

Time Synchronous Chart Parsing of Speech Integrating Unification Grammars with Statistics[†]

Hans Weber
IMMD 8
University Erlangen-Nurnberg
Germany
email: weber@faii80.informatik.uni-erlangen.de

December 1st 1994

Abstract

We present an active chart parser which parses left connected wordgraphs in a strictly time synchronous way. The parser performs a beam search on the possible paths through the word graph and on the possible derivations of the unification grammar simultaneously. A metric is given to assign scores to edges, taking into account the whole left context thereby combining acoustic probabilities, n-gram probabilities and unification grammar probabilities. A specialized model for the derivation of typed unification grammars is introduced. Different ways of coupling the parser with an LR beam decoder in an online time synchronous fashion are defined and several experimental results are presented. Two top down and one bottom up method are investigated. In bottom up mode, the decoder sends word hypotheses as they are found from left to right, while the parser keeps step. In verify mode, the decoder is always a frame ahead, while the parser verifies received hypotheses, providing language information to the decoder. In predict mode, the parser is a frame ahead, sending possible successor information to the decoder. The latter uses this information to restrict its search space. Finally a method to maximally reduce multiple paths in a left connected word graph produced by a beam

decoder is presented, which can be used in all of the three strategies.

1 Introduction

This article gives an overview of ongoing research in the area of time synchronous processing in speech understanding. Here we are mainly concerned with the design of a chart parser for that issue. The parser which we call a *Left-Right Incremental Active Chart Parser* (LR-ACP) has the following features specific to the task.

- All word hypotheses are processed simultaneously in one chart from left to right.
- The parser uses a typed unification grammar the derivations of which are subject to a probabilistic model.
- Agenda items and edges are supplied with a combined normalized score of acoustic, bigram and grammar model probabilities.
- The parser performs a beam search implemented on the agenda, thereby constraining forward and backward search.
- Extensions of the algorithm for time synchronous interaction with a one pass beam decoder are implemented.
- Receiving the decoder's word ending hypotheses frame by frame, a mapping of hypotheses belonging to the same word copy is performed during parsing.

^{*}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 author.

[†]Published in: L. Boves & A. Nijholt (eds), Proceedings of the 8th Twente Workshop on Language Technology, Dec. 1994, ISSN 0929-0672.

The ideas which led to this work are based, besides others, on works like Shieber 85 [19], Görz 88 [9], Ney 93 [16], Päseler 88 [17], Fujisaki et al. 91 [8]. For a more detailed description see Weber 94 [20]. Some of the work concerning the coupling with the beam decoder is a joint work with Andreas Hauenstein¹ and has already been published in Hauenstein & Weber 94 [10, 11].

2 LR-ACP with Beam-search on the Agenda

The use of active chart parsers and extensions of these in parsing speech has tradition. Our LR-ACP can be seen as a consequent next step to the work of Päseler 88 [17].

2.1 Basic Operations

The basic data structures of an active chart parser can be given as follows:

Definition 2.1 *Basic data structures of an ACP*

Vertices: *A chart is a directed acyclic graph, whose vertices are totally ordered. Vertices correspond to word boundary hypotheses.*

Edges: *An edge consists of a pair of vertices, a dotted rule, a (record of) score(s) and a couple of book-keeping information as there are the covered string of word hypotheses, pointers to daughter edges and also pointers to the left and right hand sides of rule instances, aso.*

Agendas: *For every vertex i there exists an agenda _{i} , keeping triples of actives, inactives and scores, such that the end vertex of the inactive of a triple is vertex i . Agendas are accessed best first.*

We use the standard active chart parser operations, adapted to unification grammars to drive the LR-ACP. For an edge A , we write selectors for the components of A , using $A.from$ and $A.to$ to refer to the begin and end vertices of A , $A.mother$ to the left hand side of A 's rule, $A.next$ to the category to the immediate right of the dot, aso.² We write $\tilde{\cup}$ for the relation of

being unifiable, and \cup for the unification operation.

