **subformula closure** of C. The clause $\mathsf{sfc}\,C$ can easily be computed.

We write $|s|$ for the size (i.e., the number of constructors) of the formula s.

Lemma 3.4.1 $|\mathsf{sfc}\{s^{\sigma}\}| \leq |s|$.

We fix some subformula universe U for the rest of this section. We write $\overline{U}$ for the set of base clauses over U:

$$\overline{U} := \{\, C \subseteq U \mid C \text{ base clause} \,\}$$

The notions of subformula universe and support are designed such that the satisfiability problem for the clauses in 2^U reduces to the satisfiability problem for the clauses in $\overline{U}$:

Proposition 3.4.2 Let $C \subseteq U$. Then C is satisfiable iff there exists some satisfiable clause $D \in \overline{U}$ such that $D \rhd C$.

We do not make explicit use of this proposition and postpone its proof (cf. Fact 3.5.12).

3.4.2 Pruning

It turns out that the set containing exactly the satisfiable clauses from $\overline{U}$ constitutes a demo. We refer to this demo as the **canonical demo for U**. We use pruning to construct the canonical demo. Starting from $\overline{U}$ we successively remove clauses violating (D□) until we are left with a demo.

Since we employ pruning for several of our results, we develop pruning in an abstract setting. Let T be a type with a choice operator. We define the pruning function $\mathsf{prune} : (T \to \mathsf{set}\, T \to \mathbb{B}) \to \mathsf{set}\, T \to \mathsf{set}\, T$ recursively[2]:

$$\mathsf{prune}\, p\, X := \begin{cases} X & \text{if } p\, x\, X \text{ for all } x \in X \\ \mathsf{prune}\, p\, (X \setminus \{\varepsilon\{\, x \in X \mid \neg p\, x\, X \,\}\}) & \text{otherwise} \end{cases}$$

Note that the set $\{\, x \in X \mid \neg p\, x\, X \,\}$ is non-empty whenever the second branch is chosen. Hence, we can use the ε-operator to pick some element from the set.

Let $p : T \to \mathsf{set}\, T \to \mathbb{B}$ be a decidable predicate. We say that a set X is **p-consistent** if $p\, x\, X$ holds for all $x \in X$. Pruning with p always terminates with a p-consistent subset of the input. Further, the recursive construction gives rise to an induction principle:

Lemma 3.4.3 Let $X : \mathsf{set}\, T$, $p : T \to \mathsf{set}\, T \to \mathbb{B}$, and let $P : \mathsf{set}\, T \to \mathsf{Prop}$.

1. $\mathsf{prune}\, p\, X \subseteq X$
2. $\mathsf{prune}\, p\, X$ is p-consistent.
3. $P\, X \to (\forall y Y.\; y \in Y \subseteq X \to P\, Y \to \neg p\, y\, Y \to P(Y \setminus \{y\})) \to P(\mathsf{prune}\, p\, X)$.

Proof Claims (2) and (3) follow by induction on $|X|$. Claim (1) follows with (3). ∎

[2] Coq's Function command provides a facility to define functions by well-founded recursion. For prune, we employ the termination measure $\lambda X. |X|$.

Remark 3.4.4 Since we do not require p to satisfy any monotonicity properties, the result of pruning may depend on the choice function for T. We will only use pruning to construct the demo of satisfiable base clauses for a given subformula universe. In this case, the result does not depend on the choice function.

We compute the canonical demo for U as follows:

$$\begin{aligned} p\,C\,\mathcal{S} &:= \forall \Box s^- \in C.\ \mathcal{S} \triangleright \mathcal{R}\,C, s^- \\ \mathcal{D}(U) &:= \mathsf{prune}\ p\ \overline{U} \end{aligned}$$

Lemma 3.4.5

1. $\mathcal{D}(U) \subseteq \overline{U}$
2. $\mathcal{D}(U)$ is a demo.

Proof Immediate with Lemma 3.4.3. ∎

3.4.3 Pruning Refutations

We have already established that $\mathcal{D}(U)$ supports only satisfiable formulas (Lemmas 3.4.5 and 3.3.3). In order to prove completeness of the Hilbert system it remains to show that the Hilbert system can refute all formulas that are not supported by $\mathcal{D}(U)$. We complement pruning with a refutation calculus that can derive all clauses not supported by $\mathcal{D}(U)$. Completeness of the Hilbert system then follows by translating these pruning refutations to Hilbert refutations.

We inductively define a refutation calculus for U using the rules:

$$\frac{C \subseteq U \qquad \mathcal{S} \not\triangleright C \qquad \mathsf{coref}\,\mathcal{S}}{\mathsf{ref}_U\, C}\ \mathsf{supp} \qquad\qquad \frac{\Box s^- \in C \qquad \mathsf{ref}_U(\mathcal{R}\,C, s^-)}{\mathsf{ref}_U\, C}\ \mathsf{jump}$$

where $\mathsf{coref}\,\mathcal{S} := \forall C \in \overline{U}.\ C \notin \mathcal{S} \to \mathsf{ref}_U\, C$. We refer to the rules as **support rule** and **jump rule**. The refutation calculus abstracts from the algorithmic details of pruning. Consequently, we will refer to derivations of the refutation calculus as **pruning refutations**.

Proposition 3.4.6 (Refutation Soundness) If $\mathsf{ref}_U\, C$, then C is unsatisfiable.

We defer the proof of this proposition to the next section (cf. Lemma 3.5.6) where we will translate pruning refutations to Hilbert refutations. Soundness of the refutation calculus then follows with soundness of the Hilbert system.

Completeness of the refutation calculus follows since pruning can be seen as a particular strategy for generating refutations for base clauses.

Lemma 3.4.7

1. $\mathsf{coref}\,\mathcal{D}(U)$
2. $\mathsf{ref}_U\, C$ whenever $C \subseteq U$ and $\mathcal{D}(U) \not\triangleright C$.

Proof *Claim 1.* We show $\mathsf{coref}\,\mathcal{D}(U)$ by induction on the construction of $\mathcal{D}(U)$ (Lemma 3.4.3(3)). We clearly have $\mathsf{coref}\,\overline{U}$. Now let $S \subseteq \overline{U}$ such that $\mathsf{coref}\,S$, let $C \in S$, and assume $\neg p\,C\,S$. We need to show $\mathsf{ref}_U\,C$. By assumption, there exists some $\Box s^- \in C$ such that $S \ntriangleright \mathcal{R}\,C, s^-$. Thus, we have $\mathsf{ref}_U\,(\mathcal{R}\,C, s^-)$ (supp-rule) and hence $\mathsf{ref}_U\,C$ (jump-rule) as required.

Claim 2. Immediate with Claim (1) and the supp-rule. ∎

Remark 3.4.8 An inspection of the proof of Lemma 3.4.7 shows that we can restrict instances of S in the supp-rule to subsets of $\overline{U}$ and instances of C in the jump-rule to elements of $\overline{U}$ without losing completeness.

Theorem 3.4.9 (Refutation Completeness) Let U be a subformula universe and let $C \subseteq U$. Then either $\mathsf{ref}_U\,C$ or C is satisfied by a model with at most $2^{|U|}$ states.

Proof If $\mathcal{D}(U) \triangleright C$ then $\mathcal{M}_{\mathcal{D}(U)}$ is a model with at most $2^{|U|}$ states satisfying C (Lemma 3.4.5 and Lemma 3.3.3). Otherwise, we have $\mathsf{ref}_U\,C$ by Lemma 3.4.7. ∎

3.5 Completeness of the Hilbert System

We now prove completeness of the Hilbert system in Figure 3.1. For this it suffices to translate pruning refutations to Hilbert refutations (Theorem 3.4.9).

In addition to the abbreviations defined in Section 3.1, we define "big" conjunctions and disjunctions indexed by lists. We use big conjunctions to reason about clauses inside the Hilbert system. If C is a clause, we refer to $\bigwedge_{s^\sigma \in C} \lfloor s^\sigma \rfloor$ as its **associated formula**.[3] When a clause occurs in the place of a formula, it is to be read as its associated formula.

Fact 3.5.1 Let $\mathcal{M}$ be a classical model and let $w \in M$. Then w satisfies the associated formula of a clause C iff w satisfies every signed formula in C.

We fix some subformula universe U. We call a set of clauses S **Hilbert corefutable** if we have $\vdash \neg C$ for all $C \in \overline{U} \setminus S$. We now translate derivations of $\mathsf{ref}_U\,C$ to Hilbert refutations of C. The translation to Hilbert refutations is compositional, i.e., we can show that the rules of ref_U are admissible for the predicate $\lambda C. \vdash \neg C$.

We start by showing admissibility of the support rule. For this, we show that if S is Hilbert corefutable, then every clause $C \subseteq U$ implies the disjunction of all clauses in S that support C. We define the **base of C in S** as

$$\mathcal{B}_S\,C := \{\, D \in S \mid D \triangleright C \,\}$$

If A is a set of clauses, we abbreviate $\bigvee_{C \in A} C$ as $\bigvee A$.

[3] We convert finite sets to lists as required.

Lemma 3.5.2 Let $\mathcal{S}$ be Hilbert corefutable and let $C \subseteq U$. Then $\vdash C \to \bigvee \mathcal{B}_{\mathcal{S}}\, C$.

Proof Let $\mathcal{T} := \{\, D \subseteq U \mid D \text{ literal} \,\}$. By propositional reasoning, it suffices to show $\vdash C \to \mathcal{B}_{\mathcal{T}} C$. We proceed by induction on the sum of the sizes of the non-literal formulas in C. If C contains only literals, the claim is trivial. So assume there exists some non-literal signed formula $u \in C$. We consider the case where $u = s \to t^+$. We have

$$\vdash C \to (C \setminus \{u\}), s^- \lor (C \setminus \{u\}), t^+$$

by propositional reasoning. By induction hypothesis this yields

$$\vdash C \to \bigvee \mathcal{B}_{\mathcal{T}}((C \setminus \{u\}), s^-) \lor \bigvee \mathcal{B}_{\mathcal{T}}((C \setminus \{u\}), t^+)$$

The claim then follows since $\mathcal{B}_{\mathcal{T}}\,((C \setminus \{u\}), s^-) \cup \mathcal{B}_{\mathcal{T}}((C \setminus \{u\}), t^+) \subseteq \mathcal{B}_{\mathcal{T}}\, C$. The case for $u = s \to t^-$ is similar. ■

Lemma 3.5.3 (Admissibility of Support Rule) Let $\mathcal{S}$ be Hilbert corefutable and let $C \subseteq U$. Then $\vdash \neg C$ whenever $\mathcal{S} \not\rhd C$.

Proof Follows immediately with Lemma 3.5.2. ■

To show admissibility of the jump rule, we need a number of facts about the Hilbert system.

Lemma 3.5.4

1. $\vdash \Box s \to \Box t$ and $\vdash \Diamond s \to \Diamond t$ whenever $\vdash s \to t$.
2. $\vdash \Box(s \land t) \leftrightarrow \Box s \land \Box t$
3. $\vdash C \to \Box(\mathcal{R}\, C)$
4. $\vdash \neg\Diamond\bot$
5. $\vdash \Box s \to \Diamond t \to \Diamond(s \land t)$

Proof Claim (1) follows with N and Nec. Claims (2), (4), and (5) follow with (1) and propositional reasoning. For Claim (3) let $D := \{\, \Box s^+ \mid s^+ \in \mathcal{R}\, C \,\}$. The claim follows since $\vdash D \leftrightarrow \Box(\mathcal{R}\, C)$ by (2) and $\vdash C \to D$ since $D \subseteq C$. ■

Lemma 3.5.5 (Admissibility of Jump Rule) Let C be a clause and let $\Box s^- \in C$. Then $\vdash \neg C$ if $\vdash \neg\mathcal{R}\, C, s^-$.

Proof Assume $\vdash \neg\mathcal{R}\, C, s^-$. It suffices to show $\vdash \neg C, \Box s^-$.

$$\begin{array}{lll} & \vdash C, \Box s^- \to \bot & \text{def. } \neg \\ \Leftarrow & \vdash C, \Box s^- \to \Diamond(\mathcal{R}\, C, s^-) & \text{Lemma 3.5.4(4), assumption.} \\ \Leftarrow & \vdash C \land \Diamond\neg s \to \Diamond(\mathcal{R}\, C \land \neg s) & \text{prop. reasoning.} \\ \Leftarrow & \vdash \Box(\mathcal{R}\, C) \land \Diamond\neg s \to \Diamond(\mathcal{R}\, C \land \neg s) & \text{Lemma 3.5.4(3)} \end{array}$$

The last claim follows with Lemma 3.5.4(5). ■

Lemma 3.5.6 $\vdash \neg C$ whenever $\mathsf{ref}_U\, C$.

Proof By induction on $\mathsf{ref}_U\, C$ using the lemmas above. ■

Together with soundness of the Hilbert system, Lemma 3.5.6 establishes the soundness of pruning refutations (Proposition 3.4.6). Moreover, we obtain completeness of the Hilbert system.

Theorem 3.5.7 (Informative Completeness) Let s be a formula. Then either $\vdash \neg s$ or s is satisfied by a model with at most $2^{|s|}$ states.

Proof Let $U := \mathsf{sfc}\{s^+\}$. By Theorem 3.4.9 we either have $\mathsf{ref}_U\,\{s^+\}$ and hence $\vdash \neg s$ by Lemma 3.5.6 or s is satisfied by a model with at most $2^{|s|}$ states (Lemma 3.4.1). ■

Remark 3.5.8 In Coq, Theorem 3.5.7 is represented as a function of type

$$\forall s.(\vdash \neg s) + (\Sigma \mathcal{M}.\Sigma w : \mathcal{M}.\ |\mathcal{M}| \leq 2^{|s|} \wedge w \vDash s)$$

Here, $|\mathcal{M}|$ is the size of the type underlying $\mathcal{M}$. This is only defined for finite models which is sufficient for our purposes.

Corollary 3.5.9 (Completeness) If s is valid, then $\vdash s$.

Corollary 3.5.10 (Decidability) Satisfiability, validity, and Hilbert provability of formulas are decidable.

Proof Follows since by soundness (Theorem 3.2.3) the two alternatives of Theorem 3.5.7 are mutually exclusive. ■

Corollary 3.5.11 (Small Models) If s is satisfiable, then s is satisfied by a model with at most $2^{|s|}$ states.

Fact 3.5.12 Let U be a subformula universe and let $C \subseteq U$ be a clause. Then

1. $\mathcal{D}(U) \rhd C$ iff C is satisfiable.
2. $C \in \mathcal{D}(U)$ iff C is a satisfiable base clause.

This establishes Proposition 3.4.2 and the fact that $\mathcal{D}(U)$ is indeed the canonical demo for U.

3.6 Gentzen System

Building on the proofs developed in the previous sections, we now prove soundness and completeness of a Gentzen system for K. The Gentzen system is essentially a clausal presentation of Fitting's "destructive tableau" for K [Fit07]. We will later extend this system to Gentzen systems for K* and CTL (Chapter 6).

The rules of the calculus are given in Figure 3.2. The system derives unsatisfiable clauses.

$$\frac{}{\Vdash C, p^-, p^+}\,\mathsf{A} \qquad \frac{}{\Vdash C, \bot^+}\,\mathsf{F}^+ \qquad \frac{\Vdash C, s^- \qquad C, t^+}{\Vdash C, s \to t^+}\,\mathsf{I}^+ \qquad \frac{\Vdash C, s^+, t^-}{\Vdash C, s \to t^-}\,\mathsf{I}^-$$

$$\frac{\Vdash \mathcal{R}C, s^-}{\Vdash C, \Box s^-}\,\mathsf{X}$$

Figure 3.2: Gentzen system for K

Theorem 3.6.1 (Soundness) If $\Vdash C$, then C is unsatisfiable.

To show completeness of the Gentzen system, it suffices to show that the rules of the refutation calculus are admissible for $\Vdash$. For the support rule, we need the following lemma.

Lemma 3.6.2 Let U be a subformula universe and let S such that $\Vdash D$ for all $D \in \overline{U} \setminus S$. Then $\Vdash C$ whenever $C \subseteq U$ and $\Vdash D$ for all $D \in \mathcal{B}_S C$.

Proof Similar to the proof of Lemma 3.5.2. ∎

Lemma 3.6.3 $\Vdash C$ whenever $\mathsf{ref}_U\, C$ for some subformula universe U.

Proof Admissibility of the support rule follows with Lemma 3.6.2. Admissibility of the jump rule is obvious. ∎

Theorem 3.6.4 (Completeness) Let C be a clause. Then either $\Vdash C$ or C is satisfied by a finite model.

Proof Follows with Lemma 3.6.3 and Theorem 3.4.9. ∎

Corollary 3.6.5 Gentzen derivability is decidable.

Proof Similar to the proof of Corollary 3.5.10 ∎

Remark 3.6.6 Theorem 3.6.1 can easily be strengthened to show $\vdash \neg C$ whenever $\Vdash C$ for some clause C. Together with Theorem 3.6.4, this provides an alternative completeness proof for the Hilbert system.

Remark 3.6.7 Normally, one would argue decidability of Gentzen derivability based on analyticity of the rules. In Coq, we represent the Gentzen system as an inductively defined predicate with one constructor per rule. This representation works very well for our proofs. However, it leaves the notion of rule implicit. Consequently, it is difficult to state the analyticity property in such a way that decidability of derivability follows from analyticity. One can obtain a direct decidability proof by expressing one-step derivability within some clause universe as a monotone and bounded function from sets of clauses to sets of clauses and showing that a clause C is derivable iff it is contained in the fixpoint of one-step derivability in $2^{\mathsf{sfc}\, C}$.

Remark 3.6.8 Our interpretation of clauses as conjunctions of signed formulas naturally lends itself to tableau-style interpretation of the Gentzen system. Alternatively, one can interpret clauses of the form $\{s_1^+, \dots, s_n^+, t_1^-, \dots, t_m^-\}$ as sequents $s_1, \dots, s_n \Rightarrow t_1, \dots, t_m$. This would correspond to associating to a clause C the formula

$$(\bigwedge_{s^+ \in C} s) \to \bigvee_{t^- \in C} t$$

It is easy to see that a clause C is unsatisfiable with respect to the conjunctive interpretation iff it corresponds to a valid sequent. Hence, the calculus in Figure 3.2 is also sound and complete for the sequent interpretation. Signed formulas thus allow for a uniform notation for tableau and sequent calculi. In this thesis we will only consider the tableau-style interpretation. We speak of Gentzen systems to emphasize the fact that, in contrast to many tableau methods, derivability is a simple inductive definition based on a collection of analytic rules.

3.7 Universal Model

We now construct a universal model for K, i.e., a classical model that satisfies all satisfiable formulas. Such a model must necessarily be infinite.

In a classical setting, one can obtain a universal model by constructing the canonical model for the Hilbert system using the Lindenbaum construction [Fit07]. For the Lindenbaum construction one takes the maximally consistent sets of formulas as states. Here, a set is maximally consistent if no subset is Hilbert refutable and every proper extension has a Hilbert refutable subset. The transition relation is defined such that there is a transition from a state A to a state B iff $\{\, s \mid \Box s \in A \,\} \subseteq B$. Every state of this model satisfies all formulas it contains.

In our constructive setting, the Lindenbaum construction is problematic for several reasons. The main problem is that the transition relation is infinitely branching. Infinite branching makes it impossible to prove that the resulting model is a classical model. We give an alternative construction using the satisfiable clauses as states together with a suitable finitely branching transition relation.

We call a model $\mathcal{M}$ a **decidable model** if equality of states, the transition relation and the labeling relation are decidable. That is, decidable models have the same decidability assumptions as finite models, but allow for an infinite number of states. Decidable models need not be classical models. However, finitely branching decidable models are classical models.

Lemma 3.7.1 Let $\mathcal{M}$ be a decidable model and let $f : \mathcal{M} \to \mathsf{set}\,\mathcal{M}$ a function such that $v \in fw$ whenever $w \Rightarrow_{\mathcal{M}} v$. Then $\mathcal{M}$ is a classical model.

Proof Similar to the proof of Lemma 3.2.5 using the fact that for $w \vDash \Box s$ we only need to check $v \vDash s$ for the finitely many successors v of w contained in fw. ∎

Our universal model for K will come in the form of a finitely branching decidable model. For the construction we exploit that Gentzen derivability is decidable.

Lemma 3.7.2 Let C such that $\nVdash C$. Then $D \rhd C$ and $\nVdash D$ for some clause D.

Proof Immediate with Lemma 3.6.2 and decidability of derivability. ■

We now define a universal model $\mathcal{M}_K$ as follows:

$$\begin{aligned} |\mathcal{M}_K| &:= \Sigma C. \nVdash C \\ C \Rightarrow_{\mathcal{M}_K} D &:= \exists \Box s^- \in C.\ D = \varepsilon(\lambda E \in |\mathcal{M}_K|. E \rhd \mathcal{R}C, s^-) \\ \Lambda_{\mathcal{M}_K}\, p\, C &:= p^+ \in C \end{aligned}$$

The type $|\mathcal{M}_K|$ is the type of dependent pairs of clauses C and proofs of $\nVdash C$. Since Gentzen derivability is decidable, $\nVdash C$ can be represented such that it has at most one proof. Hence, equality on the states of $\mathcal{M}_K$ is decidable since two states of $\mathcal{M}_K$ are equal iff the underlying clauses are the same.

For the transition relation and the labeling we suppress the proofs paired with the clauses. By Lemma 3.7.2 and the rule X, the predicate $\lambda E \in |\mathcal{M}_K|. E \rhd \mathcal{R}C, s^-$ in the definition of $\Rightarrow_{\mathcal{M}_K}$ is nonempty whenever $\Box s^- \in C$. Therefore, the ε-expression is well-defined. While the construction of $\mathcal{M}_K$ is inspired by the Lindenbaum construction, $\mathcal{M}_K$ is not a "canonical" model since the transition relation depends on the choice operator for formulas.

Lemma 3.7.3 $\mathcal{M}_K$ is a classical model.

Proof It is easy to see that $\mathcal{M}_K$ is a decidable model. Moreover, every state C has at most as many successors as there are formulas of the form $\Box s^-$ in C. Hence, $\mathcal{M}_K$ is a classical model by Lemma 3.7.1. ■

It remains to show that $\mathcal{M}_K$ satisfies all satisfiable formulas.

Lemma 3.7.4 Let $C \in \mathcal{M}_K$ such that $C \rhd s^\sigma$. Then $\mathcal{M}_K, C \vDash s^\sigma$.

Proof The clause C is locally consistent since $\nVdash C$. The claim then follows by induction on s similar to the proof of Lemma 3.3.3. ■

Note that we do not require that the states of $\mathcal{M}_K$ are literal clauses. Since non-literals are ignored by the support relations, this makes no difference.

Theorem 3.7.5 Let s be satisfiable. Then $\mathcal{M}_K, D \vDash s$ for some $D \in \mathcal{M}_K$.

Proof Since s is satisfiable, we have $\nVdash \{s^+\}$. The claim follows with Lemma 3.7.4 and Lemma 3.7.2. ■

Theorem 3.7.5 shows that there are relevant infinite classical models that can be obtained and reasoned about constructively.

We remark that the universal model based on underivable clauses essentially yields an alternative completeness proof for the Gentzen system. The construction relies on decidability of derivability in the Gentzen system, but this can be shown independently of completeness (cf. Remark 3.6.7). For the Hilbert system, the easiest way to establish decidability is through completeness. That is, while the construction can be used to provide an alternative completeness proof for the Gentzen system, it is not useful for giving a direct completeness proof for the Hilbert system. Also, the construction does not establish compactness since we only consider finite sets of formulas as states.

3.8 Remarks

The formalization of the results presented in this chapter [ACF] follows the mathematical presentation fairly closely. This is possible since we have carefully designed the proofs with the formalization in mind. Of course, the formalization includes a fair amount of additional detail that is omitted in the mathematical presentation. In particular, we have consistently ignored all side conditions that certain clauses must only contain formulas from a given subformula universe.

We remark that the notion of demo employed in the formalization is slightly more permissive than Definition 3.3.2. Instead of requiring demos to be comprised of base clauses, we only require local consistency. This does not affect the proof of Lemma 3.3.3 since non-literals are ignored by the support relation, but it means that technically only the literal clauses in a demo are guaranteed to be satisfiable.

Some of the constructions in this Chapter can be generalized in such a way that they can be reused in the developments for modal logic with transitive closure (Chapter 4) and CTL (Chapter 5). This includes some infrastructure for signed formulas and the support relation. We develop a library which for a given type of formulas, a definition of literal, and a support relation provides for instance the extensions of the support relation to clauses and sets of clauses as well as associated lemmas. Moreover, we construct Hilbert derivations in an abstract setting in such a way that the results apply to any Hilbert system extending the one in Figure 3.1. This is described in detail in Chapter 7.

4 Modal Logic with Transitive Closure

In this chapter we prove decidability of satisfiability as well as soundness and completeness of a Hilbert system for K^*. The logic K^* [KS10] (also called UB^- in [EH85]) extends K with a modality $\Box^*$ for transitive reachability. Intuitively, a formula $\Box^* s$ holds at a state w if s holds at all states that are reachable from w. By duality, this yields an eventuality $\Diamond^* s$ which holds at a state if s holds at some reachable state.

K^* is a sublogic of PDL [FL79] and the simplest representative of a class of non-compact modal logics with an EXPTIME-complete satisfiability problem [BdRV01]. Other representatives of this class are PDL, CTL [CE82, EH85], and the propositional μ-calculus [Koz83]. We are interested in K^* because it allows us to focus on the treatment of eventualities without the added complexity caused by the programs of PDL or CTL's "always until" modality.

The $\Box^*$-modality can be characterized coinductively following the embedding of K^* into the μ-calculus (i.e., $\Box^* s \equiv \nu X. s \wedge \Box X$). We give a Hilbert system whose rules and axioms directly correspond to the coinductive interpretation. A variant of this system can be obtained by adapting Segerberg's axiomatization [Seg77, Seg82] of PDL. We give a direct completeness proof for the first system. Completeness of the second system follows since the equivalence of the two Hilbert systems can easily be established syntactically.

Similar to Chapter 3, we base the completeness proof for the Hilbert system on a pruning system constructing demos and refutations. The crucial part in the definition of demos is the treatment of transitive closure. We handle the fixpoint conditions for $\Box^* s$ and $\Diamond^* s$ by adapting the notion of support. The well-foundedness condition for the eventuality $\Diamond^* s$ are treated using an inductively defined fulfillment predicate. Pruning refutations then employ inductive fulfillment in negated form providing exactly the invariants required for the translation to Hilbert refutations.

4.1 Formulas and Models

We define the **formulas** of K^* as follows:

$$s, t ::= p \mid \bot \mid s \to t \mid \Box s \mid \Box^* s$$

We continue to use the abbreviations from Section 3.1 and use the following additional abbreviation: $\Diamond^* s := \neg \Box^* \neg s$.

As for basic modal logic K (Section 3.1), the class of models for K* is the class of all labeled transition systems. Let $\mathcal{M} = (|\mathcal{M}|, \Rightarrow_{\mathcal{M}}, \Lambda_{\mathcal{M}})$ be a model. The satisfaction relation for K extends to K* as follows:

$$\begin{aligned} w \vDash p &:= \Lambda_{\mathcal{M}}\, p\, w \\ w \vDash \bot &:= \bot \\ w \vDash s \to t &:= w \vDash s \to w \vDash t \\ w \vDash \Box s &:= \forall v \in \mathcal{M}. (w \Rightarrow_{\mathcal{M}} v) \to v \vDash s \\ w \vDash \Box^* s &:= \forall v \in \mathcal{M}. (w \Rightarrow_{\mathcal{M}}^* v) \to v \vDash s \end{aligned}$$

It is often useful to think of $\Box^* s$ in terms of its encoding in the propositional μ-calculus [Koz83], i.e., $\Box^* s \equiv \nu X.\ s \wedge \Box X$. That is, $\Box^* s$ can be seen as a greatest fixpoint. In type theory, this corresponds to a coinductive definition. Let $\mathcal{M}$ be a general model. We define a coinductive predicate AG ("always globally") of type $(\mathcal{M} \to \mathsf{Prop}) \to \mathcal{M} \to \mathsf{Prop}$ with the following rule:

$$\frac{P\, w \qquad \forall v. w \Rightarrow_{\mathcal{M}} v \to \mathsf{AG}\, P\, w}{\mathsf{AG}\, P\, w}$$

The predicate AG yields an alternative characterization of $w \vDash \Box^* s$. We define $w \vDash_c \Box^* s := \mathsf{AG}(\lambda v. v \vDash s) w$.

Lemma 4.1.1 Let $\mathcal{M}$ be a model and let $w \in \mathcal{M}$. Then $w \vDash \Box^* s$ iff $w \vDash_c \Box^* s$.

For our proofs we mostly work with the coinductive characterization because it leads to slightly more direct proofs.

As we have done for K (Section 3.2), we restrict our attention to classical models. A **classical model** is a model $\mathcal{M}$ where $\vDash$ is logically decidable (i.e., $w \vDash s \vee w \nvDash s$ for all formulas s and states $w \in \mathcal{M}$). In the following, the semantic notions (satisfiability, validity, etc.) refer to the class of classical models unless stated otherwise.

4.2 Hilbert System

The valid formulas of K* can be axiomatized with a Hilbert system. The rules and axioms are given in Figure 4.1. The system extends the axiomatization of K (Figure 3.1) with two axioms (G1 and G2) and an additional rule (GI). The axioms and rules for $\Box^*$ are inspired by the coinductive interpretation of $\Box^*$. The axioms G1 and G2 correspond to inversion of the predicate AG, and GI corresponds to coinduction. Similar axioms and rules appear in the Hilbert system for CTL [EH85].

To distinguish different Hilbert systems, we write $\mathbf{A} \vdash s$ if the Hilbert system **A** can prove s. We refer to the Hilbert system in Figure 3.1 as **K** and to the Hilbert system in Figure 4.1 as **K***. Note that the Hilbert system **K*** extends the Hilbert

$$\begin{array}{rl}
\mathsf{S} & \vdash (u \to s \to t) \to (u \to s) \to u \to t \\
\mathsf{K} & \vdash s \to t \to s \\
\mathsf{DN} & \vdash ((s \to \bot) \to \bot) \to s \\
\mathsf{N} & \vdash \Box(s \to t) \to \Box s \to \Box t \\
\mathsf{G1} & \vdash \Box^* s \to s \\
\mathsf{G2} & \vdash \Box^* s \to \Box\Box^* s
\end{array}$$

$$\frac{\vdash s \qquad \vdash s \to t}{\vdash t}\,\mathsf{MP} \qquad \frac{\vdash s}{\vdash \Box s}\,\mathsf{Nec} \qquad \frac{\vdash u \to \Box u \qquad \vdash u \to s}{\vdash u \to \Box^* s}\,\mathsf{GI}$$

Figure 4.1: Hilbert system $\mathbf{K}^*$

system **K**. In particular, Lemma 3.5.4 also applies to $\mathbf{K}^*$. In this chapter, we take $\vdash s$ to mean $\mathbf{K}^* \vdash s$.

