BMBF

Verbmobil
Verbundvorhaben

Research on Architectures for Integrated Speech/Language Systems in Verbmobil

Günther Görz, Marcus Kesseler,
Jörg Spilker, Hans Weber

IMMD VIII/UER

Vm Report 126
Mai 1996

Mai 1996

Günther Görz, Marcus Kesseler, Jörg Spilker, Hans Weber

IMMD VIII – Künstliche Intelligenz
Universität Erlangen-Nürnberg
Am Weichselgarten 9
D-91058 Erlangen

Tel.: 09131/8599 - 09
Fax: 09131/8599 - 05
e-mail: {goerz,kesseler,spilker,weber}@immd8.informatik.uni-erlangen.de

Gehört zum Antragsabschnitt: 15.7: Architektur integrierter Parser für gesprochene Sprache

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) unter dem Förderkennzeichen 01 IV 101 H9 gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei den Autoren.

Research on Architectures for Integrated Speech/Language Systems in Verbmobil

Günther Görz, Marcus Kesseler, Jörg Spilker, Hans Weber
University of Erlangen-Nürnberg
IMMD (Computer Science) VIII — Artificial Intelligence
Am Weichselgarten 9
D-91058 ERLANGEN
Email: {goerz,kesseler,spilker,weber}@informatik.uni-erlangen.de

Abstract

The German joint research project Verbmobil (VM) aims at the development of a speech to speech translation system. This paper reports on research done in our group which belongs to Verbmobil's subproject on system architectures (TP15). Our specific research areas are the construction of parsers for spontaneous speech, investigations in the parallelization of parsing and to contribute to the development of a flexible communication architecture with distributed control.

1 Introduction

The German joint research project Verbmobil (VM)[1] aims at the development of a speech to speech translation system. This paper reports on research done in our group which belongs to Verbmobil's subproject on system architectures (TP15). The task of this subproject is to provide basic research results on incremental and interactive system architectures for the VM research prototype and to demonstrate their feasibility in the prototypical INTARC system. Our specific research areas are the construction of parsers for spontaneous speech, investigations in the parallelization of parsing and to contribute to the development of a flexible communication architecture with distributed control. The paper is organized as follows: Section 2 reports on the design and implementation of an incremental interactive speech parser which integrates statistics with a chart-parser employing a unification grammar (UG) formalism. Furthermore, results of experiments on the interaction between the parser and a speech recognizer using expectations, are reported. In section 3 we present experiences with a parallel version of the parser. Section 4 deals with distributed control in modular Natural Language/Speech (NLSP) systems.

2 Design and Implementation of Incremental Interactive Speech Parsers

In a Left Right Incremental architecture (LRI), higher level modules can work in parallel with lower level modules. The obvious benefits of such an arrangement are twofold: The system does not have to wait for a speaker to stop talking and top-down constraints from higher level to lower level modules can be used easily. To achieve LRI behavior the singular modules must fulfill the following requirements:

Processing proceeds incrementally along the time axis ("left to right").

Pieces of output have to be transferred to the next module as soon as possible.

So far in INTARC-1.3 we have achieved an LRI style coupling of four different modules: Word recognition module, syntactic parser, semantic module and prosodic boundary module. Our word recognition module is a modified Viterbi decoder, where two changes in the algorithm design were made: We use only the forward search pass, and whenever a final HMM state is reached for an active word model, a corresponding word hypothesis is sent to the parser. Hence backward search becomes a part of the parsing algorithm. The LRI parsing algorithm is a modified active chart parser with an agenda driven control mechanism. The chart vertices correspond to the frames of the signal representation. Edges correspond to word or phrase hypotheses, being partial in the case of active edges. A parsing cycle corresponds to a new time point related to the utterance. In every cycle a new vertex is created and new word hypotheses ending at that time point are read and inserted into the chart. In one cycle, a backward search is per-

[1]This work was funded by the German Federal Ministry for Research and Technology (BMFT) in the framework of the Verbmobil Project under Grant BMFT 01 IV 101 H / 9. The responsibility for the contents of this study lies with the authors.

formed to the beginning of the utterance or to some designated time point in the past constituting a starting point for grammatical analysis. Search is guided by a weighted linear combination of acoustic score, bigram score, prosody score, grammar derivation score and grammatical parsability. The search prodecure is a beam search implemented as an agenda access mechanism. The grammar is a probabilistic typed UG with separate rules for pauses and other spontanous speech phemomena.