Definition 2.2 *Basic operations of an ACP*

SEEK-UP: *If an inactive A is inserted, for every rule $\alpha \rightarrow \beta$, such that $A.mother \tilde{\cup} first(\beta)$, insert B , with $B.rule := \alpha \rightarrow \cdot\beta$, $B.from := A.from$, $B.to := A.to$, if B does not already exist.*

SEEK-DOWN: *If an active A is inserted, for every rule $\alpha \rightarrow \beta$, such that $A.next \tilde{\cup} \alpha$, insert B , with $B.rule := \alpha \rightarrow \cdot\beta$, $B.from := A.from$, $B.to := A.to$, if B does not already exist.*

FUNDAMENTAL RULE: *For a pair of active A and inactive B , if $A.next \tilde{\cup} B.mother$, insert C with $C.from := A.from$, $C.to := B.to$, $C.rule := copy(A.rule)$, $C.next := A.next$. Perform the following actions: $\cup copy(B.mother), C.next$. Shift $C.next$ one position.*

PUSH-EDGES: *When edge A is inserted into the chart, then if A is active, push all pairs (A,B) to agenda, such that B is inactive and $A.to = B.from$. Else, push all pairs (B,A) to the agenda, such that B is active and $B.to = A.from$. Insertion is binary according to the score of (A,B) and (B,A) .³*

The LR-ACP can be driven top down using the SEEK DOWN operation as well as bottom up with the SEEK UP operation, as usual active chart parsers can.

2.2 Efficiency Matters

Efficiency of the basic operations is really a challenge, since the search space given by a couple of thousand word hypotheses is huge and in no way comparable to the easy task of parsing deterministic input.

In the specification above, the SEEK DOWN (and SEEK UP) operations cost a lot of unifications, although no top down restriction is propagated during the rule insertion in order to keep the rule instances finite. In order to achieve efficiency with the unification grammar, we preprocessed and hashed all SEEK-DOWN operations besides the redundancy check.

¹University Hamburg, Natural Language Dept.

²We hope to meet the intuitions of the reader, since we do not explain all notational details.

³So the agendas are always kept sorted.

By testing for unifiability only, but inserting the original rule objects, we can keep the instances of empty active edges finite.

In the preprocessing step, we take every daughter of every rule and compute the series of successor rules possibly being inserted by a SEEK-DOWN operation. In this preprocessing, we use only the type skeleton of the grammar and (nondestructive) type unification in a first pass. A local redundancy check ends up recursion.

The series produced is filtered afterwards, using all of the features and propagating them through the series.

All rules and daughters have integer codes assigned, so the table lookup as well as the redundancy check in the chart become cheap.

Our treatment of rules can be seen as a variant of Shieber 85's *Restriction* mechanism [19], based on types.

As a result of this technique, there are no more unifications involved in predictor or scanner operations in Earley's terms. Only during the completer step unifications are actually carried out.

The same technique cannot be used when parsing bottom up. The trigger for a SEEK UP operation are inactive edges, which may have infinitely many extensions of feature structures. Nevertheless, the number of types the feature structure (FS) can have is finite. So we again use the type skeleton of the grammar to do a preprocessing. Anyway, we leave out the filtering step in the preprocessing of the SEEK UP operation, accepting a slight overgeneration of rules to be inserted. These additional edges will fail in the completer operation, consuming a part of those full unifications which would have to be done without a preprocessing anyway.⁴

2.3 Initial Local Agendas and Beams

To achieve an incremental time synchronous behaviour we propose the following control loop, where BEAMWIDTH must be given in advance.

1. Create Vertex V_0 . Insert active start edge. Set $T := 1$.
2. Create V_T with agenda_T . Read all word hypotheses W which end at T , and insert edges for W into the chart.

⁴The preprocessing seems to favorize a top down parsing schema.

3. Set $\text{BEAMVALUE} := \text{score}(\text{top}(\text{agenda}_T)) - \text{BEAMWIDTH}$
4. Apply fundamental rule to $\text{pop}(\text{agenda}_T)$, until $\text{score}(\text{top}(\text{agenda}_T)) \leq \text{BEAMVALUE}$.
5. Save agenda_T .
6. Increment T . Goto 2.

This pseudo code implements a beam search directly on the agenda. There are a couple of reasons for this strategy.

First of all, an agenda_T in a cycle T , right after the insertion of edges due to the reading of new hypotheses consists only of entries involving pairs of actives and word hypotheses. We call the agendas in this state of processing *Initial Agendas*. An *Initial Agenda* encodes the possible future completer operations in form of their initial pair of active edge and word hypothesis.

Since the completer step is the expensive operation in Earley's Algorithm anyway and especially in our handling of unification operations, we prune on agenda items.