The Hilbert system $\mathbf{K}^*$ is sound for classical models.

Theorem 4.2.1 (Soundness) If $\vdash s$, then s is valid.

Proof By induction on $\vdash s$. We consider the case for GI. By induction hypothesis, we know (a) that $u \to \Box u$ is valid and (b) that $u \to s$ is valid.

Now let $\mathcal{M}$ be model. By Lemma 4.1.1, it suffices to show

$$\forall w \in \mathcal{M}. w \vDash u \to w \vDash_c \Box^* s$$

by coinduction. That is, we obtain $\forall w \in \mathcal{M}. w \vDash u \to w \vDash_c \Box^* s$ as coinduction hypothesis. Assume $w \vDash u$. After applying the introduction rule for AG it suffices to show $v \vDash s$ and $v \vDash_c \Box^* s$ for all v such that $w \Rightarrow_{\mathcal{M}} v$. The first claim follows with (b). The second claim follows with (a) and the coinduction hypothesis. ■

Remark 4.2.2 Note that, in the proof above, the coinduction hypothesis is exactly the statement we are trying to prove. However, the coinduction hypothesis may only by used after the introduction rule of AG has been applied. When doing coinductive proofs in Coq, this constraint is enforced using a guardedness check at the end of the proof.

The GI-rule matches the coinductive characterization of $\Box^*$ very closely. This leads to a very direct soundness proof. Further, the GI-rule allows for Hilbert proofs which are fairly close to the mathematical intuition.

The semantics of $\Box^* s$ corresponds to a greatest fixpoint (Lemma 4.1.1). Hence, its dual $\Diamond^* s$ corresponds to a least fixpoint and behaves as if defined inductively. In particular, we have the following lemma:

Lemma 4.2.3

1. $\vdash \Box^* s \leftrightarrow s \wedge \Box\Box^* s$.
2. $\vdash \Diamond^* s \leftrightarrow s \vee \Diamond\Diamond^* s$.
3. $\vdash \Diamond^* u \to s$ whenever $\vdash \Diamond u \to u$ and $\vdash u \to s$.

4.3 Demos

We now extend the completeness and decidability results from K to K*. As before, we base our proofs on a pruning system computing demos or refutations.

We begin by adapting the notion of demo. As before, a demo will be a finite set of clauses that can be interpreted as a model where the states are the clauses from the demo and every state satisfies all formulas it supports. In fact, the model construction for K* is exactly the same as for K (cf. Section 3.3).

We continue to work with signed formulas s^σ (cf. Section 3.3.1). Recall that, semantically, negative signs correspond to top-level negations. The notion of literal remains unchanged, i.e., formulas of the form $\bot^\sigma$, p^σ, and $\Box s^\sigma$ are **literals**. In particular, $\Box^* s^\sigma$ is not a literal. A clause is **locally consistent** if it contains neither $\bot^+$ nor p^+ and p^- for any p. As before, we refer to locally consistent literal clauses as **base clauses**. The **request** of a clause is defined as before (i.e., $\mathcal{R}\, C := \{\, s^+ \mid \Box s^+ \in C \,\}$).

We extend the definitions of demo and support from Section 3.3 to the formulas of K*. Following the equivalences in Lemma 4.2.3, the **support relation** between clauses C and signed formulas s^σ extends to K* as follows:

$$\begin{aligned}
C \rhd s^\sigma &:= s^\sigma \in C && \text{if } s^\sigma \text{ is a literal}\\
C \rhd (s \to t)^+ &:= C \rhd s^- \ \lor\ C \rhd t^+\\
C \rhd (s \to t)^- &:= C \rhd s^+ \ \land\ C \rhd t^-\\
C \rhd \Box^* s^+ &:= C \rhd s^+ \ \land\ \Box\Box^* s^+ \in C\\
C \rhd \Box^* s^- &:= C \rhd s^- \ \lor\ \Box\Box^* s^- \in C
\end{aligned}$$

We continue to use the abbreviations from before:

$$C \rhd D := \forall s \in D.\, C \rhd s \qquad\qquad S \rhd C := \exists D \in S.\, D \rhd C$$

Adapting the notion of support is all that is required for formulas of the form $\Box^* s^+$. For a formula of the form $\Box^* s^-$ to be satisfied at a state w of some finite model $\mathcal{M}$, there must exist a path through $\mathcal{M}$ from w to some state v of $\mathcal{M}$ that satisfies s^-, i.e., s must be dissatisfied eventually. Following [Pnu77], we call formulas of the form $\Box^* s^-$ **eventualities** and formulas of the form $\Box\Box^* s^-$ **eventuality literals**.

Let C be a clause. If $C \rhd s^-$, we say that the eventuality $\Box^* s^-$ is **fulfilled locally** by C. Conversely, if $C \rhd \Box^* s^-$ but $C \not\rhd s^-$ we say $\Box^* s^-$ is **deferred** in C. To satisfy an eventuality, we need to ensure that the eventuality is fulfilled at some point and not deferred forever. For this purpose, we inductively define the **fulfillment relation** $S, C \rhd \Box\Box^* s^-$ between sets of clauses S, clauses $C \in S$ and eventuality literals $\Box\Box^* s^-$.

$$\frac{D \in S \qquad D \rhd \mathcal{R}\, C, s^-}{S, C \rhd \Box\Box^* s^-} \qquad\qquad \frac{D \in S \qquad D \rhd \mathcal{R}\, C \qquad S, D \rhd \Box\Box^* s^-}{S, C \rhd \Box\Box^* s^-}$$

Definition 4.3.1 (Demo for K*) A set of base clauses $\mathcal{S}$ is a **demo** if the following conditions are satisfied

(D□) If $\Box s^- \in C \in \mathcal{S}$, then $\mathcal{S} \rhd \mathcal{R}\, C, s^-$.

(D□*) If $\Box\Box^* s^- \in C \in \mathcal{S}$, then $\mathcal{S}, C \rhd \Box\Box^* s^-$.

Just like demos for K, every demo $\mathcal{S}$ can be interpreted as a model $\mathcal{M}_\mathcal{S}$ where $|\mathcal{M}_\mathcal{S}| := \mathcal{S}$, $C \Rightarrow_{\mathcal{M}_\mathcal{S}} D := D \rhd \mathcal{R}\, C$, and $\Lambda_{\mathcal{M}_\mathcal{S}}\, p\, C := p^+ \in C$ (cf. Section 3.3).

Lemma 4.3.2 Let $\mathcal{S}$ be a demo and $C \in \mathcal{S}$. Then $\mathcal{M}_\mathcal{S}, C \vDash t^\sigma$ if $C \rhd t^\sigma$.

Proof By induction on t. We consider the cases for $\Box^* s$. The remaining cases are handled exactly as in the proof of Lemma 3.3.3.

Case $t = \Box^* s^+$. By Lemma 4.1.1, it suffices to show $\forall C \in \mathcal{S}. C \rhd \Box^* s^+ \to \mathcal{M}_\mathcal{S}, C \vDash_c \Box^* s$ by coinduction. Let $C \in \mathcal{S}$ and $C \rhd \Box^* s^+$. Then $C \rhd s^+$ and assume $\mathcal{R}\, C \rhd \Box^* s^+$. After applying the introduction rule for AG we need to prove $C \vDash s$ and $D \vDash_c \Box^* s$ for all $D \in \mathcal{S}$ such that $D \rhd \mathcal{R}\, C$. The first claim follows by induction hypothesis, the second with the coinduction hypothesis.

Case $t = \Box^* s^-$. Let $C \in \mathcal{S}$ and $C \rhd \Box^* s^-$. If $\Box^* s^-$ is fulfilled locally by C, the claim follows by induction hypothesis. Otherwise, we have $\Box\Box^* s^- \in C$ and therefore also $\mathcal{S}, C \rhd \Box\Box^* s^-$. The claim then follows by induction on $\mathcal{S}, C \rhd \Box\Box^* s^-$. ■

Remark 4.3.3 Instead of using a direct inductive definition, fulfillment of eventualities can also be expressed in terms of the transitive closure of the following relation: $C \Rightarrow_\mathcal{S} D := D \in \mathcal{S} \wedge D \rhd \mathcal{R}\, C$. We employ the direct inductive definition because it leads to slightly more direct proofs and extends smoothly to CTL.

4.4 Pruning and Refutation Calculus

We now extend the pruning system and the associated refutation calculus to K*. As before, pruning removes clauses violating the demo conditions. The demo condition for eventualities gives rise to an additional refutation rule.

4.4.1 Subformula Universes

We call a finite set of signed formulas U a **subformula universe**, if it satisfies the following closure conditions:

U1. If $s \to t^\sigma \in U$ then $\{s^{\overline{\sigma}}, t^\sigma\} \subseteq U$.

U2. If $\Box s^\sigma \in U$ then $s^\sigma \in U$.

U3. If $\Box^* s^\sigma \in U$ then $\{s^\sigma, \Box\Box^* s^\sigma\} \subseteq U$.

Given a clause C, we write $\mathsf{sfc}\, C$ for the smallest subformula universe extending C. Note that the subformula closure of $\Box^* s^\sigma$ contains $\Box\Box^* s^\sigma$ which is larger than $\Box^* s^\sigma$. Nevertheless, the clause $\mathsf{sfc}\, C$ can be computed by structural recursion:

$$
\begin{aligned}
\mathsf{sub}\,(s \to t^\sigma) &:= \mathsf{sub}\, s^{\overline{\sigma}} \cup \mathsf{sub}\, t^\sigma \cup \{s \to t^\sigma\} \\
\mathsf{sub}\,(\Box s^\sigma) &:= \mathsf{sub}\, s^\sigma \cup \{\Box s^\sigma\} \\
\mathsf{sub}\,(\Box^* s^\sigma) &:= \mathsf{sub}\, s^\sigma \cup \{\Box^* s^\sigma, \Box\Box^* s^\sigma\} \\
\mathsf{sub}\, p &:= \{p\} \\
\mathsf{sub}\, \bot &:= \{\bot\} \\
\mathsf{sfc}\, C &:= \bigcup_{s^\sigma \in C} \mathsf{sub}\, s^\sigma
\end{aligned}
$$

Note that while the formula $\Box\Box^* s^\sigma$ on the right hand side of $\mathsf{sub}\,(\Box^* s^\sigma)$ is larger than $\Box^* s^\sigma$, it does not require a recursive call.

Lemma 4.4.1 Let s^σ be a signed formula and let C be a clause. Then

1. $|\mathsf{sub}\, s^\sigma| \leq 2 \cdot |s|$
2. $\mathsf{sub}\, s^\sigma$ is a subformula universe and $s^\sigma \in \mathsf{sub}\, s^\sigma$.
3. $\mathsf{sfc}\, C$ is a subformula universe and $C \subseteq \mathsf{sfc}\, C$.

4.4.2 Pruning

We fix some subformula universe U. As before, we construct the demo of satisfiable base clauses over U by using the demo conditions as pruning rules.

To use (D$\Box^*$) as a pruning rule, we need to show that the fulfillment relation is decidable. For this we make use of the fact that inductive predicates over finite sets that only employ decidable side conditions can be decided using fixpoint iteration.

Lemma 4.4.2 Let $C \in S \subseteq \overline{U}$ and let $\Box\Box^* s^- \in U$. Then $S, C \rhd \Box\Box^* s^-$ is decidable.

Proof We construct a boolean reflection of $S, C \rhd \Box\Box^* s^-$ using fixpoint iteration. For this we express the constructors of the inductive definition as a monotone and bounded function from sets of clauses to sets of clauses:

$$F := \lambda \mathcal{T}.\{\, C \in S \mid \exists D \in S. D \rhd \mathcal{R}\, C \wedge (D \rhd s^- \vee D \in \mathcal{T})\,\}$$

It then suffices to show $S, C \rhd \Box\Box^* s^- \leftrightarrow C \in \mathrm{lfp}\, F$. The direction from left to right follows by induction on $S, C \rhd \Box\Box^* s^-$ using Lemma 2.5.1. The converse direction follows by induction on the fixpoint (Lemma 2.5.2(1)). ■

We then compute the canonical demo for U as follows:

$$
\begin{aligned}
p\, C\, S &:= (\forall \Box s^- \in C.\ S \rhd \mathcal{R}\, C, s^-) \wedge (\forall \Box\Box^* s^- \in C.\ S, C \rhd \Box\Box^* s^-) \\
\mathcal{D}(U) &:= \mathsf{prune}\, p\, \overline{U}
\end{aligned}
$$

Lemma 4.4.3

1. $\mathcal{D}(U) \subseteq \overline{U}$
2. $\mathcal{D}(U)$ is a demo.

Proof Immediate with Lemma 3.4.3. ■

4.4.3 Refutation Calculus

We characterize the unsatisfiable clauses over U using an inductively defined refutation calculus. For this, we extend the refutation calculus for K with an additional rule refuting those base clauses that can be pruned because they violate the demo condition (D□*). The rules of the **refutation calculus** are given in Figure 4.2. Save for the changed notion of support, the first two rules are unchanged from the refutation calculus for K (Section 3.4.3).

It is straightforward to show that the refutation calculus refutes all clauses over U that are not supported by $\mathcal{D}(U)$.

Lemma 4.4.4 (Refutation Completeness)

1. $\mathsf{coref}_U\, \mathcal{D}(U)$.
2. $\mathsf{ref}_U\, C$ whenever $C \subseteq U$ and $\mathcal{D} \not\triangleright C$.

Proof Similar to the proof of Lemma 3.4.7 ■

While the refutation calculus is essentially complete by construction, showing soundness is more involved.

Proposition 4.4.5 (Refutation Soundness) If $\mathsf{ref}_U\, C$, then C is unsatisfiable.

Instead of giving a detailed proof of Proposition 4.4.5, we will show the stronger claim that $\mathsf{ref}_U\, C$ implies $\vdash \neg C$ (cf. Lemma 4.5.5).

Combining model existence and refutation completeness, we obtain the following theorem.

Theorem 4.4.6 Let U be a subformula universe and let $C \subseteq U$. Then either $\mathsf{ref}_U\, C$ or C is satisfied by a model with at most $2^{|U|}$ states.

4.5 Completeness of the Hilbert System

We now translate derivations from the refutation calculus to Hilbert refutations. We fix some subformula universe U. We call a set of clauses $\mathcal{S}$ **Hilbert corefutable** if we have $\vdash \neg C$ for all $C \in \overline{U} \setminus \mathcal{S}$.

Given that the Hilbert system for K* extends the Hilbert system for K, the translation of the jump rule carries over without changes. For the translation of the support rule, it suffices to adapt the proof of Lemma 3.5.2 to take care of the additional non-literals.

$$\frac{C \subseteq U \qquad S \ntriangleright C \qquad \mathsf{coref}_U\, S}{\mathsf{ref}_U\, C}\;\mathsf{supp} \qquad\qquad \frac{\Box s^- \in C \qquad \mathsf{ref}_U\,(\mathcal{R}\, C, s^-)}{\mathsf{ref}_U\, C}\;\mathsf{jump}$$

$$\frac{\Box\Box^* s^- \in C \in S \subseteq \overline{U} \qquad S, C \ntriangleright \Box\Box^* s^- \qquad \mathsf{coref}_U\, S}{\mathsf{ref}_U\, C}\;\mathsf{loop}$$

$$\mathsf{coref}_U\, S := \forall C \in \overline{U}.\; C \notin S \to \mathsf{ref}\, C$$

Figure 4.2: Refutation calculus for K*

Lemma 4.5.1 Let S be Hilbert corefutable and let $C \subseteq U$. Then $\vdash C \to \bigvee \mathcal{B}_S\, C$.

Proof Essentially the same as the proof of Lemma 3.5.2. The cases for formulas of the form $\Box^* s^\sigma$ follow with Lemma 4.2.3. ■

Lemma 4.5.2 (Admissibility of Support rule) Let S be Hilbert corefutable and let $C \subseteq U$. Then $\vdash \neg C$ whenever $S \ntriangleright C$.

The translation of the loop rule employs the induction rule GI with an elaborate invariant. Similar invariants appear in the completeness proofs of the Hilbert systems for UB [BAPM83] and CTL [EH85, Eme90]. Instead of using the GI-rule directly, we use a derived coinduction principle for $\Box\Box^* s$.

Lemma 4.5.3 If $\vdash u \to \Box(u \wedge s)$, then $\vdash u \to \Box\Box^* s$.

Proof Assume (a) $\vdash u \to \Box(u \wedge s)$. We reason as follows:

$$\begin{array}{lll}
 & \vdash u \to \Box\Box^* s & \\
\Leftarrow & \vdash \Box(u \wedge s) \to \Box\Box^* s & \text{(a)} \\
\Leftarrow & \vdash u \wedge s \to \Box^* s & \text{Lemma 3.5.4(1)} \\
\Leftarrow & \vdash u \wedge s \to \Box(u \wedge s) & \mathsf{GI}
\end{array}$$

The last claim follows with (a). ■

Lemma 4.5.4 (Admissibility of Loop Rule) Let $S \subseteq \overline{U}$ be Hilbert corefutable and let $\Box\Box^* s^- \in C \in S$. Then $\vdash \neg C$ whenever $S, C \ntriangleright \Box\Box^* s^-$.

Proof Assume $S, C \ntriangleright \Box\Box^* s^-$. To refute C it suffices to show $\vdash C \to \Box\Box^* s$. By Lemma 4.5.3, it suffices to find an invariant u for which we can prove:

$$\vdash C \to u \tag{4.1}$$

$$\vdash u \to \Box(u \wedge s) \tag{4.2}$$

We define

$$I := \{ D \in S \mid S, D \ntriangleright \Box\Box^* s^- \} \qquad\qquad u := \bigvee I$$

Claim (4.1) follows since $C \in I$. For Claim (4.2) note that for all clauses $D \in I$ we have

$$S \not\triangleright \mathcal{R}\,D, s^- \tag{4.3}$$

$$\mathcal{B}_S\,(\mathcal{R}\,D) \subseteq I \tag{4.4}$$

since violating either property would allow us to prove $S, D \triangleright \Box\Box^* s^-$. We then reason as follows

$$\begin{array}{lll}
 & \vdash u \to \Box(u \wedge s) & \\
\Leftarrow & \vdash D \to \Box(u \wedge s) & \text{for } D \in I \\
\Leftarrow & \vdash \Box(\mathcal{R}\,D) \to \Box(u \wedge s) & \text{Lemma 3.5.4(3)} \\
\Leftarrow & \vdash \mathcal{R}\,D \to u \wedge s & \text{Lemma 3.5.4(1)} \\
\Leftarrow & \vdash \bigvee \mathcal{B}_S(\mathcal{R}\,D) \to \bigvee I \text{ and } \vdash \neg(\mathcal{R}\,D, s^-) & \text{Lemma 4.5.1}
\end{array}$$

The left claim follows with (4.4) and the right claim follows with (4.3). ■

Lemma 4.5.5 $\vdash \neg C$ whenever $\mathsf{ref}_U\, C$.

Proof By induction on $\mathsf{ref}_U\, C$ using Lemmas 3.5.5, 4.5.2, and 4.5.4 ■

Note that Proposition 4.4.5 follows immediately with the lemma above and soundness of the Hilbert system. Moreover, we have that $\mathcal{D}(U)$ is the canonical demo for U.

Theorem 4.5.6 Let $C \in \overline{U}$. Then $C \in \mathcal{D}(U)$ iff C is satisfiable.

Proof If $C \in \mathcal{D}(U)$, then $C \triangleright C$. Hence, C is satisfiable by Lemma 4.3.2. If $C \notin \mathcal{D}(U)$, then $\mathsf{ref}_U\, C$ (Lemma 4.4.4(1)). Hence, C is unsatisfiable (Proposition 4.4.5).■

We now obtain the informative completeness result for the Hilbert system.

Theorem 4.5.7 (Informative Completeness) Let s be a formula. Then either $\vdash \neg s$ or s is satisfied by a model with at most $2^{2\cdot|s|}$ states.

Proof Let $U := \mathsf{sfc}\{s^+\}$. By Theorem 4.4.6, we either have $\mathsf{ref}_U\, \{s^+\}$ and hence $\vdash \neg s$ by Lemma 4.5.5 or s is satisfied by a model of the right size (Lemma 4.4.1).■

Together with soundness we obtain the following corollaries.

Corollary 4.5.8 (Completeness) If s is valid, then $\vdash s$.

Corollary 4.5.9 (Decidability) Satisfiability, validity, and Hilbert provability of formulas are decidable.

Corollary 4.5.10 (Small Models) If s is satisfiable, then s is satisfied by a model with at most $2^{2\cdot|s|}$ states.

Recall that every general model is a classical model if one assumes XM. In that case, Corollary 4.5.10 yields the small-model property with respect to arbitrary Kripke structures.

4.6 Hilbert System in Segerberg Style

Segerberg [Seg77] gives a Hilbert system for PDL. Adapting this axiomatization to K^* yields a Hilbert system where the GI-rule is replaced with the following axioms and rules:

$$\begin{array}{lll} \mathsf{N}^* & \vdash \Box^*(s \to t) \to \Box^* s \to \Box^* t \\ \mathsf{Seg} & \vdash \Box^*(s \to \Box s) \to s \to \Box^* s \end{array} \qquad \frac{\vdash s}{\vdash \Box^* s}\,\mathsf{Nec}^*$$

We call the resulting Hilbert system **S**. We show that the Hilbert systems $\mathbf{K}^*$ and **S** are equivalent by showing that the rules and axioms of each system are admissible in the other. Note that both systems are extensions of **K** and include the G1 and G2. In particular, all theorems of **K** are also theorems of $\mathbf{K}^*$ and **S**.

Theorem 4.6.1 Let s be a formula. Then $\mathbf{S} \vdash s$ iff $\mathbf{K}^* \vdash s$.

Proof It suffices to show that the GI-rule is admissible for **S** and that N^*, Seg, and Nec^* are admissible for $\mathbf{K}^*$. We show the cases for the GI-rule and the axiom Seg. The cases for Nec^* and N^* are similar to the case for Seg.

- GI-rule. Assume (a) $\mathbf{S} \vdash u \to \Box u$ and (b) $\mathbf{S} \vdash u \to s$. We need to show $\mathbf{S} \vdash u \to \Box^* s$. We reason as follows:

$$\begin{array}{lll} 1. & \mathbf{S} \vdash \Box^*(u \to \Box u) & \mathsf{Nec}^* \text{ on (a)} \\ 2. & \mathbf{S} \vdash u \to \Box^* u & \mathsf{Seg},\ 1 \\ 3. & \mathbf{S} \vdash \Box^*(u \to s) & \mathsf{Nec}^* \text{ on (b)} \\ 4. & \mathbf{S} \vdash u \to \Box^* s & \mathsf{N}^*,\ 2,\ 3 \end{array}$$

- Seg-axiom. We show $\mathbf{K}^* \vdash \Box^*(s \to \Box s) \to s \to \Box^* s$. Let $u := s \wedge \Box^*(s \to \Box s)$. By propositional reasoning, it suffices to show $\mathbf{K}^* \vdash u \to \Box^* s$ for all formulas s. We clearly have $\mathbf{K}^* \vdash u \to s$. By the GI-rule, it suffices to show $\mathbf{K}^* \vdash u \to \Box u$. This follows with Lemma 4.2.3(1) and Lemma 3.5.4(2). ■

Together with Theorem 4.5.7 we obtain completeness of the Hilbert system **S**.

4.7 Remarks

The notion of demo (Definition 4.3.1) is based on inductive fulfillment for eventuality literals rather than the eventualities themselves. This interacts smoothly with the support relation. Moreover, defining fulfillment for eventuality literals avoids the need for maximality conditions. Recall that $S, C \ntriangleright \Box^+ s^-$ implies $S \ntriangleright \mathcal{R}\,C, s^-$. For corefutable S, this yields $\vdash \neg(\mathcal{R}\,C, s^-)$. For the natural definition of $S, C \triangleright \Box^* s^-$, the negation $S, C \ntriangleright \Box^* s^-$ would only yield $C \ntriangleright s^-$. This only implies $\vdash \neg(C, s^-)$ when C is maximal in S.

The refutation calculus arising with the demo conditions (Figure 4.2) is a variation of the "analytic tableau system" employed in preliminary work [DS12]. The system in [DS12] employs a monolithic *compound rule* similar to the loop-rule that, when read in the backward direction, requires guessing a collection of clauses from the subformula universe satisfying certain closure conditions. The design goal for both systems was to obtain an inductive definition of unsatisfiability that can be translated to Hilbert refutations.

Kashima [Kas10] gives an alternative completeness proof for modal logic with transitive closure based on ideas from [BL08]. The proof is arguably more complicated than the proof presented here.

5 Computation Tree Logic

We now extend the results from the previous chapter to Computation Tree Logic (CTL) [EC82]. We prove decidability of satisfiability, the small-model property, and soundness and completeness of a Hilbert system.

CTL extends basic modal logic K with modalities $\mathsf{A}(s\,\mathsf{R}\,t)$ and $\mathsf{A}(s\,\mathsf{U}\,t)$. Intuitively, $\mathsf{A}(s\,\mathsf{U}\,t)$ holds at some state w if for every infinite path starting at w the formula s holds at every state until the path eventually reaches a state where t holds. The modality $\mathsf{A}(s\,\mathsf{R}\,t)$ holds at some state w if for every infinite path starting at w either t holds at every state along the path or s holds at some state on the path and t holds at every state up to (and including) that state. The logic has the small model property and the satisfiability problem for formulas is EXPTIME-complete [EH85, Eme90].

Similar to before, we base our completeness results on a pruning system constructing for a given input formula either a finite pseudo-model we call demo or a pruning refutation. Completeness of the Hilbert system is obtained by translating pruning refutations to Hilbert refutations. Together with soundness of the Hilbert system, the small model property and decidability of satisfiability, validity, and Hilbert provability follow as corollaries.

The central notion underlying the completeness and decidability results is the notion of demo. Our demos play the role of the pseudo-Hintikka structures employed by Emerson and Halpern [EH85, Eme90]. As before, demos will be finite sets of clauses with closure conditions ensuring that all clauses in a demo are satisfiable. The demo conditions determine the rules of a pruning system and the corresponding notion of pruning refutations. Similar to the demos for K* (Section 4.3), we employ inductive fulfillment relations to handle the eventualities of CTL. Unlike for K and K*, demos for CTL will not be models themselves, but will need to be unfolded into proper models. This is necessary since demos can be seen as possible results of filtration and filtration fails to preserve satisfaction of $\mathsf{A}(s\,\mathsf{U}\,t)$ [EH85]. For the construction of models from demos, we employ a variant of the fragment-based model construction in [Eme90].

We design our demos such that pruning refutations can not only be translated to Hilbert refutations but also to derivations of an analytic sequent system for CTL [BL08] (cf. Chapter 6). For this purpose, we introduce relaxed demos based on a relaxed version of fulfillment. The model construction in [Eme90] provides a natural fit for relaxed demos.

In the literature, the semantics of CTL is usually defined in terms of infinite paths [Eme90, BK08]. Since infinite objects are difficult to work with in a constructive setting, we interpret path formulas inductively (for $\mathsf{A}(s\,\mathsf{U}\,t)$) or coinductively (for $\mathsf{A}(s\,\mathsf{R}\,t)$) according to the following equivalences [EC80, Eme08]:

$$\mathsf{A}(s\,\mathsf{R}\,t) \equiv t \wedge (s \vee \Box\,\mathsf{A}(s\,\mathsf{R}\,t))$$
$$\mathsf{A}(s\,\mathsf{U}\,t) \equiv t \vee (s \wedge \Box\,\mathsf{A}(s\,\mathsf{R}\,t))$$

We show that some formulations of the path semantics agree constructively with the inductive semantics on finite models. We also show that for other, classically equivalent, formulations this is not the case. For general models, establishing the equivalence between the inductive semantics and the path semantics requires excluded middle and dependent choice (cf. Section 2.2).

The axioms and rules of the Hilbert system employed in the completeness proof correspond very closely to the (co)inductive interpretation of CTL. We establish the equivalence between our Hilbert system and Hilbert systems from the literature [Eme90, LS01] by giving translations to and from our system.

5.1 Inductive Semantics for CTL

We start by defining syntax and semantics of CTL as we use it for our development. We employ an inductive semantics for CTL that is inspired by the fixpoint characterization of path formulas used for model checking [EL86, Eme08].

5.1.1 Syntax and Intuitive Semantics

We fix a countable alphabet $\mathcal{A}$ of atomic propositions p and define the **formulas** of CTL as follows:

$$s, t \;:=\; p \mid \bot \mid s \to t \mid \Box s \mid \mathsf{A}(s\,\mathsf{U}\,t) \mid \mathsf{A}(s\,\mathsf{R}\,t)$$

As before, we choose to work with a minimal set of operations for technical convenience. We continue to use the abbreviations defined for K (Section 3.1) and introduce the following additional abbreviations:

$$\mathsf{E}(s\,\mathsf{U}\,t) := \neg\,\mathsf{A}(\neg s\,\mathsf{R}\,\neg t) \qquad\qquad \mathsf{E}(s\,\mathsf{R}\,t) := \neg\,\mathsf{A}(\neg s\,\mathsf{U}\,\neg t)$$

The modalities $\mathsf{A}(s\,\mathsf{U}\,t)$ and $\mathsf{A}(s\,\mathsf{R}\,t)$ are to be read as “always s until t” (*always until*) and “always s releases t” (*always release*). Intuitively, $\mathsf{A}(s\,\mathsf{U}\,t)$ holds at some state w if for every infinite path emanating from w the formula s holds at every state until the path eventually reaches a state where t holds. The modality $\mathsf{A}(s\,\mathsf{R}\,t)$ holds at some state w if for every infinite path emanating from w either t holds at every state along the path or s holds at at some state on the path and t holds at every state up to (and including) that state. The defined modalities $\mathsf{E}(s\,\mathsf{U}\,t)$ and $\mathsf{E}(s\,\mathsf{R}\,t)$ are read as *exists until* and *exists release* respectively.