2.1 Basic Objects

In the following we use record notation to refer to subcomponents of an object. A chart vertex V_t corresponds to frame number t. Vertices have four lists with pointers to edges ending in and starting in that vertex: *inactive-out, inactive-in, active-out* and *active-out*. A word hypothesis W is a quadruple *(from, to, key, score)* with *from* and *to* being the start and end frames of W. *W.Key* is the name of the lexical entry of W and *W.score* is the acoustic score of W for the frames spanned, given by a corresponding HMM acoustic word model. An edge E consists of *from*, the start vertex and *to*, a list of end vertices. Note that after a Viterbi forward pass identical word hypotheses do always come in sequence, differing only in ending time. *E.actual* is the last vertex added to *E.to* in an operation. Those "families" of hypotheses are represented as one edge with a set of end vertices. *E.words* keeps the covered string of word hypotheses while SCORE is a record keeping score components. Besides that an edge consists of a grammar rule *E.rule* and *E.next*, a pointer to some element of the right hand side of *E.rule* or *NIL*. As in standard active chart parsing an edge is passive, if *E.next = nil*, otherwise it is active. *E.cat* points to the left hand side of the grammar rule. SCORE is a record with entries for inside and outside probabilities given to an edge by acoustic, bigram, prosody and grammar model:

Inside-X Model scores for the spanned portion of an edge.

Outside-X Optimistic estimates for the portion from vertex 0 to the beginning of an edge.

For every vertex we keep a best first store of scored edge pairs. We call that store Agenda_i in cycle i.

2.2 Basic Operations

There are five basic operations to define the operations of the parsing algorithm. The two operations *Combine* and *Seek Down* are similar to the well known Earley algorithm operations *Completer* and *Predictor*. Furthermore, there are two operations to insert new word hypotheses, *Insert* and *Inherit*. All these operations can create new edges, so operations to calculate new scores from old ones are attached to them. In order to implement our beam search method appropriately but simply, we define an operation *Agenda-Push*, which selects pairs of active and passive edges to be pruned or to be processed in the future. The basic operations are given in CFG notation for simplicity.

2.2.1 Combine

For a pair of active and passive edges (A, I), if $A.next = I.cat$ and $I.from \in A.to$, insert edge E with *E.rule = A.rule, E.cat = A.cat, E.next = shift(A.next), E.from = A.from, E.to = A.to.*
For X = Bigram, Grammar and Prosody:
$E.Outside\text{-}X = A.Outside\text{-}X + I.Inside\text{-}X + Trans(X,A,I)$
$E.Inside\text{-}X = A.Inside\text{-}X + I.Inside\text{-}X + Trans(X,A,I)$
For X = Acoustic:
$E.Outside\text{-}X = A.Outside\text{-}X[I.from] \oplus I.Inside\text{-}X$ $Trans(X,A,I)$
$E.Inside\text{-}X = A.Inside\text{-}X[I.from] \oplus I.Inside\text{-}X$ $Trans(X,A,I)$
The operator $\oplus$ performs an addition of a number to every element of a set. *Trans(X,A,I)* is the specific transition penalty a model will give to two edges. In the case of acoustic scores, the penalty is always zero and can be neglected. In the bigram case it will be the transition from the last word covered by A to the first word covered by B.

2.2.2 Seek Down

Whenever an active edge A is inserted, insert an edge E for every rule R such that $A.next = E.cat$, $E.rule = R$, $E.from = A.actual$, $E.to = \{A.actual\}$.
For X = Acoustic, Prosody and Bigram:
$E.Inside\text{-}X = 0$
$E.Outside\text{-}X = A.Outside\text{-}X$
For X = Grammar:
$E.Inside\text{-}X =$ *grammar score of R*
$E.Outside\text{-}X = A.Outside\text{-}X + Trans(X,A,E) + E.Inside\text{-}X$. This recursive operation of introducing new active edges is precompiled in our parser and extremely efficient.