Further on, since we use acoustic, n-gram and grammar scores in combination, we need a representation where global scores can be compared or at least, where good estimations of global scores of paths are admissible to apply pruning.

By virtue of the INSIDE and OUTSIDE scoring of edges described in section 3 below, we can assign an optimistic estimate of a path's global score to an active edge not yet covering that path. As a consequence, the maximum score of the initial agenda is the maximum of all possible scores arising from successive completer actions.

In step 5 of the algorithm above, we save those entries of a local agenda which fell below the beam. Since we use a combination of symbolic and probabilistic restriction, we can never guarantee that one of the paths inside the beam will result in a valid analysis of some spanning sequence of word hypotheses. In order to make the system robust, a second search phase can be initiated, parsing the global agenda of pruned entries when the time synchronous search failed. When there is a spanning analysis, it will be discovered then.

3 Metrics

This section is concerned with the combination of probabilities coming from different models

and the partition of combined scores into an INSIDE score and an OUTSIDE score.⁵

3.1 Combining Scores

Different models used in one system tend to have different numbers of parameters and to be trained on different data. So, taken seriously, they cannot be combined.

In practice we overcome this problem by adjusting the magnitude of the scores involved. This is done by introducing weights relating of the scores coming from different models. We use log probabilities, since the overall search is Viterbi like, maximizing rather than summing up.

$$CS(W,D) = AS(W) + \gamma NGS(W) + \delta GS(D) \quad (1)$$

Acoustic score (AS), bi-gram score (NGS) and grammar score (GS) are just added, γ and δ regulating the relative weights. W stands for an utterance and D for a possibly partial grammatical derivation of it.

The single scores themselves are normalized by the number of operations in order to be able to compare word hypotheses of different length, utterances made up of a different number of words and trees made up of different amounts of grammatical operations.⁶

$$AS(W) = \frac{\log P(W|s_{i,j}, \text{hmm})}{j-i} \quad (2)$$

$$NGS(W) = \frac{\log P(W|\text{bi-gram})}{\text{words}(W) - 1} \quad (3)$$

$$GS(D) = \frac{\log P(D|\text{gm})}{\text{rules}(D) + \text{typeshifts}(D)} \quad (4)$$

The acoustic score for an utterance W is given by the probability assigned by the acoustic model to the signal from frame i to frame j, the portion spanned by W, divided by the number of frames spanned. Analogously we define the bi-gram score and the grammar score, the latter being normalized by the number of rule applications and type shifts, the items of observation of the grammar model *gm* (which will become more transparent in section 4).

⁵Although these two scores correspond more or less to Päseler's IS and AS ([17]), we take the names from Baker's [2] Inside-Outside-Algorithm, which we think was the first idea in that direction.

⁶We write X(Y) for the number of operations of type X used to produce Y.

This is not the only possible way to combine scores. Another method, first summing weighted probabilities, and normalizing them afterwards by the sum of all operations did not show improvements. The behavior of the system is more sensitive to the choice of values for the weights. It turned out that it was really difficult to find good settings by hand. While it was comparably easy to tune only one weight⁷, using only acoustic and n-gram scores in the system, things get more complex when additional models are involved. Although we did not yet test it, we feel that the weights should be subject to an optimizing hill climbing procedure.

For the counters for operations and the absolute log probabilities we write AO, NGO, GO, AP, NGP and GP respectively. Edges in fact carry a record with fields for these single values, since the log probabilities and operations are kept separate for edges. This makes it easier to compute new values for new edges. Combined scores are only computed for comparisons which happen to be done only on agenda items.

3.2 Inside and Outside Scores

Similar to Päseler 88 [17], we give two different scores to edges. The first one is the INSIDE score of an edge — the score coming from the portion spanned by that edge. Roughly speaking the OUTSIDE score is the cumulated score of those edges in a left context which were leading to the introduction of a certain edge.

In top down mode the OUTSIDE score of an edge i is the best score of some edges j to m, where j to m lead to the introduction of edge i plus the INSIDE score of i.⁸ Generally the edges j to m leading to an edge i are spanning the portion from frame 0 to the beginning of i.

In bottom up mode, where empty actives are introduced on the basis of inactive edges with the SEEK-UP operation, INSIDE and OUTSIDE score of an edge fall together. There is no left context for an edge to be determined from which we could say it led to the introduction of that edge.⁹ So the following is on the building of the OUTSIDE score in the case of top down parsing.