As is standard for CTL [EH85, Eme90] we only consider models where every state has at least one successor. This has the effect hat $\mathsf{A}(s\,\mathsf{U}\,t)$ is never “vacuously” satisfied due to the absence of outgoing paths. In the following, we refer to formulas of the form $\mathsf{A}(s\,\mathsf{U}\,t)$ as **universal eventualities** and to formulas of the form $\mathsf{E}(s\,\mathsf{U}\,t)$ as **existential eventualities**. Further, we refer to formulas of the form $\mathsf{A}(s\,\mathsf{R}\,t)$ and $\mathsf{E}(s\,\mathsf{R}\,t)$ as **invariants**.

5.1.2 Inductive Semantics

We now define the inductive semantics underlying our proofs. We call a relation $R : X \to X \;\; \to \mathsf{Prop}$ over some type X **serial** if every $x : X$ has some R-successor. A **model** is then a tuple $\mathcal{M} = (|\mathcal{M}|, \Rightarrow_{\mathcal{M}}, \Lambda_{\mathcal{M}})$ where

- $|\mathcal{M}|$ is a type of **states**
- $\Rightarrow_{\mathcal{M}} : |\mathcal{M}| \to |\mathcal{M}| \to \mathsf{Prop}$ is a serial **transition relation**.
- $\Lambda_{\mathcal{M}} : \mathcal{A} \to |\mathcal{M}| \to \mathsf{Prop}$ is a **labeling function**.

For the following definitions, we fix some model $\mathcal{M}$. As before, we use $\mathcal{M}$ to denote the model as well as the underlying type of states. We interpret path formulas inductively (for $\mathsf{A}(s\,\mathsf{U}\,t)$) or coinductively (for $\mathsf{A}(s\,\mathsf{R}\,t)$) according to the embedding [EL86] of CTL in the propositional μ-calculus [Koz83].

$$\mathsf{A}(s\,\mathsf{U}\,t) \equiv \mu X.\; t \vee (s \wedge \Box X) \tag{5.1}$$

$$\mathsf{A}(s\,\mathsf{R}\,t) \equiv \nu X.\; t \wedge (s \vee \Box X) \tag{5.2}$$

We define an inductive predicate AU of type

$$(\mathcal{M} \to \mathsf{Prop}) \to (\mathcal{M} \to \mathsf{Prop}) \to \mathcal{M} \to \mathsf{Prop}$$

and a coinductive predicate AR of the same type with the following rules:

$$\begin{array}{c} Q\,w \\ \hline \mathsf{AU}\,P\,Q\,w \end{array} \qquad\qquad \begin{array}{c} P\,w \qquad \forall v.(w \Rightarrow_{\mathcal{M}} v) \to \mathsf{AU}\,P\,Q\,v \\ \hline \mathsf{AU}\,P\,Q\,w \end{array}$$

$$\begin{array}{c} P\,w \qquad Q\,w \\ \hline\hline \mathsf{AR}\,P\,Q\,w \end{array} \qquad\qquad \begin{array}{c} Q\,w \qquad \forall v.(w \Rightarrow_{\mathcal{M}} v) \to \mathsf{AR}\,P\,Q\,v \\ \hline\hline \mathsf{AR}\,P\,Q\,w \end{array}$$

Based on the predicates AU and AR, we define the **satisfaction relation** $w \vDash s$ between states w of $\mathcal{M}$ and formulas s by recursion on formulas:

$$\begin{aligned}
w &\vDash \bot := \bot \\
w &\vDash p := \Lambda_{\mathcal{M}}\, p\, w \\
w &\vDash s \to t := w \vDash s \to w \vDash t \\
w &\vDash \Box s := \forall v \in \mathcal{M}.\; (w \Rightarrow_{\mathcal{M}} v) \to v \vDash s \\
w &\vDash \mathsf{A}(s\,\mathsf{U}\,t) := \mathsf{AU}\,(\lambda v.v \vDash s)\,(\lambda v.v \vDash t)\,w \\
w &\vDash \mathsf{A}(s\,\mathsf{R}\,t) := \mathsf{AR}\,(\lambda v.v \vDash s)\,(\lambda v.v \vDash t)\,w
\end{aligned}$$

We write $\mathcal{M} \vDash s$ if $w \vDash s$ for *all* states w of $\mathcal{M}$.

It is straightforward to show that the satisfaction relation is decidable on finite models.

Lemma 5.1.1 Let $\mathcal{M}$ be a finite model. Then $w \vDash s$ is decidable for every state w of $\mathcal{M}$ and every formula s.

Proof By Induction on s. Since $\mathcal{M}$ is a finite model, $L_{\mathcal{M}}\, p\, w$ and $w \Rightarrow_{\mathcal{M}} v$ are decidable for all w, v, and p. Implication and the finite quantification in the definition of $w \vDash \Box s$ preserve decidability. Hence, it suffices to show that for decidable predicates $P, Q : \mathcal{M} \to \mathbb{B}$ the predicates $\mathsf{AU}\, P\, Q$ and $\mathsf{AR}\, P\, Q$ are decidable. We construct boolean reflections for $\mathsf{AU}\, P\, Q$ and $\mathsf{AR}\, P\, Q$ using fixpoint iteration (cf. proof of Lemma 4.4.2) ∎

As we have done in previous chapters, we restrict our attention to classical models. In the case of CTL, a **classical model** is a model $\mathcal{M}$ where $\vDash$ is logically decidable (i.e., $w \vDash s \vee w \nvDash s$ for all formulas s and all states $w \in \mathcal{M}$). For the rest of this chapter, all semantic notions (satisfiability, validity, etc.) refer to classical models unless stated otherwise.

5.2 Hilbert System

Validity of formulas can be axiomatized with a Hilbert system [EH85, Eme90, Gol91, LS01]. We will show soundness and completeness of the Hilbert system **IC** given in Figure 5.1. The name **IC** stands for "induction and coinduction". The rules of the Hilbert system are motivated by the (co)inductive characterization of eventualities and invariants. The axioms U1 and U2 correspond to the introduction rules of AU and the rule UI corresponds to the induction principle for AU. Dually, the axioms R1 and R2 correspond to inversion of the rules for AR and RI corresponds to coinduction. The axioms and rules are designed such that they make minimal use of defined logical operations.

Given the close correspondence between the inductive semantics and the Hilbert system, it is straightforward to show that the Hilbert system is sound for classical models.

Theorem 5.2.1 (Soundness) If $\vdash s$ then $\mathcal{M} \vDash s$ for all classical models $\mathcal{M}$.

Proof By induction on $\vdash s$. The case for DN follows since $\vDash$ is logically decidable. The remaining cases are obvious. ∎

5.3 Relaxed Demos

We now design the decision procedure underlying our completeness and decidability results for CTL. As before, we base the proofs on a pruning system

$$
\begin{array}{rl}
\mathsf{K} & \vdash s \to t \to s \\
\mathsf{S} & \vdash ((u \to s \to t) \to (u \to s) \to u \to t) \\
\mathsf{DN} & \vdash \neg\neg s \to s \\
\mathsf{N} & \vdash \Box(s \to t) \to \Box s \to \Box t \\
\mathsf{Ser} & \vdash \neg\Box\bot \\
\mathsf{U1} & \vdash t \to \mathsf{A}(s\,\mathsf{U}\,t) \\
\mathsf{U2} & \vdash s \to \Box\,\mathsf{A}(s\,\mathsf{U}\,t) \to \mathsf{A}(s\,\mathsf{U}\,t) \\
\mathsf{R1} & \vdash \mathsf{A}(s\,\mathsf{R}\,t) \to t \\
\mathsf{R2} & \vdash \mathsf{A}(s\,\mathsf{R}\,t) \to \neg s \to \Box\,\mathsf{A}(s\,\mathsf{R}\,t)
\end{array}
$$

$$
\frac{\vdash s \qquad \vdash s \to t}{\vdash t}\;\mathsf{MP}
\qquad\qquad
\frac{\vdash s}{\vdash \Box s}\;\mathsf{Nec}
$$

$$
\frac{\vdash t \to u \qquad \vdash s \to \Box u \to u}{\vdash \mathsf{A}(s\,\mathsf{U}\,t) \to u}\;\mathsf{UI}
\qquad\qquad
\frac{\vdash u \to t \qquad \vdash u \to \neg s \to \Box u}{\vdash u \to \mathsf{A}(s\,\mathsf{R}\,t)}\;\mathsf{RI}
$$

Figure 5.1: Hilbert system **IC**

constructing demos and refutations. Similar to the demos for K* (Section 4.3), we handle the well-foundedness conditions for eventualities using inductively defined fulfillment relations. The pruning system employs the demo conditions as pruning rules. Consequently, the demo conditions determine the rules for pruning refutations. We design the notion of demos such that the resulting pruning refutations can be translated to Hilbert refutations as well as derivations of a Gentzen system for CTL [BL08](cf. Chapter 6). To allow for the translation to the Gentzen system, the fulfillment conditions for eventualities need to be relaxed. Relaxed fulfillment arises naturally from a detailed analysis of the fragment-based model construction given by Emerson [Eme90].

5.3.1 Support, Request, and Demand

We continue to use signed formulas and clauses (cf. Section 3.3.1). Recall that negative signs can be thought of as top-level negations. In particular, we have the following equivalences:

$$
\begin{aligned}
\lfloor \Box s^- \rfloor &\leftrightarrow \Diamond \neg s \\
\lfloor \mathsf{A}(s\,\mathsf{U}\,t)^- \rfloor &\leftrightarrow \mathsf{E}(\neg s\,\mathsf{R}\,\neg t) \\
\lfloor \mathsf{A}(s\,\mathsf{R}\,t)^- \rfloor &\leftrightarrow \mathsf{E}(\neg s\,\mathsf{U}\,\neg t)
\end{aligned}
$$

That is, universal eventualities appear as positive always-until formulas whereas existential eventualities (exists-until formulas) appear as negative always-release formulas.

As before, we call formulas of the form $\bot^\sigma$, p^σ, and $\Box s^\sigma$ **literals** and refer to locally consistent literal clauses as **base clauses**. The **support relation** $C \rhd s$ between base clauses C and signed formulas s^σ is defined recursively as follows:

$$\begin{array}{rcll}
C \rhd s^\sigma & := & s^\sigma \in C & \text{if } s^\sigma \text{ is a literal} \\
C \rhd (s \to t)^+ & := & C \rhd s^- \;\vee\; C \rhd t^+ & \\
C \rhd (s \to t)^- & := & C \rhd s^+ \;\wedge\; C \rhd t^- & \\
C \rhd \mathsf{A}(s \,\mathsf{U}\, t)^+ & := & C \rhd t^+ \;\vee\; (C \rhd s^+ \;\wedge\; \Box\mathsf{A}(s \,\mathsf{U}\, t)^+ \in C) & \\
C \rhd \mathsf{A}(s \,\mathsf{U}\, t)^- & := & C \rhd t^- \;\wedge\; (C \rhd s^- \;\vee\; \Box\mathsf{A}(s \,\mathsf{U}\, t)^- \in C) & \\
C \rhd \mathsf{A}(s \,\mathsf{R}\, t)^+ & := & C \rhd t^+ \;\wedge\; (C \rhd s^+ \;\vee\; \Box\mathsf{A}(s \,\mathsf{R}\, t)^+ \in C) & \\
C \rhd \mathsf{A}(s \,\mathsf{R}\, t)^- & := & C \rhd t^- \;\vee\; (C \rhd s^- \;\wedge\; \Box\mathsf{A}(s \,\mathsf{R}\, t)^- \in C) &
\end{array}$$

The cases for path formulas follow the equivalences (5.1) and (5.2). We continue to use the abbreviations from before:

$$C \rhd D := \forall s \in D.\, C \rhd s \qquad\qquad S \rhd C := \exists D \in S.\, D \rhd C$$

Let C be a clause. The **request** of C is defined as before, i.e.,

$$\mathcal{R}\,C := \{\, s^+ \mid \Box s^+ \in C \,\}$$

The **demands** of C are defined to be the following set of clauses.

$$\mathsf{Dem}\,C := \{\, \mathcal{R}\,C, s^- \mid \Box s^- \in C \,\} \cup \{\mathcal{R}\,C\}$$

The request of C and the demands of C complement each other. If a state satisfies C, then every successor must satisfy the request of C and every demand of C must be satisfied by some successor. The singleton $\{\mathcal{R}\,C\}$ ensures that $\mathsf{Dem}\ C$ is nonempty and captures the fact that we are only interested in serial models.

Let C and D be clauses. If $D \rhd \mathcal{R}\,C$, we say that there is a **possible transition** from C to D or that D is a **possible successor** of C. A demo will be a set of base clauses S that can be turned into a finite model where the states are clauses from S, possibly appearing multiple times, and the transition relation is a selection of possible transitions. The model will be constructed such that every state satisfies all formulas it supports.

For S to be a demo, we require that every demand of a clause in S is again supported by S. This ensures that every clause in S has enough possible successors in S to satisfy all literals of the form $\Box s^-$. Together with the definition of support, this also takes care of the fixpoint conditions for path formulas. It remains to take care of the well-foundedness conditions for eventualities.

Similar to demos for K* (Definition 4.3.1) we handle the well-foundedness conditions for eventualities using inductively defined fulfillment relations. We consider two variants of inductive fulfillment. We first present the notion of inductive fulfillment arising naturally with the inductive interpretation of eventualities. This yields a notion of demo. We then obtain relaxed demos by relaxing the fulfillment conditions while retaining the property that demos contain only satisfiable clauses.

5.3.2 Standard Fulfillment

We inductively define **fulfillment relations** $S, C \rhd e$ between sets of clauses S, clauses C and eventuality literals e.

A clause C fulfills $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$ in S if either $S \rhd \mathcal{R}\,C, t^-$ (i.e., the eventuality can be fulfilled in one step) or there exists a possible successor D of C such that $D \rhd \mathcal{R}\,C, t^-$ and D fulfills $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$. The rules defining $S, C \rhd \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$ inductively are:

$$\frac{S \rhd \mathcal{R}\,C, t^-}{S, C \rhd \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-} \qquad\qquad \frac{D \in S \qquad D \rhd \mathcal{R}\,C, s^- \qquad S, D \rhd \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-}{S, C \rhd \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-}$$

To fulfill the eventuality literal $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$, every successor needs to fulfill $\mathsf{A}(s\,\mathsf{U}\,t)^+$. For the model construction, we therefore want to introduce as few successors as possible. However, we need to introduce at least one successor for every demand of the clause containing $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$. Thus, we need to ensure that one of the possible successors for every demand fulfills the eventuality.

A clause C fulfills $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$ in S if for every demand E of C either $S \rhd E, t^+$ (i.e., the eventuality can be fulfilled in one step) or there exists some clause $D \in S$ which supports E, s^+ and fulfills the eventuality. This leads to the following inductive definition:

$$\frac{\forall E \in \mathsf{Dem}\,C.\ S \rhd E, t^+ \vee (\exists D \in S.\ D \rhd E, s^+ \wedge S, C \rhd \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+)}{S, C \rhd \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+}$$

Based on the inductive fulfillment relations, we define the following notion of demo.

Definition 5.3.1 A set of base clauses S is a **demo** if the following conditions are satisfied:

D1. If $C \in S$ and $D \in \mathsf{Dem}\,C$, then $S \rhd D$.

D2. If $e \in C \in S$ for some eventuality literal e, then $S, C \rhd e$.

Note that for clauses containing multiple eventualities, the derivations for fulfillment may employ a different selection of possible successors for each eventuality. Consequently, the construction of models from demos treats each combination of a clause and an eventuality on its own, thus requiring the duplication of clauses.

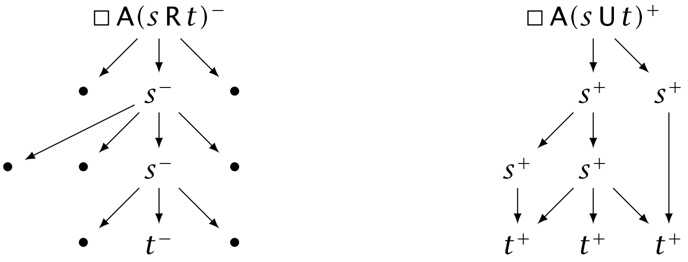


Figure 5.2: Fragments for $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$ and $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$

5.3.3 Relaxed Fulfillment

Let $\mathcal{S}$ be a demo. Derivations of $\mathcal{S}, C \triangleright e$ (with sharing of common subderivations) can be thought of as rooted directed acyclic graphs whose nodes are labeled with clauses from $\mathcal{S}$ and where the root satisfies the eventuality literal e. These rooted dags correspond to the fragments employed by Emerson and Halpern [EH85, Eme90]. Depending on the type of eventuality, fragments take one of two shapes as depicted in Figure 5.2. A fragment for $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$ is a path fulfilling the eventuality enriched with sufficiently many successors (depicted as •) to satisfy the demands of all internal nodes. A fragment for $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$ is a dag where every leaf supports t^+ and the internal nodes (except possibly the root) support s^+.

Following Emerson [Eme90], fragments can be assembled into a model where every state satisfies its labeling clause. It turns out that the construction of models from fragments does not make use of the fact that the internal nodes of a fragment are labeled with clauses from the demo. It suffices to know that the internal nodes of a fragment are labeled with base clauses. This allows us to relax the notion of fulfillment.

We distinguish between clauses that may serve as leaves of fragments ($\mathcal{S}$) and a sufficiently large set of base clauses that may serve as intermediate states ($\mathcal{T}$). We inductively define **relaxed fulfillment relations** between pairs of sets of base clauses $(\mathcal{S}, \mathcal{T})$, clauses C, and eventuality literals. The rules are given in Figure 5.3. Relaxed fulfillment differs from normal fulfillment in the quantification over $\mathcal{T}$ for the case where the eventuality is deferred. Moreover, we need to internalize the demo condition (D1) into the rules for $\mathsf{A}(s\,\mathsf{R}\,t)^-$ to ensure that the demands of all internal clauses are met (cf. Figure 5.2).

Relaxed fulfillment yields a notion of relaxed demo.

Definition 5.3.2 A pair of finite sets of base clauses $(\mathcal{S}, \mathcal{T})$ is a called a **relaxed demo**, if the following conditions hold:

R1. $\mathcal{S} \subseteq \mathcal{T}$.

R2. If $C \in \mathcal{S}$, and $D \in \mathsf{Dem}\, C$ then $\mathcal{S} \triangleright D$.

R3. If $e \in C$ is an eventuality literal and $C \in \mathcal{S}$, then $(\mathcal{S}, \mathcal{T}), C \rhd e$.

$$\frac{\begin{array}{c}\forall D \in \mathsf{Dem}\, C.\mathcal{S} \triangleright D \\ S \triangleright \mathcal{R}\, C, t^- \end{array}}{(\mathcal{S}, \mathcal{T}), C \rhd \Box\, \mathsf{A}(s\, \mathsf{R}\, t)^-} \qquad \frac{\begin{array}{c} \forall D \in \mathsf{Dem}\, C.\mathcal{S} \triangleright D \\ E \in \mathcal{T} \qquad E \triangleright \mathcal{R}\, C, s^- \qquad (\mathcal{S}, \mathcal{T}), E \rhd \Box\, \mathsf{A}(s\, \mathsf{R}\, t)^- \end{array}}{(\mathcal{S}, \mathcal{T}), C \rhd \Box\, \mathsf{A}(s\, \mathsf{R}\, t)^-}$$

$$\frac{\forall D \in \mathsf{Dem}\, C.\mathcal{S} \triangleright D, t^+ \vee (\exists E \in \mathcal{T}.E \triangleright D, s^+ \wedge (\mathcal{S}, \mathcal{T}), E \rhd \Box\, \mathsf{A}(s\, \mathsf{U}\, t)^+)}{(\mathcal{S}, \mathcal{T}), C \rhd \Box\, \mathsf{A}(s\, \mathsf{U}\, t)^+}$$

Figure 5.3: Relaxed fulfillment

Let $(\mathcal{S}, \mathcal{T})$ be a relaxed demo. Note that the set $\mathcal{T}$ may contain unsatisfiable clauses since the only requirement on the clauses in $\mathcal{T}$ is local consistency. Consequently, the construction of models from relaxed demos will only establish satisfiability of the clauses in $\mathcal{S}$. Hence, the set $\mathcal{S}$ is to be seen as the demo proper whereas $\mathcal{T}$ is a sufficiently large set of auxiliary clauses. The reason for including $\mathcal{T}$ in the definition is that the upper-bound on the size of the model for $(\mathcal{S}, \mathcal{T})$ will depend on the size of $\mathcal{T}$. When instantiating the model construction, we will take $\mathcal{T}$ to be the set of all base clauses over a given subformula universe.

Relaxed demos subsume demos in the sense that $(\mathcal{S}, \mathcal{S})$ is a relaxed demo whenever $\mathcal{S}$ is a demo. We remark that the change from normal demos to relaxed demos does not complicate the construction of models. The only real difference between the model construction for relaxed demos (next section) and a model construction for normal demos is the need to distinguish between the two components of relaxed demos.

5.4 Model Construction

We now construct models for relaxed demos. The construction consists of two parts. Fist we unwind the demo into a collection of graphs called fragments such that each fragment fulfills one eventuality in one clause. Following [Eme90], we then assemble the fragments into a finite model such that every state satisfies its labeling clause.

5.4.1 Demos to Fragments

We fix a relaxed demo $(\mathcal{S}, \mathcal{T})$ for the rest of this section and define $V := \bigcup_{C \in \mathcal{S}} C$ to be the clause containing all formulas appearing in the clauses of $\mathcal{S}$.

A **fragment** is a finite, rooted, and acyclic directed graph labeled with clauses. If G is a fragment, we write $x \in G$ if x is a node of G and $x \Rightarrow_G y$ if there is a G-edge from x to y. A node $x \in G$ is **internal** if it has some successor and a **leaf** otherwise. If $x \in G$, we write $l(x)$ for the clause labeling x. We also write x_{root} for the root of a graph if the graph can be inferred from the context. A fragment is **nontrival** if its root is not a leaf.

Let $C \in \mathcal{S}$ be a clause. A fragment G is a **fragment for C** if:

F1. If $x \in G$ is a leaf, then $l(x) \in \mathcal{S}$ and $l(x) \in \mathcal{T}$ otherwise.

F2. The root of G is labeled with C.

F3. If $x \Rightarrow_G y$, then $l(y) \rhd \mathcal{R}(l(x))$.

F4. If $x \in G$ is internal and $D \in \mathsf{Dem}\,(l(x))$, then there exists some $y \in G$ such that $x \Rightarrow_G y$ and $l(y) \rhd D$.

Let u be a signed formula. A fragment G for C is a **fragment for C and u** if:

E1. If $u = \Box\mathsf{A}(s\,\mathsf{U}\,t)^+ \in C$ and $C \rhd s^+$, then $l(x) \rhd s^+$ for every internal $x \in G$ and $l(y) \rhd t^+$ for all leaves $y \in G$.

E2. If $u = \Box\mathsf{A}(s\,\mathsf{R}\,t)^- \in C$ and $C \rhd s^-$, then $l(x) \rhd s^-$ for every internal $x \in G$ and $l(y) \rhd t^-$ for some $y \in G$.

Fragments are designed such that whenever $u \in C$ is an eventuality literal, then the root of every fragment for C and u satisfies the corresponding eventuality if it is supported. Note that if u is not an eventuality literal, then every fragment for C is also a fragment for C and u.

Definition 5.4.1 A **fragment demo** (for $(\mathcal{S},\mathcal{T})$) is an indexed collection of non-trivial fragments $(G(u,C))_{u\in V, C\in\mathcal{S}}$ where each $G(u,C)$ is a fragment for C and u.

We now show how to unfold $(\mathcal{S},\mathcal{T})$ into a fragment demo. For the construction we separate the clauses of $\mathcal{T}$ that can be used to show fulfillment into levels. Let $C \in \mathcal{T}$ such that $(\mathcal{S},\mathcal{T}), C \rhd \Box\mathsf{A}(s\,\mathsf{U}\,t)^+$. The **level** of C is the minimum depth of derivations of $(\mathcal{S},\mathcal{T}), C \rhd \Box\mathsf{A}(s\,\mathsf{U}\,t)^+$. For $\Box\mathsf{A}(s\,\mathsf{R}\,t)^-$ the notion of level is defined analogously.

Lemma 5.4.2 There exists a fragment demo (for $(\mathcal{S},\mathcal{T})$) such that every fragment has most $2 \cdot |\mathcal{T}|$ nodes.

Proof We obtain a fragment for $C \in \mathcal{S}$ and $u \in V$ as follows. We distinguish three cases.

1. $u = \Box\mathsf{A}(s\,\mathsf{U}\,t)^+ \in C$ and $C \rhd s^+$. We define:

$$\begin{aligned} A_l &:= \{\, D \in \mathcal{T} \mid (\mathcal{S},\mathcal{T}), D \rhd \Box\mathsf{A}(s\,\mathsf{U}\,t)^+ \,\} \\ A_r &:= \{\, D \in \mathcal{S} \mid D \rhd t^+ \,\} \end{aligned}$$

We then define a terminating relation $\rightharpoonup$ on the disjoint (tagged) union $A_l + A_r$ such that

$$\begin{aligned} \mathsf{inl}(D) \rightharpoonup \mathsf{inl}(E) &\;\leftrightarrow\; E \rhd \mathcal{R}\,D \wedge E \rhd s^+ \wedge \mathsf{level}\,E < \mathsf{level}\,D \\ \mathsf{inl}(D) \rightharpoonup \mathsf{inr}(E) &\;\leftrightarrow\; E \rhd \mathcal{R}\,D \end{aligned}$$

There are no transitions within A_r or from A_r to A_l. Since $(\mathcal{S},\mathcal{T})$ is a relaxed demo, we have $(\mathcal{S},\mathcal{T}), C \rhd \Box\mathsf{A}(s\,\mathsf{U}\,t)^+$ and thus $C \in A_l$. The subgraph reachable from $\mathsf{inl}(C)$ yields a fragment for C and u of the required size. It is immediate that the fragment satisfies (F1–3). Conditions (F4) and (E1) follow with the definition of fulfillment.

2. $u = \Box \mathsf{A}(s \,\mathsf{R}\, t)^- \in C$ and $C \triangleright s^-$. We define:

$$A_l := \{\, D \in \mathcal{T} \mid (\mathcal{S}, \mathcal{T}), D \rhd \Box \mathsf{A}(s \,\mathsf{R}\, t)^- \,\}$$

   Similar to the case above, we define a terminating relation $\rightharpoonup$ on the disjoint union $A_l + \mathcal{S}$ such that

$$\begin{aligned} \mathsf{inl}(D) \rightharpoonup \mathsf{inl}(E) &\;\leftrightarrow\; E \triangleright \mathcal{R}\, D \wedge E \triangleright s^- \wedge \mathsf{level}\, E < \mathsf{level}\, D \\ \mathsf{inl}(D) \rightharpoonup \mathsf{inr}(E) &\;\leftrightarrow\; E \triangleright \mathcal{R}\, D \end{aligned}$$

   Again, the subgraph reachable from $\mathsf{inl}(C)$ yields a fragment for C and u of the required size. Conditions (F1–4) are easy to verify. Condition (E2) follows by induction on the level of C using the definition of fulfillment.

3. In all other cases it suffices to construct a fragment for C. The fragment consists of a root node labeled with C and a a number of leafs. For every clause $D \in \mathsf{Dem}\, C$ we add a leaf x_D such that $l(x_D) \in \mathcal{S}$ and $l(x_D) \triangleright D$. ■

Remark 5.4.3 The levels employed in the proof of Lemma 5.4.2 are similar to the ranks employed by Ben-Ari et al. [BAPM83] to define the fulfillment conditions for eventualities. Their model construction, however, follows a different approach.

5.4.2 Fragments to Models

We now show how to assemble a fragment demo into a model. The construction is adapted from Emerson's handbook article [Eme90].

We fix some fragment demo $(G(u,C))_{u \in V, C \in \mathcal{S}}$ for $(\mathcal{S}, \mathcal{T})$. We construct a finite model $\mathcal{M}$ satisfying all clauses in the fragment demo. If V is empty, there is nothing to show, so we assume that V is nonempty.

The states of $\mathcal{M}$ are the nodes of all the fragments in the demo, i.e., every state of $\mathcal{M}$ is a dependent triple (u, C, x) with $u \in V$, $C \in \mathcal{S}$, and $x \in G(u, C)$. A state (u, C, x) is labeled with atomic proposition p iff $p^+ \in l(x)$.

To define the transitions of $\mathcal{M}$, we fix an ordering $u_0, \dots, u_n$ of the signed formulas in V. We write u_{i+1} for the successor of u_i in this ordering. The successor of u_n is taken to be u_0. The transitions of $\mathcal{M}$ are of two types. First, we lift all the internal edges of the various fragments to transitions in $\mathcal{M}$. Second, if x is a leaf in $G(u_i, C_j)$, we add transitions from (u_i, C_j, x) to all successors of the root of $G(u_{i+1}, l(x))$ (cf. Figure 5.4). This leads to the following definition:

$$\begin{aligned} |\mathcal{M}| &:= \{\, (u, C, x) \mid u \in V, C \in \mathcal{S}, x \in G(u, C) \,\} \\ \Lambda_{\mathcal{M}}\, p\; (u, C, x) &:= p^+ \in l(x) \\ (u_i, C, x) \Rightarrow_{\mathcal{M}} (v, D, y) &:= (v = u_i \wedge D = C \wedge x \Rightarrow_{G(v,D)} y) \vee \\ &\qquad (\mathsf{leaf}\, x \wedge v = u_{i+1} \wedge D = l(x) \wedge y_{root} \Rightarrow_{G(v,D)} y) \end{aligned}$$

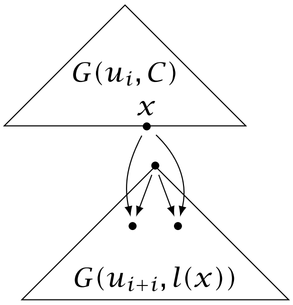


Figure 5.4: Connections between fragments

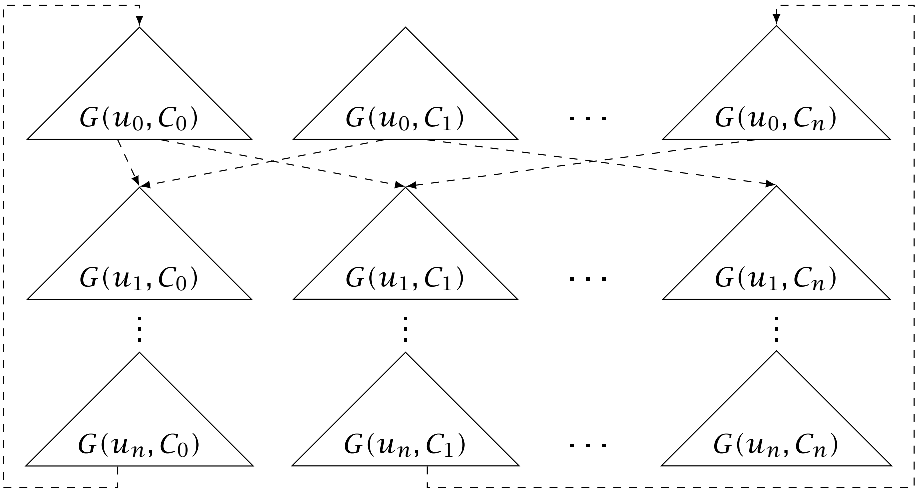


Figure 5.5: Matrix of fragments (adapted from [Eme90])

The fragments in the demo can be thought of as arranged in a matrix as shown in Figure 5.5 where the C_i are the clauses in S. Since fragment demos consist of nontrivial fragments only, the resulting transition system is serial and therefore a model. It remains to show that every state of $\mathcal{M}$ satisfies all formulas supported by its labeling clause:

Lemma 5.4.4 If $(u, C, x) \in \mathcal{M}$ and $l(x) \rhd s^\sigma$, then $\mathcal{M}, (u, C, x) \vDash s^\sigma$.