2.2.3 Insert

For a new word hypothesis $W = (a,i,key,score)$ such that no $W' = (a,i\text{-}1,key,score')$ exists, insert an edge E with $E.rule = lex(key)$, $E.cat = lex(key)$, $E.from = V_a$, $E.to = \{V_i\}$ and for X = Acoustic:
$E.Inside\text{-}X = E.Outside\text{-}X = \{(i,score)\}$,
for X = Prosody and Bigram:
$E.Inside\text{-}X = E.Outside\text{-}X = 0,$
for X *= Grammar* $E.Inside\text{-}X = E.Outside\text{-}X =$ *grammar score of lex(key).*

2.2.4 Inherit

For a new word hypothesis W = *(a,i,key,score)* such that a *W' = (a,i-1,key,score')* exists:
For all E in V_{i-1}*.inactive-in* or V_{i-1}*.active-in*: If *last(E.words) = key* then add $\{V_i\}$ to *E.to*, add *(i,E.Inside-Acoustic[i-1] - score' + score)* to *E.Inside-Acoustic* and add *(i,E.Outside-Acoustic[i-1] - score' + score)* to *E.Outside-Acoustic*.
If E is active, perform a *Seek-Down* on E in V_i.

2.2.5 Agenda Push

Whenever an edge E is inserted into the chart, if E is active then for all passive A, such that *A.from* $\in$ *E.to* and *combined-score(E,A)* $>$ *Beam-Value*, insert *(E,A,combined-score(E,A))* into the actual agenda. If E is passive then for all active A, such that *E.from* $\in$ *A.to* and *combined-score(A,E)* $>$ *Beam-Value*, insert *(A,E,combined-score(A,E))* into the actual agenda. *Combined-Score* is a linear combination of the outside components of an edge C which would be created by A and E in a *Combine* operation. *Beam-Value* is calculated as a fixed offset from the maximum *Combined-Score* on an agenda. Since we process best-first inside the beam, the maximum is known when the first triple is inserted into an agenda. *Agenda-Pop* will remove the best triple from an actual agenda and return it.

2.3 A simple LRI lattice parser

The follwing control loop implements a simple LRI lattice parser.

1. $T = 0$. Create V_T
2. Insert initial active edge E into V_T, with *E.next* $= S$
3. Increment T. Create V_T
4. For every W with *W.end = T*: *Insert(W)* or *Inherit(W)*
5. Until *Agenda[T]* is empty:
   (a) *Combine(Agenda-Pop)*
   (b) When combination with initial edge is successful, send result to SEMANTICS
6. Communicate with PROSODY and go to 3

2.4 The Grammar Model

The UG used in our experiments consists of 700 lexical entries and 60 rules. We used a variant of inside-outside training to estimate a model of UG derivations. It is a rule bigram model similar to PCFG with special extensions for UG type operations. The probability of future unifications is made dependent from the result type of earlier unifications. The model is described in more detail in (Weber 1994a; Weber 1995); it is very similar to (Brew 1995).

2.5 LRI Coupling with Prosody

In INTARC we use three classes of boundaries, B0 (no boundary), B2 (phrase boundary), B3 (sentence boundary) and B9 (real break). The prosody module, developed at the University of Bonn, classifies time intervals according to these classes. A prosody hypothesis consists of a beginning and ending time and model probabilities for the boundary types which sum up to one. A prosodic transition penalty used in the *Combine* operation was taken to be the score of the best combination of bottom-up boundary hypothesis *Bx* and a trigram score *(lword, Bx, rword)*. Here *lword* is the last word of the edge to the left and *rword* is the first word spanned by the edge to the right. Prosody hypotheses are consumed by the parser in every cycle and represented as attributes of vertices which fall inside a prosodic time interval. In a couple of tests we already achieved a reduction of edges of about 10% without change in recognition rate using a very simple trigram with only five word categories.