⁷The experiments in Hauenstein & Weber 1994 were done only with acoustic and n-gram scores.

⁸..supplied with the transition penalties between m and j.

⁹This is not really true, since we could use top down filtering (as described by Wirén [21]) during bottom up parsing.

We define the INSIDE score as well as the OUTSIDE score for the start edge directly below, without referring to the single components due to space limitations.

Definition 3.1 *INSIDE score for an edge E , from i to j , spanning string W with analysis D_W*

$$IS(E_{i,j,W,D_W}) = \begin{cases} CS(W, D_W) & i < j \\ \delta GS(E.rule) & i = j \end{cases} \quad (5)$$

Describing the OUTSIDE scores for the initial start edge is easy. The start edge is initialized with a zero value for all parts of the OUTSIDE score. New empty actives inherit the acoustic and n-gram scores from the introducing active. Since we introduce a new grammar rule in an empty active edge, the grammar score has to be updated accordingly. This mechanism ensures that an outside score is always maximal: Since we process the agenda above the beam in a best first fashion, it is guaranteed that the first SEEK DOWN operation leading to a certain empty active is the one with the best score. OS refers to the combined score, OS.X to the single component X of that score accordingly.¹⁰

Definition 3.2 *OUTSIDE score for the start edge*

$$OS(E_{0,0,\emptyset,Startgraph}) = 0 \quad (6)$$

Definition 3.3 *OUTSIDE score for empty actives resulting from a SEEK DOWN operation.*

$$\begin{aligned} OS.X(E_{j,j}) &= OS.X(E_{i,j}) \\ OS.GO(E_{j,j}) &= OS.GO(E_{i,j}) + 2 \\ OS.GP(E_{j,j}) &= OS.GP(E_{i,j}) + \\ &\log P(type(E_{j,j}.mother)|type(E_{i,j}.next), gm) + \\ &\log P(E_{j,j}.rule|type(E_{j,j}.next), gm) \end{aligned}$$

$E_{j,j}$ is introduced by $E_{i,j}$,
and X being AO, AP, NGO, NGP

(7)

Do define the OUTSIDE score which is given to a resulting edge by an application of the fundamental rule to its daughter active and inactive, we must refer to AO, NGO, GO, AP, NGP GP, and the strings covered by edges. In record access¹¹ notation: A.intro ist the string of word hypotheses from vertex 0 through the lattice, on which an OUTSIDE score of A ist based on. A.string ist the string of word hypotheses actually covered by A.

¹⁰ Again in record notation.

¹¹ which is really the way we implemented it.

Definition 3.4 *OUTSIDE score of an edge C , resulting from an active A and an inactive B by an application of the fundamental rule.*

$$\begin{aligned} OS.AO(C) &= OS.AO(A) + IS.AO(B) \\ OS.AP(C) &= OS.AP(A) + IS.AP(B) \end{aligned}$$

$$\begin{aligned} OS.NGO(C) &= OS.NGO(A) + IS.NGO(B) + 1 \\ OS.NGP(C) &= OS.NGP(A) + IS.NGP(B) + \\ &\log P(last(A.intro)|first(B.string), n-gram) \end{aligned}$$

$$\begin{aligned} OS.GO(C) &= OS.GO(A) + IS.GO(B) + 1 \\ OS.GP(C) &= OS.GP(A) + IS.GP(B) + \\ &\log P(type(B.mother)|type(A.next), gm) \end{aligned} \quad (8)$$

$$OS(C) = \frac{OS.AP(C)}{OS.AO(C)} + \gamma \frac{OS.NGP(C)}{OS.NGO(C)} + \delta \frac{OS.GP(C)}{OS.GO(C)} \quad (9)$$

When we combine two edges we compute new values for all the single components of the INSIDE and OUTSIDE score.

4 Probabilistic Typed Unification Grammars

We wanted our parser not only to help the decoder to find the best word sequence. What was intended originally was a mapping from a signal to a formal representation of a meaning. So, since the typical grammar produces a lot of derivations for a given string, we had to add some disambiguation device in order to choose one of those multiple analyses. Secondly, we did not intend to compute all of the derivations but rather prune those which were not intended early.

The straight way was to have a probabilistic model of the derivations of our unification grammar. The scores can be combined with the other scores and the general beam search will have effect on the completer operations coming from different analyses of the same word string.