We will prove Lemma 5.4.4 by induction on s. To handle the cases for eventualities, we need a number of auxiliary notions. Analogous to the predicates AU and AR (Section 5.1.2) we define an inductive predicate EU as follows:

$$\frac{Q\,w}{\mathsf{EU}\,P\,Q\,w} \qquad\qquad \frac{P\,w \qquad w \Rightarrow_{\mathcal{M}} v \qquad \mathsf{EU}\,P\,Q\,w}{\mathsf{EU}\,P\,Q\,w}$$

We then introduce the following abbreviations:

$$\mathsf{AU}_{\mathcal{M}}\,s\,t\,w := \mathsf{AU}\,(\lambda v.l(v) \rhd s^+)\,(\lambda v.l(v) \rhd t^+)\,w$$
$$\mathsf{EU}_{\mathcal{M}}\,s\,t\,w := \mathsf{EU}\,(\lambda v.l(v) \rhd s^-)\,(\lambda v.l(v) \rhd t^-)\,w$$

Here, $l(v)$ is to be read as the label of the underlying node. After applying the induction hypothesis, the cases of Lemma 5.4.4 for the two eventualities will boil down to showing $\mathsf{AU}_{\mathcal{M}}\, s\, t\, w$ whenever $l(w) \triangleright \mathsf{A}(s\,\mathsf{U}\,t)^+$ and $\mathsf{EU}_{\mathcal{M}}\, s\, t\, w$ whenever $l(w) \triangleright \mathsf{A}(s\,\mathsf{R}\,t)^-$.

For each type of eventuality, we need two lemmas. We first show that every eventuality is satisfied whenever it is supported by the root of a fragment indexed with the corresponding eventuality literal. We then show that we can lift satisfaction of eventualities from the roots of one row of $\mathcal{M}$ to the states of the row above.

Lemma 5.4.5 Let $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \in V$ and $C \in \mathcal{S}$. Then $\mathsf{AU}_{\mathcal{M}}\, s\, t\, (\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+, C, x_{root})$ whenever $l(x_{root}) \triangleright A(s\,\mathsf{U}\,t)^+$.

Proof If $l(x_{root}) \triangleright t^+$, then the claim is trivial. Otherwise we have $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \in l(x_{root})$ and $l(x_{root}) \triangleright s^+$. Hence all internal nodes of $G(\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+, C)$ support s^+ and all leaves support t^+ (E1). The claim then follows by induction on the termination of the edge relation of $G(\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+, C)$. ▪

Lemma 5.4.6 Let $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \in V$ and $C \in \mathcal{S}$. Then $\mathsf{EU}_{\mathcal{M}}\, s\, t\, (\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-, C, x_{root})$ whenever $l(x_{root}) \triangleright A(s\,\mathsf{R}\,t)^-$.

Proof Without loss of generality, we can assume $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \in l(x_{root})$ and $l(x_{root}) \triangleright s^-$. With (E2), we obtain a $G(\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-, C)$-path from x_{root} to some state y such that $l(y) \triangleright t^-$. The claim then follows by induction on this path. ▪

It remains to lift satisfaction of eventualities from one row to the row above.

Lemma 5.4.7 Let $(u_i, C, x) \in \mathcal{M}$. Then $\mathsf{AU}_{\mathcal{M}}\, s\, t\, (u_i, C, x)$ if $l(x) \triangleright \mathsf{A}(s\,\mathsf{U}\,t)^+$ and $\mathsf{AU}_{\mathcal{M}}\, s\, t\, (u_{i+1}, D, y_{root})$ for all $D \in \mathcal{S}$ such that $D \triangleright \mathsf{A}(s\,\mathsf{U}\,t)^+$.

Proof By induction on the termination of the edge relation of $G(u_i, C)$. If x is a leaf, the claim follows since $(u_{i+1}, l(x), y_{root})$ has the same label and the same successors as x (cf. Figure 5.4). Otherwise, either $l(x) \triangleright t^+$ (and the claim follows trivially) or $l(x) \triangleright \{s^+, \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+\}$. Hence, every successor of x supports $\mathsf{A}(s\,\mathsf{U}\,t)^+$ (F3) and the claim follows by induction hypothesis. ▪

Lemma 5.4.8 Let $(u_i, C, x) \in \mathcal{M}$. Then $\mathsf{EU}_{\mathcal{M}}\, s\, t\, (u_i, C, x)$ if $l(x) \triangleright \mathsf{A}(s\,\mathsf{R}\,t)^-$ and $\mathsf{EU}_{\mathcal{M}}\, s\, t\, (u_{i+1}, D, y_{root})$ for all $D \in \mathcal{S}$ such that $D \triangleright \mathsf{A}(s\,\mathsf{R}\,t)^-$.

Proof Similar to the proof of Lemma 5.4.7 using (F4) instead of (F3). ▪

Putting everything together, we can show correctness of the model construction.

Proof (of Lemma 5.4.4) By induction on s. We consider the cases for the eventualities. The reasoning for the remaining cases follows the proof of Lemma 3.3.3.

$s^\sigma = A(s \,\mathsf{U}\, t)^+$. Let $w = (u_i, C_j, x) \in \mathcal{M}$ and assume $l(x) \rhd \mathsf{A}(s \,\mathsf{U}\, t)^+$. By induction hypothesis, it suffices to show $\mathsf{AU}_{\mathcal{M}}\, s\, t\, w$. By Lemma 5.4.7, it suffices to consider the case where x is the root of $G(u_i, C_j)$. Also, we can assume without loss of generality that $\Box\,\mathsf{A}(s \,\mathsf{U}\, t)^+ \in V$. We prove $\mathsf{AU}_{\mathcal{M}}\, s\, t\, w$ by induction on the distance from u_i to $\Box\,\mathsf{A}(s \,\mathsf{U}\, t)^+$ in the ordering of V. If $u_i = \Box\,\mathsf{A}(s \,\mathsf{U}\, t)$ the claim follows with Lemma 5.4.5. Otherwise the claim follows with Lemma 5.4.7 and the (inner) induction hypothesis.

$s^\sigma = A(s \,\mathsf{R}\, t)^-$. Let $w \in M$. We need to show $w \vDash A(s \,\mathsf{R}\, t)^-$. It suffices to show $\mathsf{EU}_{\mathcal{M}}\, s\, t\, w$. This follows, similar to the case for $\mathsf{A}(s \,\mathsf{U}\, t)^+$, with Lemma 5.4.8 and Lemma 5.4.6. ∎

Putting the two parts of the model construction together, we obtain the following theorem:

Theorem 5.4.9 (Model Existence) Let $(\mathcal{S}, \mathcal{T})$ be a relaxed demo such that $V := \bigcup_{C \in \mathcal{S}} C$ is nonempty. Then there exists a model $\mathcal{M}$ with at most $2 \cdot |V| \cdot |\mathcal{T}|^2$ states such that:

$$\forall C \in \mathcal{S}\, \exists w \in \mathcal{M}\, \forall s^\sigma.\; C \rhd s^\sigma \to w \vDash s^\sigma$$

Proof Follows with Lemma 5.4.2 and Lemma 5.4.4. ∎

5.4.3 Remarks on the Formalization

The formalization of the model construction presented in this section is one of the most technical parts of the development accompanying this thesis [ACF]. We briefly mention some of the techniques used in the formalization.

Levels For the proof of Lemma 5.4.2 we employed the notion of levels. In order to obtain a decidable transition relation, these levels need to be computed. We make use of the fact that, over a given collection of clauses, the fulfillment relations correspond to the fixpoint of some monotone and bounded function F. Recall, that we construct the fixpoint of F by iterating F sufficiently often on the empty set (cf. Section 7.1). Hence, the level of a clause C can be defined as the least n such that $C \in F^{n+1}\emptyset$.[1] Note that this is only defined if $x \in \operatorname{lfp} F$. For the formalization we use a total function that takes a proof of membership in $\operatorname{lfp} F$ as an extra argument.

Let $\{x, y\} \subseteq \operatorname{lfp} F$. To formalize the proof of Lemma 5.4.2, we only need two facts to characterize levels:

$$x \in F^{\mathsf{level}\, x + 1}\emptyset$$
$$y \in F^{\mathsf{level}\, x}\emptyset \to \mathsf{level}\, y < \mathsf{level}\, x$$

We remark that, although the fulfillment relations are conceptually inductive definitions, the formal proofs only employ the fixpoint characterization of fulfillment.

[1] To be precise, the level is one less than the minimum derivation depth to avoid an empty level 0.

Matrix Construction We base our representation of fragments on finite types. We represent finite labeled graphs as relations over some finite type together with a labeling function. We then represent fragments using clause labeled graphs with a distinguished root element.

For the construction of $\mathcal{M}$, we turn the finite set $V \times S$ into a finite type I. The constructed fragment demo then corresponds to a function G from I to the type of finite graphs. The states of $\mathcal{M}$ are then obtained as the type $\Sigma i : I.\ G\, i$ (i.e, as a tagged disjoint union of the graphs indexed by I).

We lift the internal edges of the graphs $G\, i$ by defining a predicate

$$\mathsf{liftEdge}\ :\ (\Sigma i{:}I.\, G\, i) \to (\Sigma i{:}I.\, G\, i) \to \mathbb{B}$$

on the dependent pairs of an index and a node of the respective graph satisfying

$$\begin{aligned} \mathsf{liftEdge}\,(i,x)\,(i,y) &\leftrightarrow x \Rightarrow_{G\,i} y \\ i \neq j \to \neg\mathsf{liftEdge}\,(i,x)\,(j,y) \end{aligned}$$

The definition of $\mathsf{liftEdge}$ uses dependent types in a form that is well supported by Ssreflect.

The proof of Lemma 5.4.4 contains a nested induction on the distance from u_i to $\square\,\mathsf{A}(s\,\mathsf{U}\,t)$ according to the ordering of the nonempty set V. We formalize this distance using two functions $\mathsf{dist} : V \to V \to \mathbb{N}$ and $\mathsf{next} : V \to V$ satisfying

$$\begin{aligned} \mathsf{dist}\,x\,y = 0 &\to x = y \\ \mathsf{dist}\,x\,y = n+1 &\to \mathsf{dist}(\mathsf{next}\,x)\,y = n \end{aligned}$$

Note that the notion of distance defined this way is asymmetric. The definitions of dist and next employ an enumeration of V and arithmetic modulo $|V|$. The notation u_{i+1} is formalized as $\mathsf{next}\,u_i$.

Fragment Connections In [Eme90] every leaf of a fragment is *replaced* by the root with the same label on the next level. Thus, only the internal nodes of every fragment become states of the model. This would amount to using another Σ-type on the vertex type of every dag. In our model construction, we connect the leaves of one row to the successors of the equally labeled root of the next row (cf. Figure 5.4). This way, we avoid the Σ-type construction at the cost of obtaining a slightly weaker bound on the size of the constructed model. The construction makes use of the fact that CTL formulas cannot distinguish different states that are labeled with the same atomic propositions and have the same successors.

5.5 Pruning and Refutation Calculus

We now define subformula universes for CTL and show how to construct canonical relaxed demos using pruning. As before, we complement pruning with a refutation calculus whose derivations can be translated to Hilbert refutations.

We call a clause U a **subformula universe**, if it satisfies the following closure conditions:

S1. If $(s \to t)^\sigma \in U$, then $\{s^{\overline{\sigma}}, t^\sigma\} \subseteq U$.

S2. If $\Box s^\sigma \in U$, then $s^\sigma \in U$.

S3. If $\mathsf{A}(s \,\mathsf{U}\, t)^\sigma \in U$, then $\{s^\sigma, t^\sigma, \Box\,\mathsf{A}(s \,\mathsf{U}\, t)^\sigma\} \subseteq U$.

S4. If $\mathsf{A}(s \,\mathsf{R}\, t)^\sigma \in U$, then $\{s^\sigma, t^\sigma, \Box\,\mathsf{A}(s \,\mathsf{R}\, t)^\sigma\} \subseteq U$.

If C is a clause, we write $\mathsf{sfc}\, C$ for the smallest subformula universe extending C and refer to it as **subformula closure of C**. The clause $\mathsf{sfc}\, C$ can be computed using a simple structural recursion (cf. Section 4.4.1).

Lemma 5.5.1 $|\,\mathsf{sfc}\{s^\sigma\}| \le 2 \cdot |s|$.

We fix some subformula universe U for the rest of this section. As before, we write $\overline{U}$ for the set of base clauses over U. The **canonical (relaxed) demo** for U is the relaxed demo $(\mathcal{D}, \overline{U})$ where $\mathcal{D}$ contains exactly the satisfiable clauses from $\overline{U}$. We use pruning to construct the canonical demo for U:

$$\begin{aligned} p\; C\; S &:= (\forall D \in \mathsf{Dem}\; C.\; S \rhd D) \land \\ &\quad\; (\forall \Box\,\mathsf{A}(s \,\mathsf{U}\, t)^+ \in C.\; (S, \overline{U}), C \rhd \Box\,\mathsf{A}(s \,\mathsf{U}\, t)^+) \land \\ &\quad\; (\forall \Box\,\mathsf{A}(s \,\mathsf{R}\, t)^- \in C.\; (S, \overline{U}), C \rhd \Box\,\mathsf{A}(s \,\mathsf{R}\, t)^-) \\ \mathcal{D} &:= \mathsf{prune}\; p\; \overline{U} \end{aligned}$$

Lemma 5.5.2 $(\mathcal{D}, \overline{U})$ is a relaxed demo.

Proof Follows with Lemma 3.4.3. ■

We complement pruning with a refutation calculus. The rules of the calculus are given in Figure 5.6. The support rule is (save for the changed notion of support) identical to the rule for K (Section 3.4.3). For the jump rule, we need to take care of the seriality condition for CTL models. In addition, we need one loop rule per eventuality.

Lemma 5.5.3 (Refutation Completeness)

1. $\mathsf{coref}_U\, \mathcal{D}$
2. $\mathsf{ref}_U\, C$ whenever whenever $C \subseteq U$ and $\mathcal{D} \not\rhd C$.

Proof Analogous to the proof of Lemma 4.4.4 ■

Generalizing over U, we obtain the theorem below.

Theorem 5.5.4 Let U be a non-empty subformula universe and let $C \subseteq U$. Then either $\mathsf{ref}_U\, C$ or C is satisfied by a model of size at most $|U| \cdot 2^{2^{|U|+1}}$.

Proof Let $(\mathcal{D}, \overline{U})$ be the canonical demo over U. If $\mathcal{D} \not\rhd C$ we obtain a refutation of C with Lemma 5.5.3. Otherwise, we obtain a model of the required size with Theorem 5.4.9. ■

$$\frac{C \subseteq U \qquad S \not\rhd C \qquad \mathsf{coref}_U\, S}{\mathsf{ref}_U\, C}\;\mathsf{supp} \qquad\qquad \frac{D \in \mathsf{Dem}\, C \qquad \mathsf{ref}_U\, D}{\mathsf{ref}_U\, C}\;\mathsf{jump}$$

$$\frac{\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \in C \in S \subseteq \overline{U} \qquad (S, \overline{U}), C \not\rhd \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \qquad \mathsf{coref}_U\, S}{\mathsf{ref}_U\, C}\;\mathsf{loop}^+$$

$$\frac{\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \in C \in S \subseteq \overline{U} \qquad (S, \overline{U}), C \not\rhd \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \qquad \mathsf{coref}_U\, S}{\mathsf{ref}_U\, C}\;\mathsf{loop}^-$$

$$\mathsf{coref}_U\, S := \forall C \in \overline{U}.\; C \notin S \to \mathsf{ref}_U\, C$$

Figure 5.6: Refutation calculus for CTL

5.6 Completeness of the Hilbert System

We now show that the refutation rules (Figure 5.6) are admissible for the Hilbert system. We fix some subformula universe U. As before, we call a set of clauses S **Hilbert corefutable** if $\vdash \neg C$ whenever $C \in \overline{U}$ and $C \notin S$.

To show admissibility of the rules, we need a few simple facts about the Hilbert system.

Lemma 5.6.1

1. $\vdash \mathsf{A}(s\,\mathsf{U}\,t) \leftrightarrow t \vee (s \wedge \Box\,\mathsf{A}(s\,\mathsf{U}\,t))$
2. $\vdash \mathsf{A}(s\,\mathsf{R}\,t) \leftrightarrow t \wedge (s \vee \Box\,\mathsf{A}(s\,\mathsf{R}\,t))$
3. If $D \in \mathsf{Dem}\; C$, then $\vdash C \to \Diamond D$.

Admissibility of the jump-rule is an immediate consequence of Lemma 5.6.1(3).

Lemma 5.6.2 (Admissibility of Jump Rule) Let $D \in \mathsf{Dem}\, C$. Then $\vdash \neg C$ if $\vdash \neg D$.

For the support rule, we show that every clause implies the disjunction over its base in S.

Lemma 5.6.3 Let S be Hilbert corefutable. Then $\vdash C \to \bigvee \mathcal{B}_S\, C$.

Proof Similar to the proof of Lemma 3.5.2 using Lemma 5.6.1. ▪

Lemma 5.6.4 (Admissibility of Support Rule) Let $S \subseteq \overline{U}$ be Hilbert corefutable and let $C \subseteq U$. Then $\vdash \neg C$ whenever $S \not\rhd C$.

Proof Immediate with Lemma 5.6.3. ▪

It remains to show admissibility of the loop rules. The proofs employ the induction rules of the Hilbert system. We do not use the rules UI and RI directly. Instead, we use derived rules tailored for the refutation of eventuality literals.

Lemma 5.6.5

1. If $\vdash u \to \Box(t \wedge (\neg s \to u))$, then $\vdash u \to \Box\,\mathsf{A}(s\,\mathsf{R}\,t)$.
2. If $\vdash u \to \Diamond(\neg t \wedge (s \to u))$, then $\vdash u \to \neg\Box\,\mathsf{A}(s\,\mathsf{U}\,t)$.

Proof We prove (2), the proof for (1) is similar but simpler. Assume $\vdash u \to \Diamond(\neg t \wedge (s \to u))$. Let $u' := \neg(\neg t \wedge (s \to u))$. We reason as follows:

$$\begin{aligned}
& \vdash u \to \neg\Box\,\mathsf{A}(s\,\mathsf{U}\,t) && \\
\Leftarrow\quad & \vdash \neg\Box u' \to \neg\Box\,\mathsf{A}(s\,\mathsf{U}\,t) && \text{assumption, def-}\Diamond \\
\Leftarrow\quad & \vdash \mathsf{A}(s\,\mathsf{U}\,t) \to u' && \text{prop., Reg} \\
\Leftarrow\quad & \vdash t \to u' \text{ and } \vdash s \to \Box u' \to u' && \text{UI}
\end{aligned}$$

The left claim follows by propositional reasoning. The right claim is equivalent to $\vdash s \to (\neg t \wedge s \to u) \to \Diamond(\neg t \wedge s \to u)$ and hence to $\vdash s \to (\neg t \wedge s \to u) \to u$. ∎

Note that Lemma 5.6.5(2) turns the induction rule for $\mathsf{A}(s\,\mathsf{U}\,t)$ into a coinductive refutation rule. We instantiate the coinduction rules established above with elaborate invariants to refute clauses with unsupported eventuality literals. Similar invariants appear in the completeness proofs of the Hilbert systems for UB [BAPM83] and CTL [EH85, Eme90].

Lemma 5.6.6 (Admissibility of loop$^-$) Let $S \subseteq \overline{U}$ be Hilbert corefutable and let $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \in C \in S$. Then $\vdash \neg C$ whenever $(S, \overline{U}), C \not\prec \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$

Proof Assume $(S, \overline{U}), C \not\prec \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$. We define

$$I := \{\, D \in S \mid (S, \overline{U}), D \not\prec \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \,\} \qquad\qquad u := \bigvee I$$

We clearly have $\vdash C \to u$ and $\vdash C \to \neg\Box\,\mathsf{A}(s\,\mathsf{R}\,t)$. Hence, it suffices to show $\vdash u \to \Box\,\mathsf{A}(s\,\mathsf{R}\,t)$. We start to reason as follows:

$$\begin{aligned}
& \vdash u \to \Box\,\mathsf{A}(s\,\mathsf{R}\,t) && \\
\Leftarrow\quad & \vdash u \to \Box(t \wedge (\neg s \to u)) && \text{Lemma 5.6.5(1)} \\
\Leftarrow\quad & \vdash D \to \Box(t \wedge (\neg s \to u)) && D \in I
\end{aligned}$$

Since $D \in I$ we have $(S, \overline{U}), D \not\prec \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^-$. By the definition of relaxed fulfillment, there are two possibilities. It is possible that there exists some clause E such that $E \in \mathsf{Dem}\ D$ and $S \ntriangleright E$. In this case the claim follows since D is refutable (Lemma 5.6.1(3)). Otherwise we have:

$$S \ntriangleright \mathcal{R}\,D, t^- \tag{5.3}$$

$$\mathcal{B}_S\,(\mathcal{R}\,D, s^-) \subseteq I \tag{5.4}$$

We continue as follows: Since $\Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- \in C \in I$, it suffices to show:

$$\begin{aligned}
& \vdash D \to \Box(t \wedge (\neg s \to u)) && \\
\Leftarrow\quad & \vdash \Box(\mathcal{R}\,D) \to \Box(t \wedge (\neg s \to u)) && \text{Lemma 3.5.4(3)} \\
\Leftarrow\quad & \vdash \mathcal{R}\,D \to t \text{ and } \vdash \mathcal{R}\,D \wedge \neg s \to u && \text{Lemma 3.5.4(1)}
\end{aligned}$$

The left claim is equivalent to $\vdash \neg \mathcal{R}\, D, t^-$ and follows with (5.3) and Lemma 5.6.4. Similarly, the right claim is equivalent to $\vdash \mathcal{R}\, D, s^- \to u$ and follows with (5.4) and Lemma 5.6.3. ■

Lemma 5.6.7 (Admissibility of loop$^+$) Let $\mathcal{S} \subseteq \overline{U}$ be Hilbert corefutable and $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \in C \in \mathcal{S}$. Then $\vdash \neg C$ whenever $(\mathcal{S}, \overline{U}), C \not\rhd \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$.

Proof Assume $(\mathcal{S}, \overline{U}), C \not\rhd \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$. We define

$$I := \{\, D \in \mathcal{S} \mid (\mathcal{S}, \overline{U}), D \not\rhd \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \,\} \qquad\qquad u := \bigvee I$$

Similar to proof of Lemma 5.6.6, it suffices to show $\vdash u \to \neg\Box\,\mathsf{A}(s\,\mathsf{U}\,t)$. For every $D \in I$ there must exist some clause $E \in \mathsf{Dem}\ D$ such that

$$\mathcal{S} \not\triangleright E, t^+ \tag{5.5}$$

$$\mathcal{B}_{\mathcal{S}}\,(E, s^+) \subseteq I \tag{5.6}$$

since otherwise we would have $(\mathcal{S}, \overline{U}), D \rhd \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$. We reason as follows:

$$\begin{array}{lll}
 & \vdash u \to \neg\Box\,\mathsf{A}(s\,\mathsf{U}\,t) & \\
\Leftarrow & \vdash u \to \Diamond(\neg t \land (s \to u)) & \text{Lemma 5.6.5} \\
\Leftarrow & \vdash D \to \Diamond(t \land (s \to u)) & D \in I \\
\Leftarrow & \vdash \Diamond E \to \Diamond(\neg t \land (s \to u)) & \text{Lemma 5.6.1(3), } E \in \mathsf{Dem}\, D \text{ as above} \\
\Leftarrow & \vdash \neg E, t^+ \text{ and } \vdash E, s^+ \to u & \text{Lemma 3.5.4(1), prop.}
\end{array}$$

The left claim follows with (5.5) and Lemma 5.6.4, the right claim follows with (5.4) and Lemma 5.6.3. ■

Lemma 5.6.8 $\vdash \neg C$ whenever $\mathsf{ref}_U\, C$.

Proof By induction on $\mathsf{ref}_U\, C$ using the lemmas above. ■

Theorem 5.6.9 (Informative Completeness) Let s be a formula. Then either $\vdash \neg s$ or s is satisfied by a model with at most $|s| \cdot 2^{4 \cdot |s| + 2}$ states.

Proof Let $U := \mathsf{sfc}\{s^+\}$. By Theorem 5.5.4, we either have $\mathsf{ref}_U\ \{s^+\}$ and therefore $\vdash \neg s$ (Lemma 5.6.8) or we obtain a model of the required size (Lemma 5.5.1). ■

Together with soundness of the Hilbert system, we obtain the following corollaries:

Corollary 5.6.10 (Completeness) If s is valid, then $\vdash s$.

Corollary 5.6.11 (Decidability) Satisfiability, validity, and Hilbert provability of formulas are decidable.

Corollary 5.6.12 (Small Models) Let s be satisfied by a classical model. Then s is satisfied by a finite model with at most $|s| \cdot 2^{4 \cdot |s| + 2}$ states.

5.7 Remarks on the Completeness Proof

Demos serve as the interface connecting the construction of models with the construction of refutations. Our demos are designed to give a good compromise between minimizing the effort for the model construction and minimizing the effort of constructing Hilbert refutations. As it turns out, the definition of fulfillment for eventualities has a large impact on the complexity of the proofs. For the model construction one needs to construct fragments for fulfilled eventualities, and for the construction of Hilbert refutations one needs to establish closure properties for the collection of clauses that do not fulfill a given eventuality.

The proof of Lemma 5.4.2 employs a "bottom-up" fragment construction using levels. This is mainly motivated by the desire to obtain "small" fragments, i.e., fragments without duplicate labels among internal nodes. As a result, we obtain (up to constants) the same upper bound for the small-model property (Corollary 5.6.12) that has been established by Emerson and Halpern [EH85].

In Emerson's handbook article [Eme90] the fulfillment condition for eventualities requires the existence of fragments embedded in the demo (there called pseudo-Hintikka structure). Thus, the existence of small fragments for the model construction is immediate. However, the closure conditions for clauses with unfulfilled eventualities are more difficult to obtain. For $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ \in C$ one needs to show that if there were embedded fragments fulfilling $\mathsf{A}(s\,\mathsf{U}\,t)^+$ for every demand of C, then there would also exist an embedded fragment for C and $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$. While it is easy to see that the fragments for the successors can be combined into a fragment for C and $\Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+$, showing that the resulting fragment can be embedded into the demo requires nodes with identical labels to be merged.

For our definition of demos, the closure properties required to refute unfulfilled eventualities are easy to obtain. On the other hand, we have to spend some effort to obtain fragments for fulfilled eventualities. We believe that the declarative fragment construction described in the proof of Lemma 5.4.2 is easier to formalize than the iterative process of merging nodes with identical labels used by Emerson [Eme90].

Another difference between our fulfillment predicates and the fragment test by Emerson is that we define fulfillment for eventuality literals. This is technically convenient because it ensures that all arising fragments are nontrivial. It also allows us to avoid the problem of eventualities that are not fulfilled but could be fulfilled locally in a consistent extension of the clause under consideration. Emerson and Halpern [EH85, Eme90] solve this problem by defining demos only for maximal clauses. This requires subformula universes that are closed under adding or removing top-level negations and requires demos to be defined relative to a given subformula universe. Neither is the case for our definition of demo.

We remark that for the purpose of this discussion, the difference between standard demos and relaxed demos is insignificant since the effects on the proofs presented in this chapter are marginal. Relaxed demos and the associated refutation calculus are designed with the completeness proof for the Gentzen system [BL08] in mind (cf. Chapter 6). The translation to Hilbert refutations does not profit from relaxed demos. One can obtain a marginally simpler translation to Hilbert refutations when starting from the refutation calculus arising with standard demos (Definition 5.3.1) [DS15]. Since the rules for $S, C \triangleright \Box\mathsf{A}(s\,\mathsf{R}\,t)^-$ do not require the premise for $\mathsf{Dem}\ C$ present in the rules for $(S, \mathcal{T}), C \triangleright \Box\mathsf{A}(s\,\mathsf{R}\,t)^-$, the corresponding case distinction in the proof of Lemma 5.6.6 disappears.

5.8 Alternative Hilbert Systems

There are several different Hilbert systems for CTL in the literature. The first system was given by Emerson and Halpern [EH82, EH85] and later simplified in [Eme90]. Goldblatt [Gol91] gives a Hilbert system with induction rules similar to the rules UI and RI (cf. Figure 5.1). The main difference between Goldblatt's axiomatization and **IC** is that in [Gol91] the fixpoint properties of path formulas are axiomatized using equivalences while we axiomatize only one direction and establish the converse direction using the induction rules (Lemma 5.6.1). Lange and Stirling [LS01] derive another Hilbert system from a game-theoretic interpretation of CTL.

We show that the systems given in [Eme90] and [LS01] are equivalent to the system **IC**. We continue to work with the minimal syntax and the abbreviations defined in Section 5.1. Moreover, we introduce the following additional abbreviation: $\Box^* s := \mathsf{A}(\bot\,\mathsf{R}\,s)$.