2.6 Experimental Results

In a system like INTARC-1.3, the analysis tree is of much higher importance than the recovered string; for the goal of speech translation an adequate semantic representation for a string with word errors is more important than a good string with a wrong reading. The grammar scores have only indirect influence on the string; their main function is picking the right tree. We cannot measure something like a "tree recognition rate" or "rule accuracy", because there is no treebank for our grammar. The word accuracy results cannot be compared to word accuracy as usually applied to an acoustic decoder in isolation. We counted only those words as recognized which could be built into a valid parse from the beginning of the utterance. Words to the right which could not be integrated into a parse, were counted as deletions — although they might have been correct in standard word accuracy terms. This evaluation method is much harder than standard word accuracy, but it appears to be a good approximation to "rule accuracy". Using this strict method we achieved a word accuracy of 47%, which is quite promising.

Results using top down prediction of possible word hypotheses by the parser – work inspired by (Kita et. al. 1989) – have already been published in (Hauenstein and Weber 1994a; Hauenstein and Weber 1994b), (Weber 1994a), and (Weber 1995). Recognition rates had been improved there for read speech. In spontaneous speech we could not achieve the same effects.

2.7 Current Work

Our current work, which led to INTARC-2.0, uses a new approach for the interaction of syntax and semantics and a revision of the interaction of the parser with a new decoder. For the last case we implemented a precompiler for word-based prediction which to our current experience is clearly superior to the previous word-class based prediction. For the implementation of the interaction of syntax and semantics we proceed as follows: A new turn-based UG has been written, for which a context-sensitive stochastic training is being performed. The resulting grammar is then stripped down to a pure type skeleton which is actually being used for syntactic parsing. Using full structure sharing in the syntactic chart, which contains only packed edges, we achieve a complexity of $O(n^3)$. In contrast to that, for semantic analysis a second, unpacked chart is used, whose edges are provided by an unpacker module which is the interface between the two analysis levels. The unpacker, which has exponential complexity, selects only the n best scored packed edges, where n is a parameter. Only if semantic analysis fails it requests further edges from the unpacker. In this way, the computational effort on the whole is kept as low as possible.

3 Parallel Parsing

One of our main research interests has been the exploration of performance gains in NLP through parallelization. To this end, we developed a parallel version of the INTARC parser. Although the results so far are yet not as encouraging as we expected, our efforts make for interesting lessons in software engineering. The parallel parser had to obey the following restrictions: Running on our local shared memory multiprocessor (SparcServer1000) with 6 processors, parallelization should be controlled by inserting Solaris-2.4 thread and process control primitives directly into the code. The only realistic choice we had was to translate our parser with Chestnut Inc.'s Lisp-to-C-Translator *automatically* into C. Since the Lisp functions library is available in C source, we could insert the necessary Solaris parallelisation and synchronization primitives into key positions of the involved functions.

3.1 Parallelization Strategy and Preliminary Results

For effective parallelization it is crucial to keep communication between processors to a minimum. Early experiments with a fully distributed chart showed that the effort required to keep the partial charts consistent was much larger that the potential gains of increased parallelism. The chart must be kept as a single data structure in a shared memory processor, where concurrent reads are possible and only concurrent writes have to be serialized with synchronisation primitives. An analysis of profiling data shows that even the heavily optimized UG formalism causes between 50% -and 70% of the computational load in the serial case. Therefore we provide an arbitrary number of *unification workers* running in parallel which are fed unification tasks from the top of an agenda sorted by scores. Due to the high optimization level of the sequential parser, load-balancing is fairly poor. Namely, the very fast type check used to circumvent most unifications, causes large disparities in the granularity of agenda tasks. Furthermore, pathological examples have been found in which a single unification takes much longer than all other tasks combined.