Besides the work on PCFGs by Baker [2], Jelinek [13], Fujisaki [8], and others there is a handfull of publications on probabilistic versions of unification grammars, like Hemphill & Picone 89¹² [12], Briscoe & Carroll 93¹³ [3] or Magerman & Marcus 91 [14]¹⁴.

¹²Don't read it.

¹³Read it.

¹⁴The work on Pearl is rather on replacing a unification grammar, but nevertheless on this topic.

All this work is on unification grammars that do not use typed feature structures. So the main thing is to identify finitely many classes of feature structures in order to apply PCFG methods to the observation of derivations of the unification grammar. Mapping of infinitely many FS to a set of classes is usually done by creating a set of *restrictions* in the sense of Shieber 85 [19], which do not unify with each other. Observing the classes in a derivation instead of the original FS leads to the possibility to decide to which class a left hand side of a rule belongs. So we can train some n-gram models on right and left hand sides of rules.

In a typed unification grammar like ours, we use a type system with *appropriateness* as defined in Carpenter 91 [4]. In such a system, we have a finite set of type names associated with well formedness conditions on FS. Using the types of FS as class names, we do not face the same problem as we do using unification grammars without types.

On the other hand, a type is not a fixed label as eg in CFG. Simple types are usually defined in an IS-A hierarchy which determines the unifiability and subsumption relations of types. Complex types arise from unification of types where two types unified have several common subtypes.

Assume the unification of two typed FS as in (10):

$$A[f1 : v1] \cup B[f2 : v2] = C[f1 : v1, f2 : v2] \quad (10)$$

The type C can be equal to A or B or be a common subtype of both.

In our grammar rules global types of FS encode much linguistic information and a lot of linguistic relations are encoded in the type hierarchy instead of grammar rules. An example is the instantiation of the so called *Vorfeld* in german sentences. The following simplified example shows the method.

$$S2_intrans[.] \rightarrow Vorfeld[.] V_intrans[.] \quad (11)$$

$$Vorfeld > NP PN Perspron S_adv... \quad (12)$$

Having a number of rules like the one in (11) in our typed unification grammar, the type declaration in (12) specifies, FS of which type can instantiate with FS of type *Vorfeld*. The former are just stated to be subtypes of the latter.

Classifying FS by type names and using them directly for a PCFG style probabilistic model leads to trouble. We cannot guarantee to always have $\sum_{\beta} P(\beta|\alpha) = 1$ for rule schemata $\alpha \rightarrow \beta$. since we do not know what α 's actual instances will be. Compiling out the types and multiplying the rules accordingly would be possible, since in our system we only have a finite number of types – the power set of all simple types. On the one hand this leads to a huge set of rule instances, on the other hand there are systems like the one of Emele & Zajac 90 [7], where infinitely many types may occur. So the method is not general.¹⁵

So what we do in order to get a proper statistics on derivations on the typed FS rules is to decompose a rule application and a unification of two typed FS into two parts: The shift of the types being unified and the application of the rule itself. When for instance, in a sentential form of a left derivation a FS of type *Vorfeld* is unified with a FS of type *NP*, we have two observations, the shift of types (*NP*, *Vorfeld*) and the application of a rule with original lhs type *NP*. In fact we do not care about the result type of the type unification which might be a complex type.¹⁶

Definition 4.1 Given a left derivation in a typed unification grammar of the form:

$$S[.] \xRightarrow{*} x\alpha'y \Rightarrow x\beta'y \xRightarrow{*} w \quad (13)$$

where α' has been generated as an instance of α'' a member of the right hand side of some rule, and β' has been generated by application of a rule $\alpha \rightarrow \beta$ by unification of α and α' , we call the pair $type(\alpha''), type(\alpha)$ an **observed type shift**.

In our model *gm* for the derivation of the typed unification grammar we describe each rule application – in other words: step in a left derivation – as a pair of a type shift and original rule of the grammar.

Implicitly this is adding some unary rules for all type shifts and guaranteeing that in every derivation type shifts and rules are applied alternatingly.

Definition 4.2 A probabilistic typed unification grammar is:

¹⁵It is ugly, which is the main argument.

¹⁶We could have done this, but the amount of parameters would have increased by $O(2^n)$, where n is the number of simple types.