5.8.1 Hilbert system of Emerson

We adapt the system from [Eme90] to our syntax and refer to the resulting system as **E**. The rules and axioms are given in Figure 5.7. We omit axioms that correspond to definitions in our setting (e.g., $\mathsf{AF}\,s \leftrightarrow \mathsf{A}(\top\,\mathsf{U}\,s)$ and $\mathsf{EF}\,s \leftrightarrow \mathsf{E}(\top\,\mathsf{U}\,s)$). Moreover, we omit the induction axioms for AF and EF since these are merely specializations of Axioms 6 and 7. Further, we need to add Axiom 10 since the remaining axioms only characterize the abbreviation $\mathsf{E}(s\,\mathsf{U}\,t)$. (Axiom 9 is also present in [Eme90].)

Besides the fact that **E** axiomatizes $\mathsf{E}(s\,\mathsf{U}\,t)$ rather than $\mathsf{A}(s\,\mathsf{R}\,t)$, there are a number of other differences between the Hilbert systems **E** and **IC** (cf. Figure 5.1):

- The system **E** features neither the normality scheme (N) nor the necessitation rule (Nec), i.e., it does not extend an axiomatization of K.
- The system **E** employs induction axioms rather than rules.
- Axiom 6 is weaker than the corresponding rule UI in the sense that the premises of Axiom 6 do not mention the formula s.

Theorem 5.8.1 $\mathbf{E} \vdash s$ whenever $\mathbf{IC} \vdash s$.

1. Sufficiently many propositional tautologies (i.e., K, S, and DN)
2. $\Diamond(s \vee t) \leftrightarrow \Diamond s \vee \Diamond t$
3. $\mathsf{E}(s\,\mathsf{U}\,t) \leftrightarrow t \vee (s \wedge \Diamond\,\mathsf{E}(s\,\mathsf{U}\,t))$
4. $\mathsf{A}(s\,\mathsf{U}\,t) \leftrightarrow t \vee (s \wedge \Box\,\mathsf{A}(s\,\mathsf{U}\,t))$
5. $\Diamond\top \wedge \Box\top$
6. $\Box^*(u \to \neg t \wedge \Diamond u) \to u \to \neg\,\mathsf{A}(s\,\mathsf{U}\,t)$
7. $\Box^*(u \to \neg t \wedge (s \to \Box u)) \to u \to \neg\,\mathsf{E}(s\,\mathsf{U}\,t)$
8. $\Box^*(s \to t) \to \Diamond s \to \Diamond t$
9. $\Box s \leftrightarrow \neg\Diamond\neg s$
10. $\mathsf{A}(s\,\mathsf{R}\,t) \leftrightarrow \neg\,\mathsf{E}(\neg s\,\mathsf{U}\,\neg t)$

$$\frac{\vdash s \qquad \vdash s \to t}{\vdash t}\ \mathsf{MP} \qquad\qquad \frac{\vdash s}{\vdash \Box^* s}\ \mathsf{Gen}$$

Figure 5.7: Hilbert system **E**

Proof It suffices to show that the rules and axioms of **IC** are derivable in **E**. In the following we write rules using $\Longrightarrow$ (e.g., $\vdash s, \vdash s \to t \;\Longrightarrow\; \vdash t$ for MP). We first show that the axioms and rules of the Hilbert system for K (Figure 3.1) are derivable in **E**.

$$\begin{array}{rll}
a. & \vdash s \to t \;\Longrightarrow\; \vdash \Box s \to \Box t & 8, 9, \mathsf{Gen} \\
b. & \vdash s \;\Longrightarrow\; \vdash \Box s & 5, a \\
c. & \vdash \Box s \to \Box t \to \Box(s \wedge t) & 2, 9, 8, \mathsf{Gen} \\
d. & \vdash \Box(s \to t) \to \Box s \to \Box t & c, a
\end{array}$$

It remains to establish the induction rules (UI and RI), admissibility of all other axioms is straightforward. We show the case for UI, the case for RI is similar but simpler. We first obtain a weaker version of the rule UI corresponding to Axiom 6:

$$\mathsf{UI}' \qquad \vdash t \to u, \vdash \Box u \to u \;\Longrightarrow\; \vdash \mathsf{A}(s\,\mathsf{U}\,t) \to u \qquad 6, \mathsf{Gen}$$

It remains to add the assumption s to the second premise of UI′. Assume $\vdash t \to u$ and $\vdash s \to \Box u \to u$. It suffices to show $\vdash \mathsf{A}(s\,\mathsf{U}\,t) \to \mathsf{A}(s\,\mathsf{U}\,t) \to u$. After applying the rule UI′, it remains to show $\vdash \Box(\mathsf{A}(s\,\mathsf{U}\,t) \to u) \to \mathsf{A}(s\,\mathsf{U}\,t) \to u$ (the other case is straightforward). After unfolding the second occurrence of $\mathsf{A}(s\,\mathsf{U}\,t)$ (Axiom 4), the claim follows with Fact d and the two assumptions. ■

Theorem 5.8.2 **IC** $\vdash s$ whenever **E** $\vdash s$.

Proof As above, it suffices to show admissibility of all rules and axioms. Most of these have been established before (Lemma 3.5.4 and Lemma 5.6.1). We show

1. Sufficiently many propositional tautologies (i.e., K, S, and DN)
2. $\mathsf{E}(s\,\mathsf{R}\,t) \to t \wedge (s \vee \Diamond\,\mathsf{E}(s\,\mathsf{R}\,t))$
3. $\mathsf{A}(s\,\mathsf{R}\,t) \to t \wedge (s \vee \Box\,\mathsf{A}(s\,\mathsf{R}\,t))$
4. $\Box\neg s \to \neg\Box s$
5. $\Box(s \to t) \to \Box s \to \Box t$
6. $\mathsf{A}(s\,\mathsf{U}\,t) \leftrightarrow \neg\,\mathsf{E}(\neg s\,\mathsf{R}\,\neg t)$

$$\frac{\vdash s \qquad \vdash s \to t}{\vdash t}\,\mathsf{MP} \qquad\qquad \frac{\vdash u \to t \wedge (s \vee \Diamond\,\mathsf{E}((s \vee u)\,\mathsf{R}(t \vee u)))}{\vdash u \to \mathsf{E}(s\,\mathsf{R}\,t)}\,\mathsf{ERel}$$

$$\frac{\vdash s}{\vdash \Box s}\,\mathsf{Nec} \qquad\qquad \frac{\vdash u \to t \wedge (s \vee \Box\,\mathsf{A}((s \vee u)\,\mathsf{R}(t \vee u)))}{\vdash u \to \mathsf{A}(s\,\mathsf{R}\,t)}\,\mathsf{ARel}$$

Figure 5.8: Hilbert system **LS**

the case for Axiom 6, the remaining cases are straightforward. By propositional reasoning, it suffices to show $\vdash \mathsf{A}(s\,\mathsf{U}\,t) \to u \to X$ where $X := \neg\Box^*(u \to \neg t \wedge \Diamond u)$. By the rule UI, it suffices to show $\vdash \Box(u \to X) \to u \to X$. By unfolding $\Box^*$ in second occurrence of X (Lemma 5.6.1(2)), this is equivalent to $\vdash \Box(u \to X) \to u \to (u \to \neg t \wedge \Diamond u) \to \Diamond X)$ which is a theorem of K. ■

This establishes the equivalence of **IC** and **E** and therefore the soundness and completeness of **E**.

5.8.2 Hilbert system of Lange and Stirling

As with the Hilbert system in [Eme90], we adapt the Hilbert system of Lange and Stirling [LS01] to the syntax employed here. The rules and axioms are given in Figure 5.8. As before, we omit axioms corresponding to abbreviations. Also, Axiom 6 is stated in slightly different form. In addition to the axioms and rules presented in Figure 5.8, the Hilbert system in [LS01] contains the following additional axioms.

$$\vdash \mathsf{E}(s\,\mathsf{U}\,t) \to t \wedge (s \vee \Diamond\,\mathsf{E}(s\,\mathsf{U}\,t))$$
$$\vdash \mathsf{A}(s\,\mathsf{U}\,t) \to t \wedge (s \vee \Box\,\mathsf{E}(s\,\mathsf{U}\,t))$$
$$\vdash \Box s \wedge \Box t \to \Box(s \wedge t)$$

We show that the system **LS** is complete without these axioms.

The main difference between the Hilbert systems **LS** and **IC** is the form of the “induction” rules. The rules ARel and ERel to not treat the formula u as an invariant (i.e., the premises of the rules to not involve proving $\Box u$ or $\Diamond u$). Instead, the rules allow the “weakening” of a release formula with the “current context” u whenever the release formula is unfolded. Successive applications of the rules allow weakening a given release formula to the point where it becomes trivial.

Theorem 5.8.3 **LS** $\vdash s$ whenever **IC** $\vdash s$.

Proof We show admissibility of the rule RI. The case for the UI is similar and all other cases are straightforward. Assume (a) $\vdash u \to t$ and (b) $\vdash u \to \neg s \to \Box u$. We reason as follows:

$$\begin{array}{rll} & \vdash u \to \mathsf{A}(s\,\mathsf{R}\,t) & \\ \Leftarrow & \vdash u \to t \wedge (s \vee \Box\,\mathsf{A}((s \vee u)\,\mathsf{R}(t \vee u)) & \mathsf{ARel} \\ \Leftarrow & \vdash \Box u \to \Box\,\mathsf{A}((s \vee u)\,\mathsf{R}(t \vee u)) & \text{(a),(b)} \\ \Leftarrow & \vdash u \to \mathsf{A}((s \vee u)\,\mathsf{R}(t \vee u)) & 5, \mathsf{Nec} \end{array}$$

The last claim follows by applying ARel a second time. ∎

This establishes the completeness of the Hilbert system **LS**. Since the system **LS** is sound [LS01], the converse of Theorem 5.8.3 also holds. We will give a direct syntactic argument. Given the close connection between the Hilbert system **LS** and the history-based Gentzen system for CTL [BL08] we defer the proof to the next chapter (cf. Theorem 6.2.7).

5.9 Path Semantics vs. Inductive Semantics

We now show that the inductive semantics we employ in our proofs agrees with the usual semantics for CTL defined using infinite paths [Eme90, BK08]. In fact, there are several classically equivalent but intuitionistically different formulations of the path semantics. We show that the inductive semantics agrees constructively with one formulation of the path semantics on finite models. The equivalence extends to general models if one assumes excluded middle and a weak form of choice. For other formulations of the path semantics one can show that even the equivalence on finite models requires classical assumptions.

5.9.1 Path Semantics

We define a version of the path semantics for CTL that constructively agrees with the inductive semantics on finite models. Let $\mathcal{M}$ be a general model. A **path** is a function $\pi : \mathbb{N} \to \mathcal{M}$ such hat $\pi\, n \Rightarrow_{\mathcal{M}} \pi(n+1)$ for all n. The letter π ranges over paths. The **path satisfaction relation** $w \vDash_p s$ for states w of $\mathcal{M}$ and formulas s is then defined by recursion on formulas:

$$\begin{aligned} w \vDash_p \bot &:= \bot \\ w \vDash_p p &:= \Lambda_{\mathcal{M}}\, p\, w \\ w \vDash_p s \to t &:= w \vDash_p s \to w \vDash_p t \\ w \vDash_p \Box s &:= \forall v.\ (w \Rightarrow_{\mathcal{M}} v) \to v \vDash_p s \\ w \vDash_p \mathsf{A}(s\,\mathsf{U}\,t) &:= \forall \pi.\ \pi\, 0 = w \to \exists n.\ \pi\, n \vDash_p t \wedge \forall m < n.\ \pi\, m \vDash_p s \\ w \vDash_p \mathsf{A}(s\,\mathsf{R}\,t) &:= \forall \pi.\ \pi\, 0 = w \to \forall n.\ \pi\, n \vDash_p t \vee \exists m < n.\ \pi\, m \vDash_p s \end{aligned}$$

We write $\mathcal{M} \vDash_p s$ if $w \vDash_p s$ for *all* states w of $\mathcal{M}$.

5.9.2 Agreement

We now show that the path semantics agrees with the inductive semantics on finite models. We fix some finite model $\mathcal{M}$ for the rest of this section.

Lemma 5.9.1 Let $P, Q : \mathcal{M} \to \mathbb{B}$ be decidable predicates and let $w \in \mathcal{M}$. Then

$$\mathsf{AU}\,P\,Q\,w \leftrightarrow \forall \pi. \pi\,0 = w \to \exists n. Q\,(\pi\,n) \wedge \forall m < n. P\,(\pi\,m)$$

Proof The direction from left to right follows by induction on $\mathsf{AU}\,P\,Q\,w$. Since $\mathsf{AU}\,P\,Q\,w$ is decidable, we can show the direction from right to left by showing its contrapositive. Assume $\neg\mathsf{AU}\,P\,Q\,w$. We construct a path contradicting the right hand side. Consider the following decidable subrelation of $\Rightarrow_{\mathcal{M}}$:

$$u \Rightarrow v := u \Rightarrow_{\mathcal{M}} v \wedge (\neg\mathsf{AU}\,P\,Q\,u \to P\,u \to \neg\mathsf{AU}\,P\,Q\,v)$$

Since $\Rightarrow_{\mathcal{M}}$ is serial, the relation $\Rightarrow$ is serial as well. Using constructive choice we construct a function $f : \mathcal{M} \to \mathcal{M}$ selecting for every state of $\mathcal{M}$ a $\Rightarrow$-successor. We define $\pi\,n := f^n\,w$ and show

$$\forall n. \neg\mathsf{AU}\,P\,Q\,(\pi\,n) \vee \exists m < n. \neg P\,(\pi\,m)$$

by induction on n. This yields $\neg Q\,(\pi\,n) \vee \exists m < n. \neg P\,(\pi\,m)$ for all n, contradicting the right hand side as required. ∎

Lemma 5.9.2 Let $P, Q : \mathcal{M} \to \mathbb{B}$ be decidable predicates and let $w \in \mathcal{M}$. Then

$$\mathsf{AR}\,P\,Q\,w \leftrightarrow \forall \pi. \pi\,0 = w \to \forall n. Q\,(\pi\,n) \vee \exists m < n. P\,(\pi\,m)$$

Proof For the direction from left to right,assume $\mathsf{AR}\,P\,Q\,w$ and let π be a path such that $\pi\,0 = w$. We prove

$$\forall n.\ \mathsf{AR}\,P\,Q\,(\pi\,n) \vee \exists m < n. P\,(\pi\,m)$$

by induction on n. The base case follows by assumption, the induction step by inversion on $\mathsf{AR}\,P\,Q\,(\pi\,n)$. The claim then follows since $\mathsf{AR}\,P\,Q\,(\pi\,n)$ implies $Q\,(\pi\,n)$.

For the converse direction, we abbreviate the right hand side as $\mathsf{AR}'\,w$. We first show that the path characterization satisfies the inversion properties of AR:

$$\forall v.\ \mathsf{AR}'v \to Q\,v \tag{5.7}$$

$$\forall u\,v.\ \mathsf{AR}'u \to \neg P\,u \to (u \Rightarrow_{\mathcal{M}} v) \to \mathsf{AR}'v \tag{5.8}$$

For (5.7), we use constructive choice to obtain some path through the model. Property (5.8) is easy to verify. The claim then follows by coinduction using (5.7) and (5.8). ∎

Theorem 5.9.3 (Finite Agreement) Let w be a state of $\mathcal{M}$. Then $w \vDash_p s$ iff $w \vDash s$.

Proof By induction on s using Lemma 5.9.1 and Lemma 5.9.2. ■

Together with the soundness of the Hilbert system for classical models (Theorem 5.2.1, we obtain soundness of the Hilbert system for the path semantics and finite models.

Corollary 5.9.4 (Finite Soundness) If $\vdash s$, then $\mathcal{M} \vDash_p s$ for all finite models $\mathcal{M}$.

Lemma 5.9.1 and Lemma 5.9.2 extend to infinite models if we use XM instead of decidability to justify case distinctions and DC (Section 2.2) instead of constructive choice to obtain infinite paths. This yields the theorem below.

Theorem 5.9.5 (General Agreement) Assume XM and DC and let $w \in \mathcal{M}$ for some general model $\mathcal{M}$. Then $\mathcal{M}, w \vDash_p s$ iff $\mathcal{M}, w \vDash s$.

Corollary 5.9.6 (Path Soundness) Assume XM and DC. Then $\mathcal{M} \vDash_p s$ for all general models $\mathcal{M}$ whenever $\vdash s$.

Proof By Theorem 5.9.5 it suffices to show $w \vDash s$ for all states w of $\mathcal{M}$. This follows with Theorem 5.2.1. ■

Recall that the semantics with respect to all models is essentially a shallow embedding into the type theory of Coq. As it turns out, this allows us to show that XM and DC are not only sufficient but also necessary to prove path-soundness for general models.

Fact 5.9.7 Assume $\vdash s$ implies $\mathcal{M} \vDash_p s$ for all general models $\mathcal{M}$. Then XM and DC are provable.

Proof XM follows with soundness of DN (see the proof of Fact 3.2.1). For DC, let X be a type and let $R : X \to X \to \mathsf{Prop}$ be a serial relation. The relation R defines a model (the labeling is irrelevant). It is straightforward to show $\vdash \mathsf{E}(\bot\ \mathsf{R}\ \top)$. In the presence of XM, soundness yields an infinite path as required. ■

Note that in the proof of Corollary 5.9.6, DC is only used to satisfy the premise of Theorem 5.9.5. Hence, DC is necessary to show Theorem 5.9.5.

5.9.3 Remark on Release

The path semantics of the release modality can be defined in a number of classically equivalent but intuitionistically different ways [BK08, p. 256]. The definition in Section 5.9.1 was chosen because one can show constructively, as we have done, that it coincides with the coinductive characterization on finite models. We now show that for other formulations, such as the commonly found

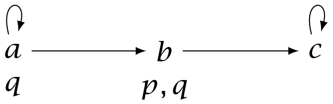
$$\begin{aligned} & w \vDash_p \mathsf{A}(s \,\mathsf{R}\, t) \\ \leftrightarrow\ & \forall \pi. \pi\, 0 = w \to (\forall n. \pi\, n \vDash_p t) \vee (\exists n. \pi\, n \vDash_p s \wedge \forall m \le n. \pi\, m \vDash_p t) \end{aligned} \tag{5.9}$$

this is not the case. We construct a finite model such that if the equivalence above were to hold in this model, we could prove a constructively non-provable proposition called "limited principle of omniscience" [Esc13].

$$\mathsf{LPO} := \forall f : \mathbb{N} \to \mathbb{B}.\ (\forall n. f\, n = \bot) \vee (\exists n. f\, n = \top)$$

Intuitively, LPO cannot be provable constructively since a constructive proof of LPO would correspond to a procedure that decides whether a given function f returns $\top$ for some argument. This would allow us to solve the halting problem (take f to be the function that checks whether a given Turing machine halts within n steps).

We construct a finite model $\mathcal{M}_3$ as follows:



Theorem 5.9.8 Assume (5.9) holds in $\mathcal{M}_3$. Then LPO is provable.

Proof It is easy to verify $a \vDash \mathsf{A}(p \,\mathsf{R}\, q)$. Hence, we have $a \vDash_p \mathsf{A}(p \,\mathsf{R}\, q)$ by Theorem 5.9.3. We fix some function $f : \mathbb{N} \to \mathbb{B}$ and show

$$(\forall n. f\, n = \bot) \vee (\exists n. f\, n = \top)$$

We define a path π that starts at a and evaluates $f\, n$ before the n-th transition. The path leaves for b if $f\, n = \top$ and otherwise stays at a. The claim then follows with the assumed equivalence (5.9). ■

Since the inductive semantics coincides with the path semantics on finite models, Theorem 5.9.8 shows that the inductive semantics and a path semantics using (5.9) as definition for always release do not coincide constructively on finite models.

6 History-Based Gentzen Systems

In this chapter we prove soundness and completeness of Gentzen systems for K* and CTL. Both systems are obtained as variants of the Gentzen system for CTL given by Brünnler and Lange [BL08]. In order to treat eventualities with local rules, the Gentzen systems employ a focusing mechanism. The focusing mechanism works by annotating eventualities with finite sets of clauses called histories. Histories record the contexts in which an eventuality has been unfolded. If the same context appears further up in the derivation, that branch of the derivation can be closed.

The original completeness proof in [BL08] works by constructing models from unsuccessful derivations. We provide an alternative completeness proof that works by translating pruning refutations to derivations of the Gentzen system. Due to the analyticity of the Gentzen systems, the precise formulation of the refutation rules is crucial for the translation to succeed. The notion of relaxed demo introduced in the previous chapter was designed with the completeness proof for the Gentzen system in mind. We show that the pruning refutations arising with relaxed demos provide exactly the structure required for a translation to the Gentzen system for CTL. This allows us to reuse the model construction used to prove the small-model property and completeness of the Hilbert system. Given the complexity of the model construction, this significantly simplifies the proof, in particular as it comes to the formalization in Coq.

To prove completeness of the Gentzen system for K*, we adapt the notion of relaxed fulfillment. We then show that, in the case of K*, pruning with respect to relaxed fulfillment still constructs canonical demos. This allows us to obtain a translation-based completeness proof for the Gentzen system without having to resort to a fragment-based model construction. In fact, no additional model construction is required to prove the completeness of either Gentzen system.

6.1 Gentzen System for K*

The main problem in designing Gentzen systems for K* and CTL is the treatment of eventualities. As argued in [BL08], a naive treatment using only the fixpoint properties of eventualities yields a sound but incomplete system. We recall this argument and explain how the history mechanism introduced in [BL08] is used to obtain complete Gentzen systems.

As with the Gentzen system for K (Section 3.6), we employ a tableau-style semantics, i.e, we treat Gentzen systems as systems deriving unsatisfiable clauses.

Recall rules of the Gentzen system for K (Figure 3.2). We extend this system to the eventualities of K* (Chapter 4). A naive approach for extending the calculus to $\Box^* s^\sigma$ might be to add rules following the characteristic equivalence for $\Box^* s$ (i.e., $\Box^* s \leftrightarrow s \wedge \Box\Box^* s$):

$$\frac{\Vdash C, s^- \qquad \Vdash C, \Box\Box^* s^-}{\Vdash C, \Box^* s^-}\,\mathsf{G}^- \qquad\qquad \frac{\Vdash C, s^+, \Box\Box^* s^+}{\Vdash C, \Box^* s^+}\,\mathsf{G}^+$$

The rules are clearly sound, but the resulting system is far from complete.

Example 6.1.1 A Gentzen proof of the "Segerberg axiom" for K* (Section 4.6) corresponds to a derivation of the clause $\{\Box^*(p \to \Box p) \to p \to \Box^* p^-\}$. The Gentzen system for K extended with the rules above fails to derive this clause. After some initial steps, it would suffice to derive the clause $\{\Box p^+, \Box\Box^*(p \to \Box p)^+, p^+, \Box\Box^* p^-\}$. Consider the following derivation attempt. (We suppress $\Vdash$ to improve readability.)

$$\dfrac{\dfrac{}{p^-, p^+, \dots} \qquad \dfrac{\dfrac{}{p^-, p^+, \dots} \qquad \dfrac{\Box^*(p \to \Box p)^+, p^+, \Box^* p^-}{\Box\Box^*(p \to \Box p)^+, \Box p^+, p^+, \Box\Box^* p^-}}{\Box\Box^*(p \to \Box p)^+, \Box p^+, p^+, \Box^* p^-}}{\dfrac{\Box\Box^*(p \to \Box p)^+, p \to \Box p^+, p^+, \Box^* p^-}{\Box^*(p \to \Box p)^+, p^+, \Box^* p^-}}$$

Note that the open branch of the derivation is exactly the clause we started with. Moreover, there is nothing else one can do aside from a few trivial permutations.

The problem with the rules given above is that the rules fail to account for the inductive character of $\Box^* s^-$, i.e., the fact that in order to satisfy $\Box^* s^-$ one must eventually reach a state satisfying s^-.

6.1.1 Histories

Intuitively, we should be able to close the last open branch in the example above since we know that the derivation can only continue to run in cycles and will never reach a satisfiable clause containing p^-. This is the purpose of the history mechanism. It allows setting a focus on some eventuality and remembering all the contexts in which the rule G^- has been applied to that eventuality. If a context reappears further up in the derivation, the branch can be closed.

A **history** is a finite set of clauses. The letter H ranges over histories. An **annotated eventuality** is a formula of the form $\Box^*_H s^-$ or $\Box\Box^*_H s^-$. The Gentzen system for K* employs **annotated clauses** $C|a$ where C is a clause and a is either an annotated eventuality or ε, signaling the absence of any annotated eventualities.

The rules of the calculus are given in Figure 6.1. We refer to the rules foc and rep as focusing and repetition rule respectively. Most rules treat annotated

$$\dfrac{}{\Vdash C, p^+, p^- | a}\;\mathsf{A} \qquad \dfrac{}{\Vdash C, \bot^+ | a}\;\mathsf{F}^+ \qquad \dfrac{\Vdash C, s^- | a \qquad \Vdash C, t^+ | a}{\Vdash C, s \to t^+ | a}\;\mathsf{I}^+ \qquad \dfrac{\Vdash C, s^+, t^- | a}{\Vdash C, s \to t^- | a}\;\mathsf{I}^-$$

$$\dfrac{\Vdash C, s^- | a \qquad \Vdash C, \Box\Box^* s^- | a}{\Vdash C, \Box^* s^- | a}\;\mathsf{G}^- \qquad \dfrac{\Vdash C, s^+, \Box\Box^* s^+ | a}{\Vdash C, \Box^* s^+ | a}\;\mathsf{G}^+$$

$$\dfrac{\Vdash \mathcal{R}\, C, s^- | \varepsilon}{\Vdash C, \Box s^- | a}\;\mathsf{X} \qquad \dfrac{\Vdash \mathcal{R}\, C | \Box^*_H s^-}{\Vdash C | \Box\Box^*_H s^-}\;\mathsf{X_H}$$

$$\dfrac{\Vdash C | \Box\Box^*_\emptyset s^-}{\Vdash C, \Box\Box^* s^- | \varepsilon}\;\mathsf{foc} \qquad \dfrac{\Vdash C, s^- | \varepsilon \qquad \Vdash C | \Box\Box^*_{H,C} s^-}{\Vdash C | \Box^*_H s^-}\;\mathsf{G_H^-} \qquad \dfrac{}{\Vdash C | \Box^*_{H,C} s^-}\;\mathsf{rep}$$

Figure 6.1: Gentzen system **GK***

$$\dfrac{\dfrac{\dfrac{}{p^-, p^+, \ldots} \qquad \dfrac{\dfrac{\dfrac{}{p^-, p^+, \ldots} \qquad \dfrac{\dfrac{}{\Box^*(p \to \Box p)^+, p^+ | \Box^*_{\{\{\Box^*(p\to\Box p)^+, p^+\}\}} p^-}\;\mathsf{rep}}{\Box p^+, \Box\Box^*(p \to \Box p)^+, p^+ | \Box\Box^*_{\{\{\Box^*(p\to\Box p)^+, p^+\}\}} p^-}\;\mathsf{X_H}}{p \to \Box p^+, \Box\Box^*(p \to \Box p)^+, p^+ | \Box\Box^*_{\{\{\Box^*(p\to\Box p)^+, p^+\}\}} p^-}\;\mathsf{I}^+}{\Box^*(p \to \Box p)^+, p^+ | \Box\Box^*_{\{\{\Box^*(p\to\Box p)^+, p^+\}\}} p^-}\;\mathsf{G}^+}{\Box^*(p \to \Box p)^+, p^+ | \Box^*_\emptyset p^-}\;\mathsf{G_H^-}}{\dfrac{\Box p^+, \Box\Box^*(p \to \Box p)^+, p^+ | \Box\Box^*_\emptyset p^-}{\Box p^+, \Box\Box^*(p \to \Box p)^+, p^+, \Box\Box^* p^- | \varepsilon}\;\mathsf{foc}}\;\mathsf{X_H}$$

Figure 6.2: Example derivation

eventualities just like regular eventualities. The history rules (bottom row) allow setting a focus on some eventuality (focusing rule). Whenever the focused eventuality is unfolded, the context is stored in the history (rule $\mathsf{G_H^-}$). If the same context is repeated further up in the derivation, the branch can be closed (repetition rule).

Example 6.1.2 The history rules allow us to prove the "Segerberg axiom". After some initial steps (cf. Example 6.1.1), it suffices to refute the annotated clause $\{\Box p^+, \Box\Box^*(p \to \Box p)^+, p^+, \Box\Box^* p^-\} | \varepsilon$. A derivation for this clause is given in Figure 6.2. (As before, we suppress $\Vdash$ to increase readability.)

The Gentzen system in Figure 6.1 is sound and complete for history-free clauses. We show soundness immediately and defer the completeness proof to Section 6.4. Even though we are mostly interested in the case of history-free clauses, we need a semantics for annotated eventualities in oder to obtain a compositional (i.e., rule by rule) soundness proof. We adapt the semantics given in [BL08].

Intuitively, a state satisfies $\Box^*_H s^-$ if a state satisfying s^- can be reached without satisfying any clause from H along the way. For the sake of simplicity, we only consider finite models. This allows us to define satisfaction for $\Box^*_H s^-$ using an inductive predicate instead of defining it as the negation of a coinductive predicate.

Let $\mathcal{M}$ be a finite model. We define

$$w \nvDash H := \forall C \in H. w \nvDash C$$

The **satisfaction relation** between states w of $\mathcal{M}$ and annotated eventualities $\Box^*_H s^-$ is defined by induction:

$$\frac{w \nvDash H \qquad w \vDash s^-}{w \vDash \Box^*_H s^-} \qquad\qquad \frac{w \nvDash H \qquad w \Rightarrow_{\mathcal{M}} v \qquad v \vDash \Box^*_H s^-}{w \vDash \Box^*_H s^-}$$

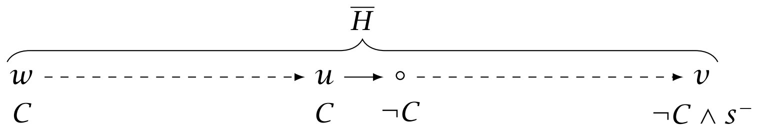
Satisfaction for $\Box\Box^*_H s^-$ is defined as one would expect:

$$w \vDash \Box\Box^*_H s^- := \exists v. w \Rightarrow_{\mathcal{M}} v \wedge v \vDash \Box^*_H s^-$$

For all rules except X, X_H, and G^-_H a local soundness argument suffices. That is, if a state satisfies the conclusion of a rule it also satisfies one of the premises of the rule. For the rules X and X_H, the premise is satisfied at a neighboring state. It turns out that the G^-_H-rule, even though it looks similar to the G^--rule, does not have this locality property.