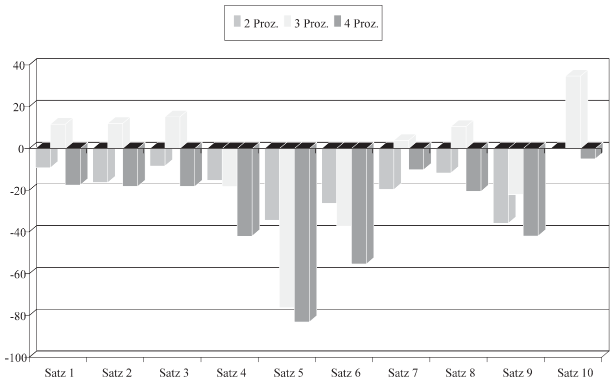

Figure 1: Percentual gains and losses over attained over 10 different sentences (Spilker 1995)

4 Distributed Control in Verbmobil

The question of control in VM is tightly knit with the architecture of the VM system. As yet, the concept of architecture in VM has been used mostly to describe the overall modularization and the interfaces implied by the data flow between modules. This so-called *domain architecture* is incomplete in the sense that it does not specify any *interaction strategies*. Within our research on interactive system architectures we developed a modular communication framework, ICE[2], in cooperation with the University of Hamburg. Now, ICE is the architectural framework of the VM research prototype.

4.1 The INTARC Architecture

The INTARC architecture as first presented by (Pyka 1992) is a distributed software system that allows for the interconnection of NLSP modules under the principles of incrementality and interactivity. Figure 2 shows the modularization of INTARC-1.3: There

[2]based on PVM (parallel virtual machine)

is a main broad channel connecting all modules in bottom-up direction, i.e., from signal to interpretation. Furthermore, there are smaller channels connecting several modules, which are used for the top-down interactive disambiguation data flow. Incrementality is required for all modules. ICE assumes

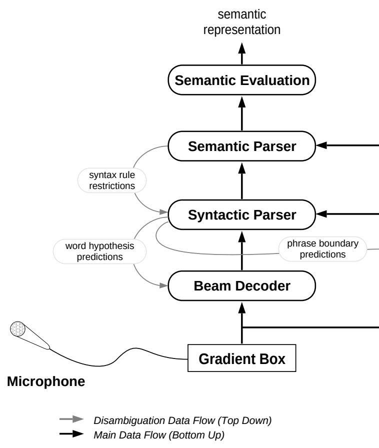

Figure 2: The interactive, incremental INTARC-1.3 architecture

that each module has a local memory that is not directly accessible to other modules. Modules communicate explicitly with one another via messages sent over bidirectional channels. This kind of communication architecture is hardly new and confronts us directly with a large number of unresolved issues in distributed problem solving, cf. (Durfee et al. 1989). In the last 20 years there have been numerous architecture proposals for distributed problem solving among computing entities that exchange information explicitly via message passing. None of these models include explicit strategies or paradigms to tackle the problem of distributed control.

4.2 Structural Constraints of Verbmobil

Modularity, being a fundamental assumption in VM (Wahlster 1992), does still leave us with two problems: First, modules have to communicate with one another, and second, their local behaviors have to be somehow coordinated into a coherent global, possibly optimal, behavior. Unfortunately, the task of system integration has to obey some *structural constraints* which are mostly pragmatic in nature:

- Some of the modules are very complex software systems in themselves. Highly parameterizable and with control subtly spread over many interacting submodules, understanding and then integrating such systems into a common control strategy can be a very daunting task.

- Control issues are often very tightly knit with the domain the module is aimed at, i.e., it is very difficult to understand the control strategies used without sound knowledge of the underlying domain. The problem even gets worse if what is to be fine-tuned is the *interaction* between several complex modules.

These two arguments are similar in nature, but differ in the architectural levels that they apply to. The former is implementation related, the latter algorithm and theory related.

4.3 Layers of Control

Modules have to communicate with one another and their local behaviors have to be coordinated into a coherent global, possibly optimal, behavior. In highly distributed systems we generally find the following levels of control:

System Control: The minimal set of operating system related actions that each participating module must be able to perform which will typically include means to start up, reset, monitor, trace and terminate individual modules or the system as a whole.