Type hierarchy: *A lattice of types with a top and bottom element which defines the subsumption relations of types.*

Lexicon: *We assume the lexicon to be a set of unary productions, where the right hand side of the production is a word string.*

N-ary grammar rules: *The grammar rules consisting of one typed feature structure as a left hand side and a sequence of typed feature structures as a right hand side.*

Model gm: *Assigning a probability to all pairs of two types and to all pairs of type and rule.*

The model gm is organized as two bi-grams, one for the type shifts and one for the rules. The type shifts and rules are thought to be independent. So the following relation holds:

$$\sum_B \sum_{\beta} (A \xrightarrow{\text{tsh}} B, B[.] \xrightarrow{\text{rule}} \beta | gm) = 1 \quad (14)$$

During training, we treated the type shifts as if they were unary productions. We parsed a corpus with the original typed unification grammar. Parses were represented as lists of the type shifts and numbers of rules that were involved. We used a variant of the unsupervised training method of Fujisaki et al. 91 [8] to estimate the models.¹⁷

During parsing we can distinguish three types of type shifts.

- Those pairs of types which are not unifiable are type shifts of probability 0. Thus, we can prune an impossible analysis on grounds of gm instead of performing a unification that will fail anyway.
- Possible but not observed type shifts should receive a smoothed value as usual in standard bi-gram techniques.
- Observed type shifts receive their trained probabilities.

By our method of having an additional bi-gram of type shifts, we achieve a couple of pleasing properties. The first one is, that we can

¹⁷We do not object to supervised learning. We just did not have a tree bank.

have a correct probabilistic model of a unification grammar which relates typed feature structures with each other. Furthermore, by using the types as model classes, we can capture a lot of information of the unification grammar in our model, since types are tied to the feature structures by *appropriateness* checking as described in Carpenter 91 [4] and the subsumption relations of types in the hierarchy are captured as well. The method generalizes to the powerful type systems in the style of Emele & Zajac 90 [7] since only pairs of simple types¹⁸ are used in the model. Since in type unification the resulting type is determined by these two types we do not lose any information.

Finally, if for a given grammar the hierarchy of types is flat, the model works like a real PCFG backbone for a unification grammar, as it is used for example by the TINA parser [18]. In that case, all types are unifiable only with themselves and with the top element of the type lattice. When no rules of the grammar use top as a global type of a FS, we will arrive at a collection of observed type shifts all of which have the form (X,X) with probability 1.

5 Tightly Coupling the Parser with a Beam Decoder

Some extensions of the LR-ACP allow for different time synchronous couplings with a beam decoder for word hypotheses. So far we investigated three modes, namely time synchronous parallel bottom up (*BUI*), time synchronous bottom up with top down verification (*BUITDV*) and time synchronous top down predicting mode (*TDPI*). Some results of these couplings driving the parser with an n-gram only have already been presented in Hauenstein & Weber 94 [10, 11]. The decoder is a viterbi one pass beam decoder¹⁹ extended for the different protocols as explained below.

¹⁸In fact complex types may also be used. The bound comes from the type names occurring in the original grammar rules before any grammatical operation took place.

¹⁹.. developed in the VERBMOBIL TP 15 project, by Andreas Hauenstein, University of Hamburg.

5.1 BUI

In BUI we run the decoder and the parser concurrently. In every frame processed by the decoder, word ending hypotheses above the decoders beam are immediately sent to the parser. The decoder does not use language model²⁰. The hypotheses are quadruples (*from, to, key, score*), where *key* is the name of the wordform and *score* is the acoustic score assigned to the frames *from* to *to* by a word copy of the model for *key*.

The LR-ACP works exactly as described in section 2.3.

The BUI coupling uses no feedback messages from the parser to the decoder, so real parallelism is possible.

5.2 BUITDV

In BUITDV mode the decoder uses top down verifications of the parser as a language model. Since for one word copy a language penalty has to be added only once, a fifth field is added to the bottom up word hypotheses as a flag signalling to the parser whether a hypothesis is new²¹ or has already been verified. The whole procedure works as follows.

Definition 5.1 *An example cycle of BUITDV for frame I:*

1. The decoder finds $m+n$ new word ending hypotheses at frame I . n had already been found at $I-1$, m are new. All of them are sent to the parser in cycle I with the flags set accordingly.
2. The parser takes all $m+n$ word hypotheses and performs PUSH-EDGES resulting in an initial agenda $_I$.
3. The parser processes agenda $_I$. The first successful application of the fundamental rule to a pair of edges involving a new word hypothesis leads to a verification of that hypothesis.
4. When the beam of agenda $_I$ is reached, the rest of agenda $_I$ is searched for word hypotheses not yet verified. The first one found leads to a verification of that hypothesis.