Soundness of the rule G^-_H hinges on the fact that whenever we unfold an eventuality (a least fixpoint) we can strengthen it with the current context. This observation goes back to Kozen's work on the propositional μ-calculus [Koz83]. It also underlies the focus-games for temporal logics [LS01] which lead to the development of history-based Gentzen systems [BL08].

Intuitively, soundness of the rule G^-_H can be argued as follows. Assume $w \vDash C|\Box^*_H s^-$ for some state w of some model $\mathcal{M}$ and assume that $C, s^-|\varepsilon$ is not satisfied by any state of $\mathcal{M}$. Then there exists a path through $\mathcal{M}$ from w to some state v such that $v \vDash s^-$ and no state on this path satisfies any clause from H. Since $C, s^-|\varepsilon$ is not satisfied anywhere the state v cannot satisfy C. In particular, the path must be nontrivial. Let u be the last state on the path that satisfies C. The situation can be depicted as follows (adapted from [BL08]):



It is easy to see that u satisfies $C|\Box\Box^*_{H,C} s^-$. Note that the soundness argument is non-local in the sense that we used the fact that $C, s^-|\varepsilon$ is not satisfied at any state to construct a state u that may be arbitrarily far away from the state w

satisfying the conclusion of the rule. For the formal soundness proof below we transform the intuitive argument above into an inductive argument that avoids reasoning about paths.

Theorem 6.1.3 (Soundness) Let $\mathcal{M}$ be a finite model. If $\Vdash C|a$, then $C|a$ is not satisfied by any state of $\mathcal{M}$.

Proof By induction on $\Vdash C|a$. Soundness of all rules except the G_H^--rule is straightforward. For the G_H^--rule, we obtain two induction hypotheses:

$$\forall w \in \mathcal{M}.\ w \nvDash C, s^-|\varepsilon \tag{6.1}$$

$$\forall w \in \mathcal{M}.\ w \nvDash C|\Box\Box^*_{H,C}s^- \tag{6.2}$$

Now assume $w \vDash C$ and $w \vDash \Box^*_H s^-$ for some state w of $\mathcal{M}$. We derive a contradiction. By (6.1), we have $w \nvDash s^-$. By inversion on $w \vDash \Box^*_H s^-$, there exists some state v such that $w \Rightarrow_\mathcal{M} v$ and $v \vDash \Box^*_H s^-$. By (6.2), it suffices to show $v \vDash \Box^*_{H,C}s^-$ to obtain a contradiction. We prove $v \vDash \Box^*_{H,C}s^-$ by induction on $v \vDash \Box^*_H s^-$.

Case 1. We have $v \nvDash H$ and $v \vDash s^-$. The claim follows since $v \nvDash C$ by (6.1).

Case 2. We have $v \nvDash H$, $v \Rightarrow_\mathcal{M} u$ and $u \vDash \Box^*_{H,C}s^-$ for some state u of $\mathcal{M}$. By induction hypothesis, it suffices to show $v \nvDash C$. This follows with (6.2). ■

Remark 6.1.4 The proof of Theorem 6.1.3 is one of several examples in this thesis, where the desire to obtain a concise formalization leads to proofs that differ significantly from the informal proofs.

Recall that satisfiability with respect to classical models and finite models coincides for history-free clauses (Corollary 4.5.10 and Fact 3.5.1). Hence, the restriction to finite models only concerns "auxiliary" clauses with annotated eventualities. Also note that, just like the original system for CTL [BL08], the Gentzen system for K* is sound but not complete for arbitrary annotated clauses.

Example 6.1.5 (Incompleteness) The annotated clause $\emptyset|\Box^*_{\{\{p^-\}\}}p^-$ is clearly unsatisfiable. The only rule that applies to $\emptyset|\Box^*_{\{\{p^-\}\}}p^-$ is G_H^-. This yields $\{p^-\}|\varepsilon$ as left premise which is clearly satisfiable and therefore underivable (Theorem 6.1.3).

Before we argue completeness of the Gentzen system for K*, we first extend the system to CTL.

6.2 Gentzen System for CTL

We now consider a history-based Gentzen system for CTL (Chapter 5). The system is obtained as a variant of the sequent calculus CT [BL08] adapted to our use of signed formulas and clauses. We continue to use annotated clauses. In the case of CTL, **annotated eventualities** take one of the following four forms.

$$\mathsf{A}(s\,\mathsf{U}_H\,t)^+ \mid \Box\,\mathsf{A}(s\,\mathsf{U}_H\,t)^+ \mid \mathsf{A}(s\,\mathsf{R}_H\,t)^- \mid \Box\,\mathsf{A}(s\,\mathsf{R}_H\,t)^-$$

As for **GK***, an annotated eventuality is satisfied if the eventuality can be satisfied without satisfying any clause in the history along the way.

The request of a clause C is defined as before (i.e., $\mathcal{R}\,C := \{\, s^+ \mid \Box s^+ \in C \,\}$). Since annotations for CTL may contain positive boxes, we need to extend the notion of request to annotations. The **request** of an annotation a, written $r\,a$, is defined analogously to the request for clauses, i.e. $r\,(\Box\,\mathsf{A}(s\,\mathsf{U}_H\,t)^+) = \mathsf{A}(s\,\mathsf{U}_H\,t)^+$ and $r\,a = \varepsilon$ for all other annotations.

The Gentzen system **GCT** derives unsatisfiable annotated clauses. The rules of the system are given in Figure 6.3. As before, we write $\Vdash C|a$ if $C|a$ is derivable using the tableau rules. Compared to **GK*** (Figure 6.1) the system **GCT** features one set of history rules for each of the two eventualities of CTL. Moreover, the system features a jump rule for clauses without diamonds (rule X_s) to capture the fact that we only consider serial models for CTL. We will show the system **GCT** sound for all clauses and complete for annotation-free clauses.

Remark 6.2.1 The presentation of CT [BL08] employs sequents comprised of annotated and non-annotated formulas. This simplifies the presentation of rules, in particular for those rules that do not change the annotation. In order to obtain an analytic system, every rule of CT carries the proviso that sequents may contain at most one annotated eventuality. Our use of annotated clauses enforces this condition without an explicit side condition at the cost of a more verbose notation. Aside from these syntactic changes, the main differences between CT and **GCT** are that in CT the X-rules may only be applied to literal clauses and that all the rules carry the proviso that the active formula in the conclusion does not appear in the context. We impose no such restrictions. The reason for this is simply convenience. Our completeness proof will not make use of this added flexibility.

Recall that an annotated eventuality is satisfied if the eventuality can be satisfied without satisfying any clause in the history along the way. In the case of CTL, this can be expressed using formulas [BL08]. As a consequence we can translate derivations of **GCT** to refutations in the Hilbert System for CTL (Section 5.2). This yields a purely syntactic soundness argument and, once we have established completeness of **GCT**, an alternative completeness proof for the Hilbert system.

We define the **associated formula of a history** H to be the formula $\bigwedge_{C\in H} \neg C$. In unsigned formulas, we let $s\,\mathsf{U}_H\,t$ abbreviate $(s \wedge H)\,\mathsf{U}(t \wedge H)$. The **associated**

$$\frac{}{C, p^+, p^- | a}\,\mathsf{A} \qquad \frac{}{C, \bot^+ | a}\,\mathsf{F}^+ \qquad \frac{C, s^- | a \qquad C, t^+ | a}{C, s \to t^+ | a}\,\mathsf{I}^+ \qquad \frac{C, s^+, t^- | a}{C, s \to t^- | a}\,\mathsf{I}^-$$

$$\frac{C, t^+ | a \qquad C, s^+, \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^+ | a}{C, \mathsf{A}(s\,\mathsf{U}\,t)^+ | a}\,\mathsf{U}^+ \qquad \frac{C, t^-, s^- | a \qquad C, t^-, \Box\,\mathsf{A}(s\,\mathsf{U}\,t)^- | a}{C, \mathsf{A}(s\,\mathsf{U}\,t)^- | a}\,\mathsf{U}^-$$

$$\frac{C, s^+, t^+ | a \qquad C, t^+, \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^+ | a}{C, \mathsf{A}(s\,\mathsf{R}\,t)^+ | a}\,\mathsf{R}^+ \qquad \frac{C, t^- | a \qquad C, s^-, \Box\,\mathsf{A}(s\,\mathsf{R}\,t)^- | a}{C, \mathsf{A}(s\,\mathsf{R}\,t)^- | a}\,\mathsf{R}^-$$

$$\frac{\mathcal{R}\,C | r\,a}{C | a}\,\mathsf{X_s} \qquad \frac{\mathcal{R}\,C, u^- | r\,a}{C, \Box u^- | a}\,\mathsf{X} \qquad \frac{\mathcal{R}\,C | \mathsf{A}(s\,\mathsf{R}_H\,t)^-}{C | \Box\,\mathsf{A}(s\,\mathsf{R}_H\,t)^-}\,\mathsf{R_H^-}$$

$$\frac{C | \mathsf{A}(s\,\mathsf{U}_\emptyset\,t)^+}{C, \mathsf{A}(s\,\mathsf{U}\,t)^+ | \varepsilon}\,\mathsf{U}_\emptyset \qquad \frac{C, t^+ | \varepsilon \qquad C, s^+ | \Box\,\mathsf{A}(s\,\mathsf{U}_{H,C}\,t)^+}{C | \mathsf{A}(s\,\mathsf{U}_H\,t)^+}\,\mathsf{U_H} \qquad \frac{}{C | \mathsf{A}(s\,\mathsf{U}_{H,C}\,t)^+}\,\overline{\mathsf{U}}$$

$$\frac{C | \mathsf{A}(s\,\mathsf{R}_\emptyset\,t)^-}{C, \mathsf{A}(s\,\mathsf{R}\,t)^- | \varepsilon}\,\mathsf{R}_\emptyset \qquad \frac{C, t^- | \varepsilon \qquad C, s^- | \Box\,\mathsf{A}(s\,\mathsf{R}_{H,C}\,t)^-}{C | A(s\,\mathsf{R}_H\,t)^-}\,\mathsf{R_H} \qquad \frac{}{C | \mathsf{A}(s\,\mathsf{R}_{H,C}\,t)^-}\,\overline{\mathsf{R}}$$

Figure 6.3: Gentzen system **GCT**

formula of an annotation is then defined as follows:

$$\begin{aligned}
\mathsf{af}(\varepsilon) &:= \top \\
\mathsf{af}(\mathsf{A}(s\,\mathsf{U}_H\,t)^+) &:= \mathsf{A}(s\,\mathsf{U}_H\,t) \\
\mathsf{af}(\Box\,\mathsf{A}(s\,\mathsf{U}_H\,t)^+) &:= \Box\,\mathsf{A}(s\,\mathsf{U}_H\,t) \\
\mathsf{af}(\mathsf{A}(s\,\mathsf{R}_H\,t)^-) &:= \mathsf{E}(\neg s\,\mathsf{U}_H\,\neg t) \\
\mathsf{af}(\Box\,\mathsf{A}(s\,\mathsf{R}_H\,t)^-) &:= \Diamond\,\mathsf{E}(\neg s\,\mathsf{U}_H\,\neg t)
\end{aligned}$$

The admissibility proofs for all rules except $\mathsf{U_H}$ and $\mathsf{R_H}$ are straightforward. We prove admissibility $\mathsf{U_H}$ and $\mathsf{R_H}$ as separate lemmas.

Lemma 6.2.2 (Admissibility of $\mathsf{U_H}$) If $\vdash \neg(C \wedge t)$ and $\vdash \neg(C \wedge s \wedge \Box\,\mathsf{A}(s\,\mathsf{U}_{H,C}\,t))$, then $\vdash \neg(C \wedge \mathsf{A}(s\,\mathsf{U}_H\,t))$.

Proof Assume (a) $\vdash \neg(C \wedge t)$ and (b) $\vdash \neg(C \wedge s \wedge \Box\,\mathsf{A}(s\,\mathsf{U}_{H,C}\,t))$. We first argue that it suffices to show

$$\vdash \mathsf{A}(s\,\mathsf{U}_H\,t) \to \mathsf{A}(s\,\mathsf{U}_{H,C}\,t) \tag{6.3}$$

Assume C and $\mathsf{A}(s\,\mathsf{U}_H\,t)$. We obtain $\neg t$ with (a) and therefore s, H, and $\Box\,\mathsf{A}(s\,\mathsf{U}_H\,t)$ (Lemma 5.6.1(1)). We then obtain $\Box\,\mathsf{A}(s\,\mathsf{U}_{H,C}\,t)$ with (6.3). Together with (b), this yields the required contradiction.

It remains to show (6.3). After applying the rule UI, it remains to show

$$\vdash t \wedge H \to \mathsf{A}(s\,\mathsf{U}_{H,C}\,t) \tag{6.4}$$

$$\vdash s \wedge H \to \Box\,\mathsf{A}(s\,\mathsf{U}_{H,C}\,t) \to \mathsf{A}(s\,\mathsf{U}_{H,C}\,t) \tag{6.5}$$

Claim (6.4) follows with (a) and U1. For (6.5), assume s, H and $\Box\,\mathsf{A}(s\,\mathsf{U}_{H,C}\,t)$. By U2, it suffices to show $\neg C$. This follows with (b). ■

For the admissibility proof for the rule $\mathsf{R_H}$, we employ the rules and axioms for $\mathsf{A}(s\,\mathsf{R}\,t)$ in their dualized form.

Lemma 6.2.3

1. $\vdash \mathsf{E}(s\,\mathsf{U}\,t) \leftrightarrow t \vee (s \wedge \Diamond\,\mathsf{E}(s\,\mathsf{U}\,t))$
2. If $\vdash t \to u$ and $\vdash s \to \Diamond u \to u$, then $\vdash \mathsf{E}(s\,\mathsf{U}\,t) \to u$

The structure of the proof then follows that of Lemma 6.2.2.

Lemma 6.2.4 (Admissibility of $\mathsf{R_H}$) If $\vdash \neg(C \wedge t)$ and $\vdash \neg(C \wedge s \wedge \Diamond\,\mathsf{E}(s\,\mathsf{U}_{H,C}\,t))$, then $\vdash \neg(C \wedge \mathsf{E}(s\,\mathsf{U}_H\,t))$.

Proof Assume (a) $\vdash \neg(C \wedge t)$ and (b) $\vdash \neg(C \wedge s \wedge \Diamond\,\mathsf{E}(s\,\mathsf{U}_{H,C}\,t))$. We first argue that it suffices to show

$$\vdash \mathsf{E}(s\,\mathsf{U}_H\,t) \to \mathsf{E}(s\,\mathsf{U}_{H,C}\,t) \tag{6.6}$$

Assume C and $\mathsf{E}(s\,\mathsf{U}_H\,t)$. We obtain $\neg t$ with (a) and therefore s, H, and $\Diamond\,\mathsf{E}(s\,\mathsf{U}_H\,t)$ (Lemma 6.2.3(1)). With (6.6), we also obtain $\Diamond\,\mathsf{E}(s\,\mathsf{U}_{H,C}\,t)$. Together with (b), this yields the required contradiction.

We now show (6.6). After applying Lemma 6.2.3(2), we need to show:

$$\vdash t \wedge H \to E(s\,\mathsf{U}_{H,C}\,t) \tag{6.7}$$

$$\vdash s \wedge H \to \Diamond\,\mathsf{E}(s\,\mathsf{U}_{H,C}\,t) \to \mathsf{E}(s\,\mathsf{U}_{H,C}\,t) \tag{6.8}$$

Claim (6.7) follows with (a) and Lemma 6.2.3(1) For (6.8), assume s, H, and $\Diamond\,\mathsf{E}(s\,\mathsf{U}_{H,C}\,t)$. With (b) we also obtain $\neg C$. Finally, we obtain $\mathsf{E}(s\,\mathsf{U}_{H,C}\,t)$ with Lemma 6.2.3(1). ■

Theorem 6.2.5 $\vdash \neg(C \wedge \mathsf{af}(a))$ whenever $\Vdash C|a$.

Proof By induction on $\Vdash C|a$. The cases for the rules $\mathsf{U_H}$ and $\mathsf{R_H}$ follow with the lemmas above. All other cases are straightforward. ■

One immediate consequence is that the soundness result from Section 5.2 transfers to the tableau system in the case of annotation free clauses.

Corollary 6.2.6 If $\Vdash C|\varepsilon$, then C is unsatisfiable.

Note that the proofs of Lemma 6.2.2 and Lemma 6.2.4 follow the inductive argument from the proof of Theorem 6.1.3. For a proof at the level of Hilbert derivations, it is essential that the proof does not mention paths.

Also note that neither the proof of Lemma 6.2.2 nor the proof of Lemma 6.2.4 makes essential use of the fact that H is the associated formula of a history or that C is the associated formula of a clause. That is, if one replaces $s \, \mathsf{U}_H \, t$ with $(s \wedge H) \, \mathsf{U} (t \wedge H)$ and $s \, \mathsf{U}_{H,C} \, t$ with $(s \wedge H \wedge \neg C) \, \mathsf{U} (t \wedge H \wedge \neg C)$ one can treat H and C as formulas. The resulting lemmas correspond, up to propositional reasoning, to the rules ERel and ARel of the Hilbert system **LS** (cf. Figure 5.8). Hence, we obtain:

Theorem 6.2.7 **LS** $\vdash s$ iff **IC** $\vdash s$.

Proof The direction from right to left was established as Theorem 5.8.3. The direction from left to right follows with the argument above. ■

6.3 Completeness of Gentzen System for CTL

We now show that the system **GCT** is complete for history-free clauses. The proof works by showing the rules of the refutation calculus from Figure 5.6 admissible for the Gentzen system. This is the place where we profit from the model construction for relaxed demos (Sections 5.3 and 5.4).

We fix some subformula universe U. We call a clause C **Gentzen refutable** if $\Vdash C|\varepsilon$. Further, we call a set of clauses S **Gentzen corefutable** if every clause in $\overline{U} \setminus S$ is Gentzen refutable.

Lemma 6.3.1 (Admissibility of Support Rule) Let $C \subseteq U$.

1. $\Vdash C|a$ whenever $\Vdash D|a$ for all $D \in \mathcal{B}_{\overline{U}}\, C$.
2. $\Vdash C|\varepsilon$ whenever S is Gentzen corefutable and $S \not\rhd C$.

Proof Claim (1) follows by induction on the total size of the non-literal formulas in C using the rules in the first three rows of Figure 6.3. Claim (2) is an immediate consequence of Claim (1). ■

Lemma 6.3.2 (Admissibility of Jump Rule) Let $D \in \mathsf{Dem}\ C$. Then $\Vdash C|a$ whenever $\Vdash D|r\,a$.

Proof Follows immediately with the rules X and X_s. ■

Note that showing admissibility of the jump rule only requires the case where a (and hence $r\,a$) is ε. The added generality is required to prove admissibility of the loop rules.

Lemma 6.3.3 (Admissibility of loop$^+$) Let $S \subseteq \overline{U}$ be Gentzen corefutable and let $\Box\, \mathsf{A}(s\, \mathsf{U}\, t)^+ \in C \in S$. Then $\Vdash C|\varepsilon$ whenever $(S, \overline{U}), C \not\rhd \Box\, \mathsf{A}(s\, \mathsf{U}\, t)^+$.

Proof We define $I := \{\, D \in \overline{U} \mid (S, \overline{U}), D \not\rhd \Box\, \mathsf{A}(s\, \mathsf{U}\, t)^+ \,\}$. It suffices to show $\Vdash C|\varepsilon$ for $C \in I$. Since $C \in I$, there exists some clause $D \in \mathsf{Dem}\ C$ such that $S \not\vartriangleright D, t^+$ and $\mathcal{B}_{\overline{U}}\,(D, s^+) \subseteq I$. By Lemma 6.3.2, it suffices to show $D|\varepsilon$. Since $\mathsf{A}(s\, \mathsf{U}\, t)^+ \in D$ we can apply the rule $\mathsf{U}_\emptyset$. It then remains to prove $\Vdash D|\, \mathsf{A}(s\, \mathsf{U}_\emptyset\, t)^+$. We prove the following generalization

$$\forall H\, D.\ H \subseteq 2^U \wedge D \subseteq U \wedge S \not\vartriangleright D, t^+ \wedge \mathcal{B}_{\overline{U}}\,(D, s^+) \subseteq I \rightarrow\ \Vdash D|\, \mathsf{A}(s\, \mathsf{U}_H\, t)^+$$

by induction on the size of $2^U \setminus H$. Let D and H as in the statement above. We can assume $D \notin H$ since otherwise the claim follows with the rule $\overline{\mathsf{U}}$. After applying the rule $\mathsf{U_H}$, there are two cases:

$\Vdash D, t^+|\varepsilon$. Follows with Lemma 6.3.1(2)

$\Vdash D, s^+|\Box\, \mathsf{A}(s\, \mathsf{U}_{H,D}\, t)^+$. By Lemma 6.3.1(1) it suffices to show $\Vdash E|\Box\, \mathsf{A}(s\, \mathsf{U}_{H,D}\, t)^+$ for $E \in \mathcal{B}_{\overline{U}}\,(D, s^+)$. By assumption we have $E \in I$. Hence, there exists some clause $F \in \mathsf{Dem}\ E$ such that $S \not\vartriangleright F, t^+$ and $\mathcal{B}_{\overline{U}}\,(F, s^+) \subseteq I$. The claim then follows with Lemma 6.3.2 and the induction hypothesis. ■

The admissibility proof of the rule loop^- follows a similar pattern.

Lemma 6.3.4 (Admissibility of loop$^-$) Let $S \subseteq \overline{U}$ be Gentzen corefutable and let $\Box\, \mathsf{A}(s\, \mathsf{R}\, t)^- \in C \in S$. Then $\Vdash C|\varepsilon$ whenever $(S, \overline{U}), C \not\rhd \Box\, \mathsf{A}(s\, \mathsf{R}\, t)^-$.

Proof We define $I := \{\, D \in \overline{U} \mid (S, \overline{U}), D \not\rhd \Box\, \mathsf{A}(s\, \mathsf{R}\, t)^- \,\}$. It suffices to show $\Vdash C|\varepsilon$ for $C \in I$. We can assume that there is no clause $D \in \mathsf{Dem}\, C$ such that $S \not\vartriangleright D$ since otherwise we have $\Vdash C|\varepsilon$ by Lemma 6.3.2. Hence, $S \not\vartriangleright \mathcal{R}\, C, t^-$ and $\mathcal{B}_{\overline{U}}(\mathcal{R}\, C, s^-) \subseteq I$. After applying the rules X and $\mathsf{R}_\emptyset$, we need to show $\Vdash \mathcal{R}\, C|\, \mathsf{A}(s\, \mathsf{R}_\emptyset\, t)^-$. Similar to above, we prove the generalization

$$\forall H\, D.\ H \subseteq 2^U \wedge D \subseteq U \wedge S \not\vartriangleright (D, t^-) \wedge \mathcal{B}_{\overline{U}}(D, s^-) \subseteq I \rightarrow\ \Vdash D|\, \mathsf{A}(s\, \mathsf{R}_H\, t)^-$$

by induction on the size of $2^U \setminus H$. Let H and D as in the statement above. As above, we can assume $D \notin H$. After applying the rule $\mathsf{R_H}$, there are two cases:

$\Vdash D, t^-|\varepsilon$. Follows with Lemma 6.3.1(2).

$\Vdash D, s^-|\Box\, \mathsf{A}(s\, \mathsf{R}_{H,D}\, t)^-$. Let $E \in \mathcal{B}_{\overline{U}}(D, s^-)$. It suffices to show $\Vdash E|\Box\, \mathsf{A}(s\, \mathsf{R}_{H,D}\, t)^-$ (Lemma 6.3.1(1)). Since $E \in I$, we can reason as for C above and obtain $S \not\vartriangleright \mathcal{R}\, E, t^-$ and $\mathcal{B}_{\overline{U}}(\mathcal{R}\, E, s^-) \subseteq I$. The claim then follows with the rule $\mathsf{X_H}$ and the induction hypothesis. ■

Putting everything together, we obtain (ref refers to the rules in Figure 5.6):

Lemma 6.3.5 $\Vdash C|\varepsilon$ whenever $\mathsf{ref}_U\, C$.

The admissibility proofs for the loop rules start by setting a focus on the eventuality under consideration and then proceed by induction on the number of clauses that can still be added to the history. In both cases, the induction establishes that all clauses from $\overline{U}$ where the eventuality under consideration is not inductively fulfilled are refutable. This is slightly more general than needed since we only need to refute clauses from the corefutable set S. The generalization to $\overline{U}$ has the effect that we do not have to rely on the existing refutations for clauses in $\overline{U} \setminus S$. Instead we can continue to build up the history for the eventuality currently in focus. This is important since in the presence of a focused eventuality the refutations for clauses in $\overline{U} \setminus S$ would be of little use. The reason for this is that a refutation of $C|\varepsilon$ does not necessarily provide a refutation of $C|a$ when a is an annotated eventuality (consider the case where the derivation of $C|\varepsilon$ ends with one of the focusing rules). The refutation calculus arising with relaxed demos provides exactly the right invariants for the inductive proofs to go through. Hence, the notion of relaxed fulfillment allows us to handle the fact that the Gentzen system does not allow re-focusing, i.e., dropping an annotated eventuality and focusing on another eventuality.

Theorem 6.3.6 (Informative Completeness) Let C be clause. Then either $\Vdash C|\varepsilon$ or C is satisfied by a finite model.

Proof If C is empty, it is satisfied by any nonempty model. If C is nonempty, so is $\mathsf{sfc}\,C$. With Theorem 5.5.4, we either obtain $\mathsf{ref}_{\mathsf{sfc}\,C}\,C$ or a finite model satisfying C. The claim then follows with Lemma 6.3.5. ■

Together with soundness, we obtain that **GCT** derives exactly the unsatisfiable history-free clauses.

Corollary 6.3.7 $\Vdash C|\varepsilon$ iff C is unsatisfiable.

Since the Gentzen system is only complete for history-free clauses, soundness and completeness only establish decidability of derivability for history-free clauses. The decidability result for arbitrary annotated clauses can be established using fixpoint iteration. For this one shows that every annotated clause is contained in a finite universe of annotated clauses that is closed under backward application of the rules. Decidability then follows by expressing one-step derivability as a monotone function bounded by the clause universe (cf. Remark 3.6.7). The construction is fairly technical and the details are spelled out in the formalization [ACF].

Theorem 6.3.8 Derivability of annotated clauses is decidable.

We remark that while the completeness proof given here does not depend on decidability of derivability, it does rely on the fact that the calculus is analytic. Analyticity ensures that histories cannot grow indefinitely and provides for the inductions in the admissibility proofs for the loop rules.

6.4 Completeness of Gentzen System for K*

We now return to the completeness proof for the Gentzen system for K* (Figure 6.1). For the rest of this section, we take $\Vdash$ to mean derivability in **GK***. As we have seen in the previous section, relaxed fulfillment is crucial for the translation to the Gentzen system to succeed. In order to obtain pruning refutations that can be translated to derivations in **GK***, we adapt the notion of relaxed fulfillment (Section 5.3) to K*.

We fix some subformula universe U for the rest of this section. As before, we call a set of clauses $S \subseteq \overline{U}$ **Gentzen corefutable** if $\Vdash C|\varepsilon$ for all clauses $C \in \overline{U} \setminus S$.

Let C be a clause. Similar to CTL, we define the **demands** of C to be the following set of clauses.

$$\mathsf{Dem}\, C := \{\, \mathcal{R}\, C, s^- \mid \Box s^- \in C \,\}$$

Since models for K* may include terminal states, $\mathcal{R}\, C$ is not a demand of C. We inductively define the **relaxed fulfillment relation** $S, C \rhd \Box\Box^* s^-$ between sets of clauses S, clauses C and eventuality literals $\Box\Box^* s^-$.

$$\frac{\forall E \in \mathsf{Dem}\, C.\, S \triangleright E \qquad D \in S \qquad D \triangleright \mathcal{R}\, C, s^-}{S, C \rhd \Box\Box^* s^-} \qquad \frac{\forall E \in \mathsf{Dem}\, C.\, S \triangleright E \qquad D \in \overline{U} \qquad D \triangleright \mathcal{R}\, C \qquad S, D \rhd \Box\Box^* s^-}{S, C \rhd \Box\Box^* s^-}$$

We will show that replacing $S, C \triangleright \Box\Box^* s^-$ with $S, C \rhd \Box\Box^* s^-$ for the purpose of pruning still yields the canonical demo for U. Moreover, we complement this relaxed version of pruning with a refutation calculus whose rules are admissible for **GK***. We define:

$$\begin{aligned} p\, C\, S &:= (\forall D \in \mathsf{Dem}\, C.\, S \triangleright C) \wedge (\forall \Box\Box^* s^- \in C.\, S, C \rhd \Box\Box^* s^-) \\ \mathcal{D}_r &:= \mathsf{prune}\, p\, \overline{U} \end{aligned}$$

The clauses over U that are not supported by $\mathcal{D}_r$ can be characterized using a refutation calculus. The rules of the refutation calculus are given in Figure 6.4. The only difference between the refutation calculus employed here and the refutation calculus in Figure 4.2 is the use of $S, C \not\rhd \Box\Box^* s^-$ in the wloop-rule. Using (the negation of) relaxed fulfillment for the loop rule yields an a priori *weaker* rule. This is exactly what allows us to show the wloop-rule admissible for the Gentzen system.

Lemma 6.4.1

1. $\mathsf{wcoref}\, \mathcal{D}_r$.
2. $\mathsf{wref}\, C$ whenever $C \subseteq U$ and $\mathcal{D}_r \not\triangleright C$.

$$\frac{C \subseteq U \qquad \mathcal{S} \not\triangleright C \qquad \mathsf{wcoref}\,\mathcal{S}}{\mathsf{wref}\,C}\ \mathsf{supp} \qquad\qquad \frac{\Box s^- \in C \qquad \mathsf{wref}(\mathcal{R}\,C, s^-)}{\mathsf{wref}\,C}\ \mathsf{jump}$$

$$\frac{\Box\Box^* s^- \in C \in \mathcal{S} \subseteq \overline{U} \qquad \mathcal{S}, C \not\triangleleft \Box\Box^* s^- \qquad \mathsf{wcoref}\,\mathcal{S}}{\mathsf{wref}\,C}\ \mathsf{wloop}$$

$$\mathsf{wcoref}\,\mathcal{S} := \forall C \in \overline{U} \setminus \mathcal{S}.\mathsf{wref}\,C$$

Figure 6.4: Refutation calculus for relaxed pruning

Lemma 6.4.2 Let C be a clause. Then $\Vdash C|\varepsilon$ whenever $C \subseteq U$ and $\mathsf{wref}\,C$.