Isolated Local Control: The control strategies used within the module disregarding any interactions beyond initial input of data and final output of solutions. There is only one thread of control active at any time.

Interactive Local Control: Roughly, this can be seen as isolated local control extended with interaction capabilities. *Incrementality* is given by the possibility of control flowing back to a certain internal state after an output operation. Higher *interactivity* is made possible by entering a state more often from various points within the module and by adding a new waiting loop to check for any top-down requests. The requirement for anytime behavior is a special case of that (Görz and Kesseler 1994).

In our experience the change to interactive control will tremendously increase the complexity of the resulting code. But we are still making the simplifying assumptions that the algorithm can be used incrementally — but there are algorithms unsuitable for incremental processing (e.g. A^*). Incrementality can lead to the demand for a complete redesign of a module. Furthermore we assume that simply by exchanging data and doing simple extensions in the control flow everything will balance out nicely on the system scale which is enormously naive. Even for the sequential architecture implied by the case of isolated local control, we have to solve a whole plethora of new problems that come along with interactivity:

- Mutual deadlock
- Mutual live-lock
- Race conditions (missing synchronization)
- Over-synchronization

Dialogue Control: In systems like VM there *is* a module that comes close to possessing the "integrated view" of a centralized blackboard control: the *dialogue module*. So it seems the right place to handle some of the global strategic control issues, like:

- Domain error handling
- Observe timeout constraints
- Resolve external ambiguities/unknowns

The fact that the dialogue module exercises a kind of *global control* does not invalidate what has been said about the unfeasability of central control, because the control exercised by it is very coarse grained. To handle finer grained control issues in any module would take us back to memory and/or communication system contention.

References

Chris Brew. *Stochastic HPSG.* Proceedings of the European ACL Conference 1995, Edinburgh, 1995

Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. *Cooperative Distributed Problem Solving*, pages 83–147. Volume 4 of Avron Barr et al. (Ed.). *The Handbook of Artificial Intelligence* Reading, Mass.: Addison-Wesley, 1989.

Günther Görz and Marcus Kesseler. *Anytime Algorithms for Speech Parsing?* Proceedings of COLING-94, Kyoto, 1994

Andreas Hauenstein and Hans Weber. *An investigation of tightly coupled time synchronous speech language interfaces using a unification grammar.* In: P. McKevitt (Ed.): Proceedings of the Workshop on Integration of Natural Language and Speech Processing, AAAI-94, Seattle, 1994, 42–49

Andreas Hauenstein and Hans Weber. *An Investigation of Tightly Coupled Time Synchronous Speech Language Interfaces.* Proceedings of KONVENS 94, Vienna: Springer, September 1994, 141–150

Marcus Kesseler. *Distributed Control in Verbmobil.* University of Erlangen-Nürnberg, IMMD VIII, Verbmobil Report 24, 1994.

Kita, K., Kawabata, T. and Saito, H. *HMM contiuous speech recognition using predictive LR parsing. IEEE ICASSP Proceedings*, 1989, 703–706.

Claudius Pyka. *Management of hypotheses in an integrated speech-language architecture. Proceedings of ECAI-92*, 1992, 558–560

Jörg Spilker. *Parallelisierung eines inkrementellen, aktiven Chartparsers.* Diploma thesis, University of Erlangen-Nürnberg, Erlangen, December 1995.

Wolfgang Wahlster and Judith Engelkamp, editors. *Wissenschaftliche Ziele und Netzpläne für das VERBMOBIL-Projekt.* DFKI, Saarbrücken, 1992.

Hans Weber. *Time Synchronous Chart Parsing of Speech Integrating Unification Grammars with Statistics.* Speech and Language Engineering, Proceedings of the Eighth Twente Workshop on Language Technology, (L. Boves, A. Nijholt, Ed.), Twente, 1994, 107–119

Hans Weber. *LR-inkrementelles probabilistisches Chartparsing von Worthypothesenmengen mit Unifikationsgrammatiken: Eine enge Kopplung von Suche und Analyse.* Ph.D. Thesis, University of Hamburg, 1995, Verbmobil Report 52.