²⁰But a fixed transition penalty in order to prevent an inflation of short words.

²¹Then this word copy's final state has been above the decoder's beam for the first time

5. The decoder receives verification messages. All those new hypotheses in I , which were not verified are set to a zero probability. All other new hypotheses have the verified language penalty added.

Verification messages are built by the parser using agenda items. They are quintuples (*from, to, key, score, flag*) as the associated bottom up hypotheses are. *from, to* and *key* identify the decoder's active word copy which was to be verified. The *flag* field is supplied with the predecessor word hypothesis' string, which led to the best score inside the parser's search. For *score* a couple of options are given.

Definition 5.2 *Transition score with bi-gram and gm supplied by a verification message for a pair (A, W). of edges on the agenda.*

$$\begin{aligned} \text{score} = & \\ & \gamma * \log P(\text{first}(W.\text{string})|\text{last}(A.\text{intro}), n\text{-gram}) + \\ & \delta * \log P(\text{type}(W.\text{mother})|\text{type}(A.\text{next}), gm) + \\ & \delta * \log P(W.\text{rule}|\text{type}(W.\text{mother}), gm) \end{aligned} \quad (15)$$

Definition 5.3 *Transition score with bi-gram supplied by a verification message for a pair (A, W). of edges on the agenda.*

$$\begin{aligned} \text{score} = & \\ & \gamma * \log P(\text{first}(W.\text{string})|\text{last}(A.\text{intro}), n\text{-gram}) \end{aligned} \quad (16)$$

The score which is used as a transition penalty between words by the decoder can consist of a bi-gram score and a score given by the grammar model for those grammar operations, which are used to incorporate the word hypothesis into an existing partial analysis. The latter consist always of the score for the lexical access²² and the type shift which takes place when the lexical entry is combined with some rule actually processed by some active edge. This is the part of the grammar operations which can directly be related to a local transition between words. Again both scores are weighted, hence not normalized. Alternatively, only the bi-gram score can be used as in common decoding.

Maximization is not local here. Since predecessor word and analysis are selected on the base of agenda items, the criterion for the maximization on predecessors is global.

²²treated as an unary rule, namely W.rule in def. 5.2.

One difficulty we have to face is that the decoder will have access to transition penalties when a word model runs into a final state and not at the beginning of a word. We can overcome this problem with a method inspired by Aubert’s et al. 1994 [1] handling of tree lexica in one pass decoding. For both the n-gram transition and the part of the grammar score used above there exists a stable maximum for a word with respect to all possible immediate predecessor words.

So starting a word copy in the decoder we can use this maximum as a transition penalty adding the difference with respect to the maximum when the copy has been verified.

5.3 TDPI

TDPI is the opposite method of coupling to BUITDV in terms of control, since the parser predicts possible successor word hypotheses starting in a given frame while the decoder selects among those.

The basic procedure in TDPI goes as follows:

Definition 5.4 *An example cycle of TDPI for frame I:*

1. *The decoder has sent all word ending hypotheses at frame I.*
2. *Having parsed all word hypotheses ending in vertex I, the parser takes all the active edges in vertex I and calculates a set of predicted predecessor words. These are sent to the decoder supplied with transition scores as a prediction for I.*
3. *The decoder in frame I starts only those new word copies which occur inside the prediction. Newly started word copies are initialized with the supplied transition penalty.*

A grammar based calculation of predictions is too expensive since a full backward search (completer step) would have to be executed for each candidate. So we propose a generate and test algorithm for the computation of a set of predictions, based on the chart up to vertex I.²³

Definition 5.5 *Computing predictions*

1. *Let ACTIVE be the nonempty active edges ending in vertex I, $P_I = \emptyset$*

²³A broad discussion of efficiency in chart based computation of predictions can be found in Weber 94 [20].

2. *Take the n best successors s of any $w = last(A.intro)$, $A \in ACTIVE$, maximizing $logP(s|w, n\text{-gram})$.*
3. *Create vertex_{-I} and insert inactive edges for all s into vertex_I, vertex_{-I}, thus filling agenda_{-I}.*
4. *Until below_beam(agenda_{-I}), let (A,S) be pop(agenda_{-I}):*
When combine(A,S) \neq fail, remove all pairs (X,S) from agenda_{-I} and set $P_I = P_I \cup (I, \theta, S, last(A.intro), score)$.