Proof The reasoning for the support rule and the jump rule is essentially the same as for **GCT**. In particular, we have an analog to Lemma 6.3.1:

$$\forall C \subseteq U.(\forall D. \in \mathcal{B}_{\overline{U}}\,C \to \Vdash D|a) \to \Vdash C|a \tag{6.9}$$

$$\forall \mathcal{S}\,\forall C \subseteq U.\ \mathsf{wcoref}\,\mathcal{S} \to \mathcal{S} \not\triangleright C \to \Vdash C|\varepsilon \tag{6.10}$$

For the wloop-rule, let $\mathcal{S} \subseteq \overline{U}$ and let $\Box\Box^* s^- \in C \in \mathcal{S}$ such that $\mathsf{wcoref}\,\mathcal{S}$ and $\mathcal{S}, C \not\triangleleft \Box\Box^* s^-$. We show $\Vdash C|\varepsilon$. Since $\Box\Box^* s^- \in C$, we can apply the foc-rule. It then suffices to show

$$\forall H \subseteq 2^U\,\forall C \in \overline{U}.\ \mathcal{S}, C \not\triangleleft \Box\Box^* s^- \to \Vdash C|\Box\Box^*_H s^-$$

by induction on the size of $2^U \setminus H$. Let $H \subseteq 2^U$ and $C \in \overline{U}$ and assume $\mathcal{S}, C \not\triangleleft \Box\Box^* s^-$. We consider two cases:

Case 1. $\mathcal{S} \not\triangleright D$ for some $D \in \mathsf{Req}\,C$. After applying the X-rule, it suffices to show $\Vdash D|\varepsilon$. This follows with (6.10).

Case 2. Otherwise, we have

$$\mathcal{S} \not\triangleright \mathcal{R}\,C, s^- \tag{6.11}$$

$$\forall D \in \overline{U}.D \triangleright \mathcal{R}\,C \to \mathcal{S}, D \not\triangleleft \Box\Box^* s^- \tag{6.12}$$

After applying the $\mathsf{X_H}$-rule it suffices to show $\Vdash \mathcal{R}\,C|\Box^*_H s^-$. Without loss of generality $\mathcal{R}\,C \notin H$. Applying the $\mathsf{G}_{\overline{\mathsf{H}}}$-rule leaves us with two claims. The first claim, $\Vdash \mathcal{R}\,C, s^-|\varepsilon$, follows with (6.10) and (6.11). It remains to prove $\Vdash \mathcal{R}\,C|\Box\Box^*_{H,\mathcal{R}\,C} s^-$. By (6.9) it suffices to show $\Vdash D|\Box\Box^*_{H,\mathcal{R}\,C} s^-$ for $D \in \mathcal{B}_{\overline{U}}(\mathcal{R}\,C)$. This follows with (6.12) and the induction hypothesis. ∎

Remark 6.4.3 In the proofs of Lemma 6.3.3 and Lemma 6.3.4 the non-fulfillment assumption needs to be inverted twice, once during the induction and once before starting the inductive part. This is required due to a slight mismatch between the refutation calculus, which works with eventuality literals, and the Gentzen system where a focus can only be set on a non-literal eventuality. For K*, we avoided this duplication by using a focusing rule for eventuality literals (cf. Section 6.1). For CTL, we choose to stay closer to the system CT [BL08].

We have established that **GK*** can refute all clauses over U that are not supported by $\mathcal{D}_r$ (Lemmas 6.4.1(2) and 6.4.2). In order to prove completeness for **GK***, it remains to show that all clauses supported by $\mathcal{D}_r$ are satisfiable. For this, we show that $\mathcal{D}_r$ can be extended to a demo over U. Note that $\mathcal{D}_r$ is corefutable (Lemma 6.4.1(1)) and, by soundness of **GK***, must contain all satisfiable clauses from $\overline{U}$. Hence, the extension cannot add satisfiable clauses to $\mathcal{D}_r$. It is merely a technical device to show that $\mathcal{D}_r$ contains only satisfiable clauses. We define

$$\mathcal{E} := \{\, C \in \overline{U} \mid \forall D \in \mathsf{Dem}\, C . \mathcal{D}_r \triangleright D \,\}$$

We show that $\mathcal{D}_r \cup \mathcal{E}$ is a demo according to Definition 4.3.1.

Lemma 6.4.4 Let $\Box s^- \in C \in \mathcal{D}_r \cup \mathcal{E}$. Then $\mathcal{D}_r \triangleright \mathcal{R}\, C, s^-$.

Proof We clearly have $\mathcal{R}\, C, s^- \in \mathsf{Dem}\ C$. If $C \in \mathcal{D}_r$, the claim follows with the correctness of pruning (Lemma 3.4.3). Otherwise the claim is trivial. ■

Note that the lemma above is stronger than the demo property for $\Box s^-$ which only requires $\mathcal{D}_r \cup \mathcal{E} \triangleright \mathcal{R}\, C, s^-$. The stronger lemma is needed to establish the demo condition for eventuality literals.

Lemma 6.4.5 Let $\Box\Box^* s^- \in C \in \mathcal{D}_r$. Then $\mathcal{D}_r \cup \mathcal{E}, C \triangleright \Box\Box^* s^-$.

Proof We have $\mathcal{D}_r, C \rhd \Box\Box^* s^-$ since $C \in \mathcal{D}_r$ (Lemma 3.4.3). We proceed to prove $\mathcal{D}_r \cup \mathcal{E}, C \triangleright \Box\Box^* s^-$ by induction on $\mathcal{D}_r, C \rhd \Box\Box^* s^-$.

Case 1. We have $D \triangleright \mathcal{R}\, C, s^-$ for some $D \in \mathcal{D}_r$. Hence, $\mathcal{D}_r \cup \mathcal{E}, C \triangleright \Box\Box^* s^-$.

Case 2. By induction hypothesis, there exists some clause $D \in \overline{U}$ such that $D \triangleright \mathcal{R}\, C$ and both $\mathcal{D}_r, D \rhd \Box\Box^* s^-$ and $\mathcal{D}_r \cup \mathcal{E}, D \triangleright \Box\Box^* s^-$. Inversion on $\mathcal{D}_r, D \rhd \Box\Box^* s^-$ yields $D \in \mathcal{E}$ and therefore $\mathcal{D}_r \cup \mathcal{E}, C \triangleright \Box\Box^* s^-$ as required. ■

Lemma 6.4.6 Let $\Box\Box^* s^- \in C \in \mathcal{E}$. Then $\mathcal{D}_r \cup \mathcal{E}, C \triangleright \Box\Box^* s^-$.

Proof By Lemma 6.4.4, there exists some clause $D \in \mathcal{D}_r$ such that $D \triangleright \mathcal{R}\, C, \Box^* s^-$. Hence, either $D \triangleright s^-$ and the claim follows immediately with definition of fulfillment or $\Box\Box^* s^- \in D$ and the claim follows with Lemma 6.4.5. ■

Lemma 6.4.7 $\mathcal{D}_r \cup \mathcal{E}$ is a demo.

Proof Immediate with the lemmas above. ■

Together with the demo properties from Chapter 4, we obtain completeness of **GK*** for history-free clauses.

Theorem 6.4.8 Let C be a clause. Then either $\Vdash C|\varepsilon$ or C finitely satisfiable.

Proof Let $U := \mathsf{sfc}\, C$. If $\mathcal{D}_r \rhd C$, then C is finitely satisfiable (Lemma 6.4.7 and Lemma 4.3.2). Otherwise, we have $\Vdash C|\varepsilon$ by Lemma 6.4.1(2) and Lemma 6.4.2. ■

Corollary 6.4.9 $\Vdash C|\varepsilon$ iff C is unsatisfiable.

Proof Follows with Theorem 6.1.3 and Corollary 4.5.10. ■

Fact 6.4.10 $\mathcal{D}_r = \mathcal{D}(U)$.

Proof Follows since both $\mathcal{D}_r$ and $\mathcal{D}(U)$ (Section 4.4) contain exactly the satisfiable clauses from $\overline{U}$. ■

The demo construction above is inspired by the model construction for the relaxed demos for CTL (cf. Chapter 5). The key insight is that one does not need to know a priori that the inductive fulfillment holds for every eventuality at every state. It suffices if from every state with an eventuality one can reach a state where this eventuality is fulfilled. This is possible since eventualities “propagate” to successor states until they are fulfilled. For CTL, this is used to fulfill universal eventualities (i.e., “always until” formulas) one after the other. In the proof of Lemma 6.4.6 we need a single extra transition to get from $\mathcal{E}$ back to $\mathcal{D}_r$.

6.5 Remarks

Our completeness proof for the Gentzen system for CTL differs considerably from the corresponding completeness proof given by Brünnler and Lange [BL08]. Their proof works by constructing models from unsuccessful derivations of a more restrictive sequent calculus called CT’. The calculus CT’ is designed such that backward derivations are finite. This requires a rule whose applicability is only defined for backward proof search. Further, derivations in CT’ carry along a rotating list of eventualities to ensure that during unsuccessful backward derivations every eventuality is focused at some point. This ensures that, after being collapsed to Hintikka sets, maximal unsuccessful derivations exhibit a structure similar to the model constructed in Section 5.4. More precisely, the model constructed in [BL08] roughly corresponds to a “top-down” construction of a model where one obtains neither sharing within fragments nor sharing of fragments. While the model constructed this way is finite, it may be exponentially larger than the fragment-based construction underlying our proofs. Our development for CTL is designed such that we obtain the small-model property and the completeness of the Hilbert system and the Gentzen system with a single model construction. With the results from Chapter 5 in place, soundness and completeness of the Gentzen system can be formalized in about 150 lines each. Compared to the effort for the model construction, this is almost for free.

In preliminary work [DS14], we employed a direct construction of fragments for annotated clauses that are not Gentzen derivable. The construction relies on the closure properties of underivable clauses and, similar to the proof in [BL08], uses the fact that histories cannot grow indefinitely to ensure termination. The "top-down" construction of fragments introduces sharing of fragments but no sharing within fragments. Consequently, the construction also does not provide a reasonable small-model property as corollary.

We prove soundness for **GCT** by showing the individual rules admissible for the Hilbert system **IC**. This is possible since CTL can express the until operator underlying the semantics of histories. Together with completeness of the Gentzen system, this yields an alternative completeness proof for the Hilbert system. That is, for CTL the Gentzen system can serve as decision method underlying a constructive completeness proof for the Hilbert system. This approach was taken in [DS14].

For K*, the situation is different. The Hilbert system for K* cannot express the until operator [EH85, Theorem 8.3] and therefore cannot express the semantics of histories. Consequently, the Gentzen system for K* cannot be used as underlying decision method to prove completeness of the Hilbert system. Pruning, on the other hand, is flexible enough to underpin constructive completeness proofs for Gentzen and Hilbert systems for K, K*, and CTL.

Kashima [Kas10] gives an alternative completeness proof for a Hilbert system for K*. Similar to the arguments in [BL08] and [GHL$^+$07], the proof exploits that every satisfiable eventuality can be fulfilled using a cycle-free path of bounded length. Since K* cannot enforce cycle-freeness using histories, this requires an explicit enumeration of all possible paths. The proof is arguably not as simple as the one given in Chapter 4.

7 Overview of the Formalization in Coq

All results in this thesis have been formalized in the proof assistant Coq [Coq15] using the Ssreflect [GMT08] extension. Altogether, the formalization accompanying this thesis [ACF] consists of about 6800 lines split about half and half between specifications (e.g., definitions and lemma statements) and proofs. The relatively small size of the formal development is achieved through careful engineering. In particular, we build reusable components whenever possible.

The mathematical development in the previous chapters makes extensive use of finite sets (e.g., clauses, demos, and histories). Further, the completeness proofs for the Hilbert systems require the construction of a fair amount of Hilbert derivations. Both issues are handled informally in the mathematical presentation, but require a certain amount of engineering before one can do proofs in Coq at a level that is close to the mathematical presentation.

In this chapter we describe the finite set library and the infrastructure for the generation of Hilbert proofs underlying our development. We conclude the chapter by outlining the structure of the formal development and by describing how the formalization effort distributes over the different parts of this thesis.

7.1 Finite Set Library

For our development we make extensive use of typed finite sets over countable types. Recall that for our purposes finite sets are data types. In particular, we are only interested in the case where the elements are "data" (e.g., formulas or finite sets of formulas). We use finite sets to implement decision procedures such as pruning (Section 3.4.2) or to show decidability of inductive definitions using fixpoint iteration (cf. Lemma 4.4.2). These decision procedures serve as the basis for our constructive proofs. Since we are not interested in executing these procedures on concrete instances, we implement finite sets and their operations without regard for computational costs.

7.1.1 Quotient Construction

Let T be a countable type. We realize the type $\mathsf{set}\,T$ of finite sets over T as a constructive quotient on the type of lists over T. That is, every finite set over is T represented using some canonical duplicate free list with the same elements. The construction makes use of the choice operator for countable types.

Let $p, q : T \to \mathbb{B}$ be decidable predicates. Using the function $\mathsf{xchoose}_T$ (Section 2.4) one can define a function $\mathsf{choose}_T : (T \to \mathbb{B}) \to T \to T$ satisfying the following properties:

$$p\,x \to p(\mathsf{choose}_T\,p\,x) \tag{7.1}$$

$$(\forall x.p\,x = q\,x) \to p\,x \to p\,y \to \mathsf{choose}_T\,p\,x = \mathsf{choose}_T\,q\,y \tag{7.2}$$

That is, if x satisfies p, then the result of $\mathsf{choose}_T\,p\,x$ also satisfies p and depends neither on x nor on any intensional properties of p. Hence, if $e : T \to T \to \mathbb{B}$ is a decidable equivalence relation and $x : T$, then $\mathsf{choose}_T(e x)\,x$ yields some canonical representative for the equivalence class containing x. Now let $\mathsf{uniq} : \mathsf{list}\,T \to \mathbb{B}$ be a predicate that checks that a list does not contain duplicates and let $\mathsf{perm} : \mathsf{list}\,T \to \mathsf{list}\,T \to \mathbb{B}$ be a binary predicate that checks whether one list is a permutation of the other.[1] The type $\mathsf{set}\,T$ is then defined as:

$$\mathsf{set}\,T := \Sigma l : \mathsf{list}\,T.\ \mathsf{uniq}\,l \wedge l = \mathsf{choose}_{\mathsf{list}\,T}(\mathsf{perm}\,l)\,l$$

Here, we exploit that since T is countable, the type $\mathsf{list}\,T$ is countable as well. Set membership is defined as membership in the list representing the set. The type $\mathsf{set}\,T$ is extensional in the sense that for $X, Y : \mathsf{set}\,T$ we have:

$$(\forall x.x \in X \leftrightarrow x \in Y) \leftrightarrow X = Y$$

Extensionality ensures that set membership on all levels (sets, sets of sets, etc.) is just membership in the list representing the set. Moreover, it simplifies proofs since one never has to show that a function respects set equivalence.

The proof of extensionality relies on two facts. First, for every finite collection of elements, there is exactly one duplicate-free list l with the same elements satisfying $l = \mathsf{choose}_{\mathsf{list}\,T}(\mathsf{perm}\,l)\,l$. Moreover, $\mathsf{uniq}\,l \wedge l = \mathsf{choose}_{\mathsf{list}\,T}(\mathsf{perm}\,l)\,l$ is a decidable property and can be represented such that it is proof irrelevant, i.e., has at most one proof. Hence, there is exactly one object of type $\mathsf{set}\,T$ representing a given collection of elements.

We define the cardinality of a set to be the length of the list representing the set. This is correct since we only allow duplicate-free representatives.

Remark 7.1.1 While Ssreflect provides a generic quotient construction for decidable equivalence relations [Coh13], we construct the quotient manually. This allows us to restrict to duplicate-free representatives.

It is straightforward to define functions $\mathsf{set\_of} : \mathsf{list}\,T \to \mathsf{set}\,T$ and $\mathsf{elements} : \mathsf{set}\,T \to \mathsf{list}\,T$ that preserve elements. Hence, subset and operations like union, separation $\{\, x \in X \mid p\,x \,\}$, replacement $\{\, f x \mid x \in X \,\}$ and powerset 2^A can be obtained by lifting the corresponding predicates and operations on lists. Moreover, we instantiate Ssreflect's big-operator library [BGBP08] which provides indexed unions $\bigcup_{x \in X} f x$. Altogether, the finite set library includes about 170 lemmas.

[1] The functions choose_T, uniq and perm are all part of the Ssreflect libraries.

7.1.2 Related Libraries

There are a number of libraries for Coq that provide finite sets in one form or the other. The standard library for Coq features two implementations of finite sets (FSet and the more recent MSet). These libraries are designed for the extraction of executable programs and therefore do not use an extensional representation. Moreover, Fset and MSet are implemented as functors and need to be instantiated separately for every type which is cumbersome. There is also an MSet-like library implemented using type classes [Les11], which avoids the need for manual instantiation but also does not employ an extensional representation.

The Mathematical Components Library [GMR$^+$07, GGMR09] distributed with Ssreflect [GMT08] provides extensional finite sets, but only over finite base types. Regarding the case of finite sets over countable base types, there is a small "proof of concept" library developed by Strub [Str14]. Moreover, there is a draft library by Cohen [Coh15] incorporating many of the ideas of the library developed for this thesis.

Remark 7.1.2 Many of the results presented in this thesis could also be obtained using Ssreflect's finite sets over finite types. When working with finite base types, pruning, demos, and Gentzen systems need to be defined relative to some "input" formula. This approach was taken in some of our preliminary work [DS11, DS12]. While relative definitions are fairly natural for pruning and demos, indexed collections of Gentzen systems appear ad-hoc. The main problem when working with a finite type of subformulas, however, is the fact that elements of this type are dependent pairs of formulas and proofs of membership in the subformula universe. These proof terms also appear in lemma statements. Consequently, side conditions regarding membership in the subformula universe may need to be established while stating lemmas. While this is doable, it is cumbersome and leads to cluttered lemma statements.

7.2 Modular Library for Hilbert Derivations

In order to formalize the completeness results for the Hilbert systems presented in this thesis, we need to construct a fair amount of derivations in a number of different Hilbert systems. Given that Hilbert systems are fairly low-level, this requires some engineering. The problem of constructing Hilbert derivations is in many respects similar to the construction of proof terms. We use Coq's tactic language to provide goal management, rewriting, and assumption management for the construction of Hilbert derivations. Moreover, we develop a modular library of Hilbert systems for propositional logic, basic modal logic and temporal logics that allows seamless reuse of facts established for subsystems.

7.2.1 Rewriting

We use setoid rewriting [Soz09] to rewrite with provable equivalences (e.g., De Morgan laws) inside formulas. While rewriting with equivalences is convenient, it is often too restrictive for our purposes because several important lemmas (e.g., Lemma 3.5.4(3)) are implications. Therefore, we also use setoid rewriting to rewrite with provable implications using compatibility rules such as

$$\frac{\vdash s' \to s \qquad \vdash t \to t'}{\vdash (s \to t) \to (s' \to t')}\;\mathsf{C}_{\to} \qquad\qquad \frac{\vdash s \to s' \qquad \vdash t \to t'}{\vdash s \wedge t \to s' \wedge t'}\;\mathsf{C}_{\wedge} \qquad\qquad \frac{\vdash s \to s'}{\vdash \Box s \to \Box s'}\;\mathsf{C}_{\Box}$$

Setoid rewriting helps significantly in obtaining shorter and more linear proof scripts. For example, assume we know $\vdash t \to t'$. This allows reducing the claim $\vdash s \wedge t \to u$ to $\vdash s \wedge t' \to u$ with a single setoid rewrite. Performing the same reduction manually requires a non-linear derivation:

$$\frac{\dfrac{\dfrac{}{\vdash s \to s}\;\mathsf{Id} \qquad \vdash t \to t'}{\vdash s \wedge t \to s \wedge t'}\;\mathsf{C}_{\wedge} \qquad \vdash s \wedge t' \to u}{\vdash s \wedge t \to u}\;\mathsf{Trans}$$

The linearization achieved with setoid rewriting leads to proof scripts that are easier to follow when interactively stepping through the proofs. Moreover, multiple rewrites can be chained together allowing for larger and more meaningful proof steps.

7.2.2 Assumption Management

In addition to rewriting, we use big conjunctions [BGBP08] to provide for an ND-style assumption management. Let A be a list of formulas. We abbreviate $\bigwedge_{s \in A} s$ as $\bigwedge A$ and define an **entailment relation** as follows:

$$A \vdash s \;:=\; \vdash \bigwedge A \to s$$

For the entailment relation, we use the derived rules shown in Figure 7.1. The rules in the upper row are realized with lemmas, whereas the rules in the lower row are realized as tactics using the Ltac [Coq15] tactic language. Besides the fact that the rules App and $\mathsf{App}_{\mathsf{H}}$ are difficult to state as lemmas, applying the rules would be cumbersome. With Ltac we can use Coq's unification mechanism to find instances where n is minimal. The rule $\mathsf{App}_{\mathsf{H}}$ is used to apply previously established facts, and we use unification to instantiate universally quantified variables in the leftmost premise. Using these derived rules we define a collection of tactics corresponding to the Coq tactics `intro`, `apply`, and `assert`.

The infrastructure for assumption management is used extensively to establish basic (e.g., propositional) facts. For the more high-level proofs, setoid rewriting is often the main source of automation.

$$\frac{s \in A}{A \vdash s}\ \textsf{Asm} \qquad \frac{A, s \vdash t}{A \vdash s \to t}\ \textsf{Intro} \qquad \frac{A \vdash t \qquad A, t \vdash s}{A \vdash s}\ \textsf{Cut} \qquad \frac{s, A \vdash t}{A, s \vdash t}\ \textsf{Rot}$$

$$\frac{A \vdash s_1 \quad \dots \quad A \vdash s_n}{A, s_1 \to \dots \to s_n \to t \vdash t}\ \textsf{App} \qquad \frac{\vdash s_1 \to \dots \to s_n \to t \qquad A \vdash s_1 \quad \dots \quad A \vdash s_n}{A \vdash t}\ \textsf{App}_{\textsf{H}}$$

Figure 7.1: Assumption management

We remark that in order to establish the lemmas underlying the rules in Figure 7.1, we need to prove about a handful of simple propositional facts without infrastructure support. These facts have simple proofs in the form of λ-terms. We use a simple Haskell script to translate these lambda terms first to SK-combinator terms and then to proof scripts.

7.2.3 Modular Hierarchy

The rules for assumption management shown in Figure 7.1 are not specific to a particular Hilbert system. The construction works for all Hilbert systems extending classical propositional logic.

Let F be some type of formulas. We call a predicate $\vdash : F \to \textsf{Prop}$ a **P-system** if it can prove all theorems of classical propositional logic, i.e., if it satisfies the following conditions:

P1. There exist formulas $\bot : F$ and $\to : F \to F \to F$.

P2. If $\vdash s \to t$ and $\vdash s$, then $\vdash t$.

P3. $\vdash s \to t \to s$

P4. $\vdash ((u \to s \to t) \to (u \to s) \to u \to t)$

P5. $\vdash ((s \to \bot) \to \bot) \to s$

The rules in Figure 7.1 are available for every P-system.

In addition to P-systems, we also define **K-systems** to be P-systems that feature a $\Box$-operator and can prove N and Nec (Figure 3.1). All Hilbert systems considered in this thesis are K-systems. Hence, we can reuse the facts established for the Hilbert system for K also for our developments for modal logic with transitive closure (Chapter 4) and CTL (Chapter 5).

Technically, the modular hierarchy is realized using canonical structures. P-systems are realized as records comprised of a type of formulas, a predicate on formulas (i.e., the Hilbert system) and the conditions mentioned above:[2]

$$\textsf{pSystem} := \{\textsf{form} :> \textsf{countType};\ \vdash : \textsf{form} \to \textsf{Prop};\ \dots\}$$

[2] The actual implementation also distinguishes between propositional and minimal logic.

The definition abstracts both form the concrete Hilbert system and from the type of formulas. This is important since we are considering different types of formulas, each with multiple Hilbert systems. The notation :> turns form into a coercion from pSystem to the underlying countable type of formulas. This allows stating generic lemmas like

$$\forall (s\, t : \mathsf{pSystem}).\ \vdash (\neg t \to \neg s) \to s \to t$$

These lemmas then apply to every type of formulas and every Hilbert system for which a corresponding pSystem record has been constructed.

The record for K-systems is defined as follows.

$$\mathsf{kSystem} := \{\mathsf{pform} :> \mathsf{pSystem};\ \Box : \mathsf{pform} \to \mathsf{pform};\ \mathsf{Nec}\ s : \vdash s \to \vdash \Box s;\ \ldots\}$$

The coercion to pSystem is required to state the type of Nec which involves the provability relation (i.e., ⊢) from the underlying pSystem record. Moreover, the coercion ensures that propositional reasoning is available when reasoning abstractly about K-systems.

We also define abstract K*-systems and CTL-systems based on the Hilbert systems **K*** (Figure 4.1) and **IC** (Figure 5.1). Both extend extend K-systems in the same manner as K-Systems extend P-systems. The style of defining substructures by including a record for the superstructure and coercing to it is sometimes referred to as telescoping [GGMR09]. Telescoping is simple to implement but may cause performance problems for deeply nested structures. In our case, the hierarchy is only four levels deep, so telescoping is sufficient.

We establish most of the basic facts needed for our completeness and translation results for abstract systems. Altogether the library for Hilbert systems contains more than 100 lemmas. We remark that, since we do not consider extensions of CTL and K*, abstracting over their axiomatizations does not provide any immediate benefits. However, both logics have natural extensions in the literature: the axiomatization of K* can be extended to an axiomatization of UB [BAPM83] and the axiomatization CTL can be extended to an axiomatization of ECTL [Kas14].

7.3 Structure of the Formalization

Altogether the formal development [ACF] consists of about 6800 lines of code. This includes the three developments on K (800 lines), K* (1200 lines), and CTL (2900 lines) as well as shared libraries totaling about 1800 lines (e.g., finite sets and infrastructure for Hilbert proofs). The formal proof of Theorem 5.5.4 (i.e., the shared part of the completeness proofs for the Hilbert system and the Gentzen system for CTL) needs about 800 lines with the construction of models from relaxed demos requiring most of the effort (670 lines). This is significantly more than the formalization effort required for the translation to Hilbert refutations

(230 lines, not counting the library for Hilbert proofs) and Gentzen derivations (150 lines).

The structure of the formal development differs significantly from the structure of the mathematical development presented in the preceding chapters. While the mathematical development necessarily follows a linear structure, the formalization consists of a collection of modules with a non-linear dependency structure. The correspondence between the individual sections of this thesis and the modules of the formalization is given in Figure 7.2. The table also lists the number of lines of specification (e.g., definitions and lemma statements) and proof script[3] for each module in order to give some indication of the formalization effort for the various parts of the development.

The dependencies between the individual modules of the development are given in Figure 7.3. For each logic, we include a module named “results” where we restate the main results for that logic. The fine-grained structure, in particular for the development on CTL serves two purposes. It helps visualizing the dependency structure between the individual parts of the development. Moreover, it allows for faster proof checking on multi-core machines since the current version of Coq (i.e, version 8.4) uses at most one CPU core to check a given module. The checking time is mainly relevant for the construction of Hilbert proofs. The combination of setoid rewriting and the modular library for Hilbert facts leads to proofs that take significantly longer to check than the remaining parts of the development.

[3] Generated with `coqwc` and corrected for the known issue of counting proofs starting with “Proof with” as specification. The total includes some small unlisted files.

Folder	File(s)	Content	Section(s)	Spec.	Proof
libs	edone, bcase, base	basic infrastructure	N/A	133	143
	fset	finite set library	7.1	416	384
	modular_hilbert	library for Hilbert proofs	7.2	340	232
K	K_def	formulas, models, Hilbert system, soundness	3.1, 3.2	203	113
	demo	demos, model existence, pruning	3.3, 3.4	53	44
	hilbert_ref	completeness of Hilbert system	3.5	73	68
	gentzen	soundness and completeness of Gentzen system	3.6	52	55
	universal_model	construction of universal model	3.7	60	72
Kstar	Kstar_def	formulas, models, Hilbert system, soundness	4.1, 4.2, 4.6	263	179
	demo	demos, model existence, pruning	4.3, 4.4	74	89
	hilbert_ref	completeness of Hilbert system	4.5	64	90
	gen_def	Gentzen system and soundness	6.1	87	59
	gen_ref	relaxed pruning, completeness of Gentzen system	6.4	92	170
CTL	CLT_def	formulas, models	5.1	307	206
	hilbert	hilbert system **IC**, soundness	5.2	34	20
	dags, demo	fragment-based model construction	5.3, 5.4	266	415
	relaxed_pruning	pruning and refutation calculus for relaxed demos	5.5	54	72
	hilbert_ref	pruning refutations to Hilbert refutations	5.6	74	154
	agreement	Agreement between inductive and path semantics	5.9	90	227
	hilbert_Eme90, hilbert_LS	hilbert systems from [Eme90] and [LS01]	5.8	144	147
	gen_def	definition of Gentzen system for CTL	6.2	79	2
	hilbert_hist,gen_hsound	Hilbert soundness for Gentzen system	6.2	41	107
	gen_ref	pruning refutations to Gentzen derivations	6.3	43	119
	gen_dec	decidability of Gentzen derivability	6.3	109	174
			Total:	3349	3423

Figure 7.2: Modules of the formalization

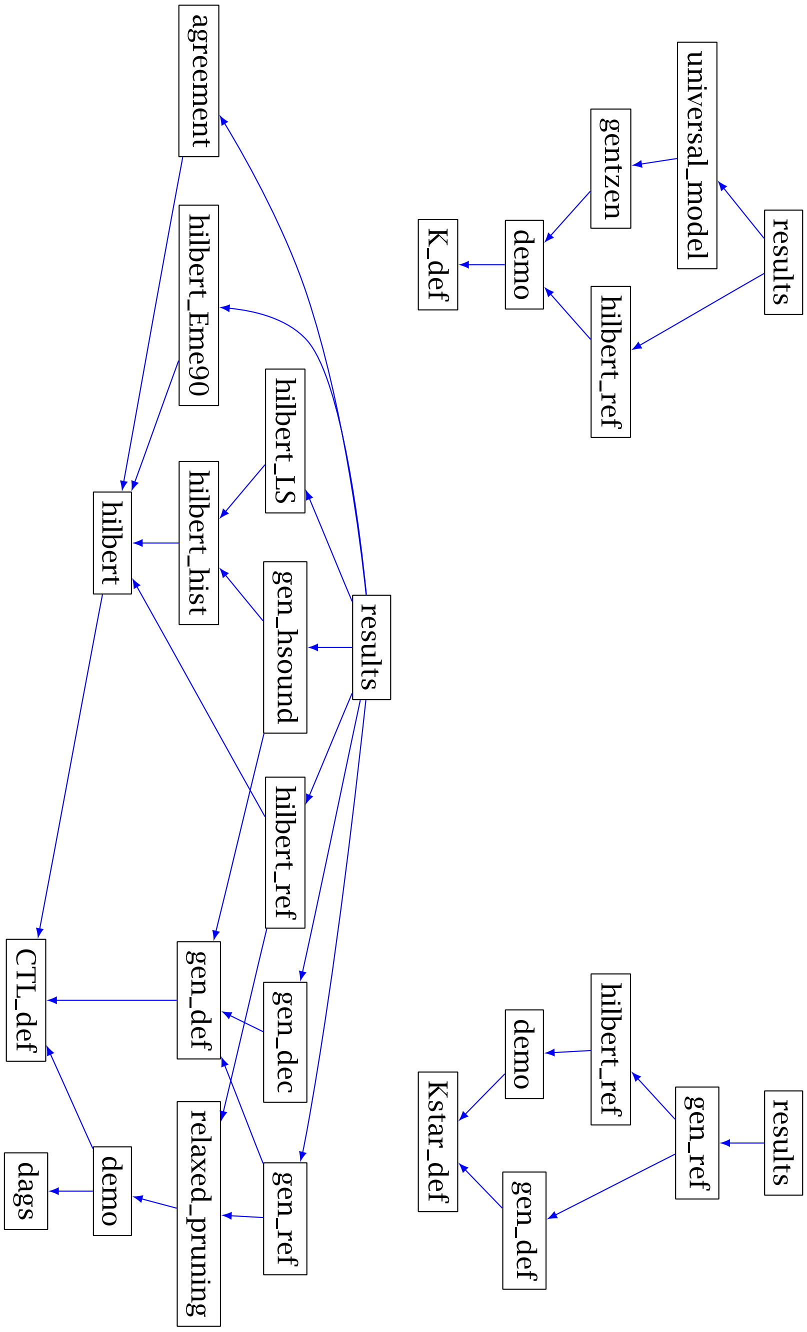


Figure 7.3: Structure of the developments on K, K*, and CTL

8 Conclusion

In this thesis we have given formal and constructive proofs of metatheoretic properties of CTL and subsystems. In this chapter we review the main contributions of this thesis and mention some possibilities for future research.

8.1 Summary of Results

We have seen that a number of interesting metatheoretic results for classical modal and temporal logics can be established completely constructively. We have given formal and constructive soundness and completeness proofs for Hilbert and Gentzen systems with respect to a class of models we call *classical models*. For CTL, the class of classical models was based on an inductive interpretation of the path modalities allowing us to avoid reasoning about infinite paths.

Our completeness results were established as *informative decision methods* constructing for a given formula either a finite model or a refutation in some inference system. The central notions underlying our completeness results were *demos* and *pruning*. Demos were designed such that all clauses of a demo can be jointly satisfied by a finite model. We used pruning to partition the clauses over a given subformula universe into clauses supported by the canonical demo and pruning refutable clauses. We then obtained completeness of Hilbert and Gentzen systems by translating pruning refutations to derivations in the respective system.

For our demos, we employed *inductive fulfillment relations* to handle eventualities. This allowed for a uniform treatment of the eventualities of K^* and CTL. In the case of CTL, the fulfillment relations replace the test for embedded fragments employed by Emerson and Halpern [EH85, Eme90]. For us, inductive fulfillment provided a better compromise between the complexity of the model construction and the construction of Hilbert refutations. We have shown that non-fulfillment has the inversion properties required for the translation to Hilbert refutations. For the construction of fragments from inductive fulfillment we have given a simple bottom-up construction.

For the translation of pruning refutations to Gentzen derivations the precise formulation of the pruning rules turned out to be crucial. In order to obtain pruning refutations that can be translated to derivations in the history-based Gentzen system for CTL [BL08], we introduced the notion of *relaxed demo*. The pruning rules arising with relaxed demos allowed us to handle the nondeterminism arising with the focusing rules of the Gentzen system. Consequently, we obtained the small-model theorem for CTL as well as completeness of Hilbert and Gentzen system with a single model construction.

Relaxed fulfillment for CTL can be seen as arising naturally from a close inspection of the fragment-based model construction given by Emerson [Eme90]. The development of relaxed fulfillment happened in two stages. In [DS14] we proved completeness of the Hilbert system for CTL by translating Gentzen derivations to Hilbert refutations and by giving a direct construction of fragments for the collection of (Gentzen-) underivable clauses over a given subformula universe. This required a relaxation of the fragment conditions in order to handle clauses with histories. The direct construction of fragments from underivable clauses turned out to be difficult to formalize. One of our reviewers described the proof as a “formal verification tour de force”. For the following invited journal version [DS15], we focused on the completeness result for the Hilbert system and on the small-model property (which was not treated in the conference paper). For this, we employed a pruning system based on inductive fulfillment as the decision method underlying the completeness result. Combining the relaxed fragment conditions employed in [DS14] with the inductive fulfillment relations introduced in [DS15] yields relaxed demos.

We have formalized all our results in the proof assistant Coq using the Ssreflect extension. Altogether the formalization consists of roughly 6800 lines. We believe that, at least for someone familiar with Coq, the formalization is accessible enough to provide additional insights into the proofs presented in this thesis. In fact, one of the reviewers for [DS15] remarked that she was “impressed by the quality of the formalization”. For the formalization, we profit greatly from the Ssreflect [GMT08] tactic language and from the libraries distributed with Ssreflect.

In retrospect, coming up with proofs that lead to an elegant and concise formalization was far more challenging than the subsequent work of carrying out the formalization. This is particularly true for the completeness proof for the Gentzen system for CTL. The translation from pruning refutations to Gentzen derivations takes a mere 150 lines to formalize. Everything else required to show completeness of the Gentzen system is shared with the completeness proof for the Hilbert system.

8.2 Future Work

The results presented in this thesis can be extended in a number of different directions. We briefly mention some directions we deem particularly interesting.

One possible direction would be to extend our results for K* to PDL [FL79, Pra79]. K* can be seen as a fragment of PDL where a and a^* (for some primitive program a) are the only programs. Adding the full language of programs significantly complicates the definition of subformula universes and the support relation. While the alpha-beta decomposition of formulas given by Fischer and Ladner [FL79] stays within a finite subformula universe, a naive recursive definition does not terminate (cf. [AGW09, Wid10, Kam12]). Hence, for PDL it may be easier to work with Hintikka sets rather than the recursive definition of support employed in this thesis. Alternatively, a terminating decomposition for PDL can be found in [Kam12]. Either approach should provide for constructive completeness proofs for Hilbert systems (e.g, the Hilbert system given by Segerberg [Seg77]). As suggested by Brünnler and Lange [BL08], it should also be possible to give a history-based Gentzen system for PDL.

Another interesting direction might be the extension of our results to hybrid logics. In the presence of nominals, using pruning as the basis for constructive completeness proofs becomes problematic. In earlier work [DS11], we have given a formal and constructive proof of decidability for H* (i.e., K* extended with nominals) based on pruning. The proof follows [KSS11] and establishes nominal consistency through a guessing stage prior to pruning. While pruning is incremental enough to construct refutations for clauses as they are removed, it appears unlikely that the guessing stage can be used to construct refutations. Therefore, incremental methods, such as the methods described in [Kam12], appear more promising as a means of obtaining constructive completeness proofs for hybrid logics.

More ambitious projects would be to obtain formal completeness proofs for the axiomatizations of the propositional μ-calculus [Koz83] or CTL* [EH86]. Both logics are decidable and have the small-model property. Consequently, both should be amenable to a constructive treatment. For the μ-calculus, a simple axiomatization was suggested in the original work [Koz83]. However, it took over a decade before completeness of this axiomatization was established [Wal95] and the proof is fairly involved. A simpler, but still involved, proof is suggested by Tamura [Tam14]. For CTL*, Reynolds gives a complete axiomatization [Rey01] and tableau-based decision methods [Rey11]. The axiomatization of CTL* is nonstandard in that it includes a rule that quantifies over sets of atomic propositions. As for the μ-calculus, the completeness proofs are fairly involved. A first step towards obtaining formal completeness proofs for either the μ-calculus or CTL* could be to obtain a formal completeness proof for ECTL [Kas14] (called $\mathsf{B}(\mathsf{F}, \mathsf{X}, \mathsf{U}, \mathsf{F}^\infty, \wedge, \neg)$ in [EH86]). ECTL has a fairly standard axiomatization and the completeness proof [Kas14] is relatively simple even though the embedding of ECTL in the μ-calculus involves nested alternating fixpoints [EL86].

Satisfiability of CTL* can also been characterized using games [FLL13]. These games are similar to the focus-games [LS01] for CTL. Given that focus-games for CTL can be used to “extract” [LS01] a complete Hilbert system, it might be worthwhile to investigate whether a simpler axiomatization for CTL* can be derived from the games for CTL*.

Beyond completeness and decidability, it would also be interesting to obtain formal proofs for some of the complexity results for modal and temporal logics. For this, one would need a formal model of computation that allows both programming and reasoning about programs with reasonable effort. One promising candidate for such a model is the weak call-by-value lambda calculus [LM08, For14].

Bibliography

[ACF] Coq formalization accompanying this thesis. `https://www.ps.uni-saarland.de/static/doczkal-diss`.

[AGW09] Pietro Abate, Rajeev Goré, and Florian Widmann. An on-the-fly tableau-based decision procedure for PDL-satisfiability. In Carlos Areces and Stéphane Demri, editors, *Proc. 5th Workshop on Methods for Modalities (M4M-5)*, volume 231 of *Electr. Notes Theor. Comput. Sci.*, pages 191–209. Elsevier, 2009.

[AV14] Andreas Abel and Andrea Vezzosi. A formalized proof of strong normalization for guarded recursive types. In Jacques Garrigue, editor, *Programming Languages and Systems (APLAS 2014)*, volume 8858 of *LNCS*, pages 140–158. Springer, 2014.

[BAPM83] Mordechai Ben-Ari, Amir Pnueli, and Zohar Manna. The temporal logic of branching time. *Acta Inf.*, 20:207–226, 1983.

[BDL06] Sandrine Blazy, Zaynah Dargaye, and Xavier Leroy. Formal verification of a C compiler front-end. In Jayadev Misra, Tobias Nipkow, and Emil Sekerinski, editors, *FM 2006: Formal Methods*, volume 4085 of *LNCS*, pages 460–475. Springer, 2006.

[BdRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001.

[Ber79] Francine Berman. A completeness technique for D-axiomatizable semantics. In Michael J. Fischer, Richard A. DeMillo, Nancy A. Lynch, Walter A. Burkhard, and Alfred V. Aho, editors, *Proc. 11h Annual ACM Symp. on Theory of Computing*, pages 160–166. ACM, 1979.

[Bet55] Evert W. Beth. *Semantic Entailment and Formal Derivability*. Noord-Hollandsche Uitg. Maatschappij, 1955.

[BGBP08] Yves Bertot, Georges Gonthier, Sidi Ould Biha, and Ioana Pasca. Canonical big operators. In Mohamed et al. [MMT08], pages 86–101.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[BL08] Kai Brünnler and Martin Lange. Cut-free sequent systems for temporal logic. *J. Log. Algebr. Program.*, 76(2):216–225, 2008.

[BNUW09] Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors. *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings*, volume 5674 of *LNCS*. Springer, 2009.

[BP12] Thomas Braibant and Damien Pous. Deciding kleene algebras in Coq. *Log. Meth. Comp. Sci.*, 8(1), 2012.

[BPP13] Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors. *Interactive Theorem Proving, 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *LNCS*. Springer, 2013.

[BR09] Stefan Berghofer and Markus Reiter. Formalizing the logic-automaton connection. In Berghofer et al. [BNUW09], pages 147–163.

[CE82] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Dexter Kozen, editor, *Logics of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1982.

[CES83] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. In John R. Wright, Larry Landweber, Alan J. Demers, and Tim Teitelbaum, editors, *Proc. 10th Annual ACM Symp. on Principles of Programming Languages (POPL '83)*, pages 117–126. ACM Press, 1983.

[CH88] Thierry Coquand and Gérard P. Huet. The calculus of constructions. *Inf. Comput.*, 76(2/3):95–120, 1988.

[CJNU00] Robert L. Constable, Paul B. Jackson, Pavel Naumov, and Juan C. Uribe. Constructively formalizing automata theory. In Gordon D. Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 213–238. The MIT Press, 2000.

[Coh13] Cyril Cohen. Pragmatic quotient types in Coq. In Blazy et al. [BPP13], pages 213–228.

[Coh15] Cyril Cohen. A finset and finmap DRAFT library. `https://github.com/Barbichu/finmap`, May 2015. Accessed Oct. 21st, 2015.

[Coq15] Coq Development Team. *The Coq Proof Assistant Reference Manual*, 8.4 edition, 2015.

[Cou81] Patrick Cousot. Semantic foundations of program analysis. In Steven S. Muchnick and Neil D. Jones, editors, *Program Flow Analysis: Theory and Applications*, chapter 10, pages 303–342. Prentice-Hall, 1981.

[CS11] Thierry Coquand and Vincent Siles. A decision procedure for regular expression equivalence in type theory. In Jouannaud and Shao [JS11], pages 119–134.

[dB68] Nicolaas G. de Bruijn. The mathematical language Automath, its usage, and some of its extensions. In *Symposium on Automatic Demonstration*, volume 125 of *LNCS*, pages 29–61. Springer, 1968.

[DKS13] Christian Doczkal, Jan-Oliver Kaiser, and Gert Smolka. A constructive theory of regular languages in Coq. In Georges Gonthier and Michael Norrish, editors, *Certified Programs and Proofs (CPP 2013)*, volume 8307 of *LNCS*, pages 82–97. Springer, 2013.

[DS11] Christian Doczkal and Gert Smolka. Constructive formalization of hybrid logic with eventualities. In Jouannaud and Shao [JS11], pages 5–20.

[DS12] Christian Doczkal and Gert Smolka. Constructive completeness for modal logic with transitive closure. In Chris Hawblitzel and Dale Miller, editors, *Certified Programs and Proofs (CPP 2012)*, volume 7679 of *LNCS*, pages 224–239. Springer, 2012.

[DS14] Christian Doczkal and Gert Smolka. Completeness and decidability results for CTL in Coq. In Gerwin Klein and Ruben Gamboa, editors, *Interactive Theorem Proving (ITP 2014)*, volume 8558 of *LNCS (LNAI)*, pages 226–241. Springer, 2014.

[DS15] Christian Doczkal and Gert Smolka. Completeness and decidability results for CTL in constructive type theory. `http://www.ps.uni-saarland.de/Publications/details/DoczkalSmolka:2014:CTL-completeness-pruning.html`, Accepted for publication in *J. Autom. Reason*, 2015.

[EC80] E. Allen Emerson and Edmund M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In J. W. de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming*, volume 85 of *LNCS*, pages 169–181. Springer, 1980.

[EC82] E. Allen Emerson and Edmund M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Programming*, 2(3):241-266, 1982.

[EH82] E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. In *Proc. 14th Annual ACM Symp. on Theory of Computing (STOC'82)*, pages 169–180. ACM, 1982.

[EH85] E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. System Sci.*, 30(1):1–24, 1985.

[EH86] E. Allen Emerson and Joseph Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.

[EL86] E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In *Proc. First Annual IEEE Symp. on Logic in Computer Science (LICS 1986)*, pages 267–278. IEEE Computer Society, 1986.

[Eme90] E. Allen Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Models and Sematics*, volume B, pages 995–1072. Elsevier, 1990.

[Eme08] E. Allen Emerson. The beginning of model checking: A personal perspective. In Orna Grumberg and Helmut Veith, editors, *25 Years of Model Checking - History, Achievements, Perspectives*, volume 5000 of *LNCS*, pages 27–45. Springer, 2008.

[Esc13] Martín Escardó. Infinite sets that satisfy the principle of omniscience in any variety of constructive mathematics. *J. Symb. Log.*, 78(3):764–784, 2013.

[Fit83] Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Reidel, 1983.

[Fit07] Melvin Fitting. Modal proof theory. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 85–138. Elsevier, 2007.

[FL79] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. System Sci.*, pages 194–211, 1979.

[FLL13] Oliver Friedmann, Markus Latte, and Martin Lange. Satisfiability games for branching-time logics. *Log. Meth. Comp. Sci.*, 9(4), 2013.

[For14] Yannick Forster. *A Formal and Constructive Theory of Computation*. Bachelor's thesis, Saarland University, 2014.

[GAA$^+$13] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A machine-checked proof of the odd order theorem. In Blazy et al. [BPP13], pages 163–179.

[Gab77] Dov M. Gabbay. Axiomatization of logic programs, 1977. Text of a letter to V. Pratt.

[GGMR09] François Garillot, Georges Gonthier, Assia Mahboubi, and Laurence Rideau. Packaging mathematical structures. In Berghofer et al. [BNUW09], pages 327–342.

[GHL$^+$07] Joxe Gaintzarain, Montserrat Hermo, Paqui Lucio, Marisa Navarro, and Fernando Orejas. A cut-free and invariant-free sequent calculus for PLTL. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic (CSL 2007)*, volume 4646 of *LNCS*, pages 481–495. Springer, 2007.

[Gim94] Eduardo Giménez. Codifying guarded definitions with recursive schemes. In Peter Dybjer, Bengt Nordström, and Jan M. Smith, editors, *Types for Proofs and Programs (TYPES'94)*, volume 996 of *LNCS*, pages 39–59. Springer, 1994.

[GMR$^+$07] Georges Gonthier, Assia Mahboubi, Laurence Rideau, Enrico Tassi, and Laurent Théry. A modular formalisation of finite group theory. In Klaus Schneider and Jens Brandt, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2007)*, volume 4732 of *LNCS*, pages 86–101. Springer, 2007.

[GMT08] Georges Gonthier, Assia Mahboubi, and Enrico Tassi. A small scale reflection extension for the Coq system. Research Report RR-6455, INRIA, 2008.

[Göd33] Kurt Gödel. Eine interpretation des intuitionistischen aussagenkalküls. In *Ergebnisse eines mathematischen Kolloquiums*, volume 4, pages 34–40, 1933.

[Gol91] Robert Goldblatt. *Logics of Time and Computation*. Number 7 in CSLI Lecture Notes. Center for the Study of Language and Information, 2nd edition, 1991.

[Gol06] Robert Goldblatt. Mathematical modal logic: A view of its evolution. In Dov M. Gabbay and John Woods, editors, *Logic and the Modalities in the Twentieth Century*, volume 7 of *Handbook of the History of Logic*, pages 1–98. Elsevier, 2006.

[Gon08] Georges Gonthier. Formal proof — the four-color theorem. *Notices Amer. Math. Soc.*, 55(11):1382–1393, 2008.

[HAB+15] Thomas C. Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, and Roland Zumkeller. A formal proof of the kepler conjecture. *CoRR*, abs/1501.02155, 2015.

[Her12] Hugo Herbelin. A constructive proof of dependent choice, compatible with classical logic. In *Proc. 27th Annual IEEE Symp. on Logic in Computer Science (LICS 2012)*, pages 365–374. IEEE Computer Society, 2012.

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. The MIT Press, 2000.

[JS11] Jean-Pierre Jouannaud and Zhong Shao, editors. *Certified Programs and Proofs, First International Conference, CPP 2011, Kenting, Taiwan, December 2011, Procedings*, volume 7086 of *LNCS*. Springer, 2011.

[Kam12] Mark Kaminski. *Incremental Decision Procedures for Modal Logics with Nominals and Eventualities*. PhD thesis, Saarland University, 2012.

[Kas10] Ryo Kashima. Completeness proof by semantic diagrams for transitive closure of accessibility relation. In Lev D. Beklemishev, Valentin Goranko, and Valentin B. Shehtman, editors, *Advances in Modal Logic 8*, pages 200–217. College Publications, 2010.

[Kas14] Ryo Kashima. An axiomatization of ECTL. *J. Log. Comput.*, 24(1):117–133, 2014.

[KDE09] Gerwin Klein, Philip Derrin, and Kevin Elphinstone. Experience report: sel4: formally verifying a high-performance microkernel. In Graham Hutton and Andrew P. Tolmach, editors, *Proc. 14th ACM SIGPLAN Intl. Conf. on Functional Programming (ICFP 2009)*, pages 91–96. ACM, 2009.

[Koz83] Dexter Kozen. Results on the propositional μ-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[KP81] Dexter Kozen and Rohit Parikh. An elementary proof of the completness of PDL. *Theor. Comput. Sci.*, 14:113–118, 1981.

[Kri59] Saul A. Kripke. A completeness theorem in modal logic. *J. Symb. Log.*, 24(1):1–14, 1959.

[Kri63] Saul A. Kripke. Semantical analysis of modal logic I: Normal modal propositional calculi. *Z. Math. Logik Grundlagen Math.*, 9:67–96, 1963.

[KS10] Mark Kaminski and Gert Smolka. Terminating tableaux for hybrid logic with eventualities. In Jürgen Giesl and Reiner Hähnle, editors, *Automated Reasoning (IJCAR 2010)*, volume 6173 of *LNCS*, pages 240–254. Springer, 2010.

[KS14] Mark Kaminski and Gert Smolka. A goal-directed decision procedure for hybrid PDL. *J. Autom. Reason.*, 52(4):407–450, 2014.

[KSS11] Mark Kaminski, Thomas Schneider, and Gert Smolka. Correctness and worst-case optimality of Pratt-style decision procedures for modal and hybrid logics. In Kai Brünnler and George Metcalfe, editors, *TABLEAUX 2011*, volume 6793 of *LNCS (LNAI)*, pages 196–210. Springer, 2011.

[Lad77] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, 6(3):467–480, 1977.

[Ler06] Xavier Leroy. Formal certification of a compiler back-end or: programming a compiler with a proof assistant. In J. Gregory Morrisett and Simon L. Peyton Jones, editors, *Proc. 33rd ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL 2006)*, pages 42–54. ACM, 2006.

[Les11] Stéphane Lescuyer. First-class containers in Coq. *Stud. Inform. Univ.*, 9(1):87–127, 2011.

[Lew18] Clarence I. Lewis. *A Survey of Symbolic Logic*. Univerisity of California Press, 1918.

[LL32] Clarence I. Lewis and Cooper H. Langford. *Symbolic Logic*. Dover, 1932.

[LM08] Ugo Dal Lago and Simone Martini. The weak lambda calculus as a reasonable machine. *Theor. Comput. Sci.*, 398(1-3):32–50, 2008.

[LS77] Edward J. Lemmon and Dana Scott. *The 'Lemmon Notes': An Introduction to Modal Logic*. Blackwell, 1977.

[LS01] Martin Lange and Colin Stirling. Focus games for satisfiability and completeness of temporal logic. In *Proc. 16th Annual IEEE Symp. on Logic in Computer Science (LICS 2001)*, pages 357–365. IEEE Computer Society, 2001.

[Luo89] Zhaohui Luo. ECC, an extended calculus of constructions. In *Proc. Fourth Annual Symp. on Logic in Computer Science (LICS '89)*, pages 386–395. IEEE Computer Society, 1989.

[Luo94] Zhaohui Luo. *Computation and Reasoning*. Oxford Science Publications, 1994.

[LW11] Gyesik Lee and Benjamin Werner. Proof-irrelevant model of CC with predicative induction and judgmental equality. *Log. Meth. Comp. Sci.*, 7(4), 2011.

[ML84] Per E. R. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[MMT08] Otmane Aït Mohamed, César Muñoz, and Sofiène Tahar, editors. *Theorem Proving in Higher Order Logics, 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings*, volume 5170 of *LNCS*. Springer, 2008.

[MP79] Zohar Manna and Amir Pnueli. The modal logic of programs. In Hermann A. Maurer, editor, *Automata, Languages and Programming*, volume 71 of *LNCS*, pages 385–409. Springer, 1979.

[MT13] Assia Mahboubi and Enrico Tassi. Canonical structures for the working Coq user. In Blazy et al. [BPP13], pages 19–34.

[Nak00] Hiroshi Nakano. A modality for recursion. In *Proc. 15th Annual IEEE Symp. on Logic in Computer Science (LICS 2000)*, pages 255–266. IEEE Computer Society, 2000.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Makarius Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[Par78] Rohit Parikh. The completeness of propositional dynamic logic. In Józef Winkowski, editor, *Mathematical Foundations of Computer Science*, volume 64 of *LNCS*, pages 403–415. Springer, 1978.

[Pau15] Lawrence C. Paulson. A formalisation of finite automata using hereditarily finite sets. In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction (CADE-25)*, volume 9195 of *LNCS*, pages 231–245. Springer, 2015.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symp. on Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEE Computer Society Press, 1977.

[Pra79] Vaughan R. Pratt. Models of program logics. In *Proc. 20th Annual Symp. on Foundations of Computer Science (FOCS'79)*, pages 115–122. IEEE Computer Society Press, 1979.

[Rey01] Mark Reynolds. An axiomatization of full computation tree logic. *J. Symb. Log.*, 66(3):1011–1057, 2001.

[Rey11] Mark Reynolds. A tableau-based decision procedure for CTL*. *Formal Asp. Comput.*, 23(6):739–779, 2011.

[Saï99] Amokrane Saïbi. *Outils Génériques de Modélisation et de Démonstration pour la Formalisation des Mathématiques en Théorie des Types: application à la Théorie des Catégories.* PhD thesis, Université Paris VI, 1999.

[SB14] Gert Smolka and Chad Brown. Introduction to computational logic. Lecture Notes, Saarland University, `https://www.ps.uni-saarland.de/courses/cl-ss14/script/icl.pdf`, 2014.

[Seg71] Krister Segerberg. *An Essay in Classical Modal Logic.* Number 13 in Filosofiska Studier. University of Uppsala, 1971.

[Seg77] Krister Segerberg. A completeness theorem in the modal logic of programs. *Notices Amer. Math. Soc.*, 24:A–552, 1977.

[Seg82] Krister Segerberg. A completeness theorem in the modal logic of programs. In T. Traczyk, editor, *Universal Algebra and Applications*, volume 9 of *Banach Centre Publications*, pages 31–46. PWN-Polish Scientific Publishers, 1982.

[SH12] Vincent Siles and Hugo Herbelin. Pure type system conversion is always typable. *J. Funct. Program.*, 22(2):153–180, 2012.

[Smu68] Raymond M. Smullyan. *First-Order Logic.* Springer, 1968.

[SO08] Matthieu Sozeau and Nicolas Oury. First-class type classes. In Mohamed et al. [MMT08], pages 278–293.

[Soz09] Matthieu Sozeau. A new look at generalized rewriting in type theory. *J. Form. Reason.*, 2(1), 2009.

[Str14] Pierre-Yves Strub. A finset theory over a choice type. `https://github.com/strub/ssrmisc/blob/master/fset.v`, Dec. 2014. Accessed Oct. 21st, 2015.

[Tam14] Kuniaki Tamura. Completeness of kozen's axiomatization for the modal mu-calculus: A simple proof. *CoRR*, abs/1408.3560, 2014.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics.* `http://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

[vBJ77] L. S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the AUTOMATH system.* PhD thesis, Technische Hogeschool Eindhoven, 1977.

[Wad15] Philip Wadler. Propositions as types. *Commun. ACM*, 2015. to appear.

[Wal95] Igor Walukiewicz. Completeness of kozen's axiomatisation of the propositional mu-calculus. In *Proc. 10th Annual IEEE Symp. on Logic in Computer Science (LICS'95)*, pages 14–24. IEEE Computer Society, 1995.

[Wer94] Benjamin Werner. *Une théorie des Constructions Inductives*. PhD thesis, Université Paris, 1994.

[Wid10] Florian Widmann. *Tableaux-based Decision Procedures for Fixed Point Logics*. PhD thesis, Australian National University, 2010.

[WZU14] Chunhan Wu, Xingyuan Zhang, and Christian Urban. A formalisation of the myhill-nerode theorem based on regular expressions. *J. Autom. Reasoning*, 52(4):451–480, 2014.

List of Notations

$M : N$	M has type N, page 15
$M + N$	sum type, page 17
$\Sigma x : M.\, NB$	sigma type, page 17
$\varepsilon\, p$	canonical element of p, page 19
2^A	powerset of A, page 20
$\lvert\mathcal{M}\rvert$	states of the model $\mathcal{M}$, page 22
$\Rightarrow_{\mathcal{M}}$	transition relation of the model $\mathcal{M}$, page 22
$\Lambda_{\mathcal{M}}$	labeling relation of the model $\mathcal{M}$, page 22
$w \in \mathcal{M}$	w is a state of $\mathcal{M}$, page 23
$\mathbf{A} \vdash s$	s is theorem of the Hilbert system **A**, page 38
s^σ	signed formula, page 25
$\lfloor s^\sigma \rfloor$	unsigned equivalent of signed formula, page 25
$C \triangleright s^\sigma$	C supports s^σ (K), page 26
$C \triangleright s^\sigma$	C supports s^σ (K*), page 40
$C \triangleright s^\sigma$	C supports s^σ (CTL), page 54
$C \triangleright D$	C supports every formula in D, page 26
$S \triangleright C$	some clause in S supports C, page 26
$\mathcal{R}\, C$	request of C, page 26
$S, C \triangleright \Box\Box^* s^-$	inductive fulfillment (K*), page 40
$S, C \triangleright e$	inductive fulfillment (CTL), page 55
$(S, \mathcal{T}), C \rhd e$	relaxed fulfillment (CTL), page 57
$S, C \rhd \Box\Box^* s^-$	relaxed fulfillment (K*), page 88
$\overline{U}$	base clauses over U, page 28
$\mathsf{prune}\, p\, X$	pruning of X with respect to p, page 28
$\mathcal{D}(U)$	canonical demo for U, page 29
$\mathcal{B}_S\, C$	base of C in S, page 30
$\mathsf{Dem}\, C$	demands of C, page 54