The score of a quintuple (from, to, predicted word, predecessor, score) in P_I is the same as for the verification messages given in 5.2. Again, the actual score returned is globally selected. The reason for the n best technique in step 2 is, that we can precompute this step and hash the results. So the complexity of the whole calculation is determined by n times the number of active edges ending in vertex I in the worst case. We do not use a completer step to test the predictions, since one success with some active edge is sufficient to show that there is at least one spanning analysis according to the type skeleton of the grammar. Adding all the features to the test would be prohibitive. If we use a class based n-gram model where the classes correspond with types of our grammar we could reduce n considerably.²⁴

6 Incremental Time Mapping using Edge Inheritance

The set of word hypotheses transferred bottom up from the decoder to the parser in a time synchronous coupling constitutes a *left connected word graph* seen a posteriori. In a non incremental architecture where word recognition ends before parsing starts, we can delete all dead ends in a word graph using an offline backward search phase through the set of word hypotheses. An abstracting step from frames to vertices can omit superfluous paths in the graph consisting of word ending hypotheses differing only in one frame. Chien et al. 90, 93 [6, 5] present such a mapping from frames to vertices.²⁵

²⁴We did not try that yet.

²⁵Their method is a bit cryptical, but the only reference we found in the literature.

In our incremental architecture this does not work. When the decoder finds a word ending hypothesis we never know whether the word copy in question will still be above the beam in the next frame and if yes, whether its normalized acoustic score will increase or decrease. This leads to the effect, that sometimes the parser receives exactly the same set of word hypotheses in two successor frames, differing only with respect to their end vertex. The same completer actions are carried out two times in such a case. In order to implement a normalization without any lookahead, we developed a method we call incremental time mapping, which consists of a slight modification of the LR-ACP where some book-keeping is used to do all completer actions only once which stem from the same decoder's word copy. The basic idea is to inherit all those edges from vertex I to vertex $I+1$ which resulted from a word hypothesis in vertex I for which an identical one has later been found with ending time $I+1$. All word hypotheses found for the first time are parsed as usual.

For that issue, we add a little non monotonicism to the chart parser:

- One edge can belong to a number of vertices. Seen from the vertices the edge is shared. Seen from the edges, an edge now possesses a list of end vertices the head of which is the actual one.
- We allow for the replacement of edges by others. This mechanism is only used on empty active edges. It is guaranteed that no inconsistencies arise, since we replace only edges ending in vertex I in cycle I .

The basic modified control loop for the LR-ACP can be paraphrased as follows.²⁶

Definition 6.1 *Description of a cycle $_I$ in an LR-ACP with edge inheritance.*

READ HYPOS :

KNOWN, NEW, INH_EMPTIES be \emptyset .

Read in all hypotheses ending at I . Add those which have a predecessor at $I-1$ differing only in ending time to KNOWN.

Let $\text{pred}(\text{KNOWN})$ be the corresponding hypotheses in vertex $_{I-1}$. Add all others to NEW.

²⁶A more precise description can be found in Weber 94 [20].

INHERITANCE :

For all edges A ending in vertex $_{I-1}$, A coming from some $w \in \text{pred}(\text{KNOWN})$:

When $\text{score}(w) < \text{score}(w)'$ in KNOWN, update the acoustic score and operations of A .

If empty_active(A), add A to INH_EMPTIES

Else, add A to vertex $_I$.edge-in, add vertex $_I$ to A .to. When A is active, perform a SEEK-DOWN on A in vertex $_I$.

For all the pairs (A,B) in save_agenda $_{I-1}$, B coming from some $w \in \text{pred}(\text{KNOWN})$, push (A,B) to agenda $_I$.

PARSE :

Insert all word hypotheses in NEW into the chart. $\text{MAXVALUE} := \max(\max(\text{OS}(A) \setminus A \in \text{vertex}_I.\text{edge-in}), \text{score}(\text{top}(\text{agenda}_I)))$

$\text{BEAMVALUE} := \text{MAXVALUE} - \text{BEAMWIDTH}$

Apply fundamental rule to $\text{pop}(\text{agenda}_I)$ until $\text{score}(\text{top}(\text{agenda}_I)) \leq \text{BEAMVALUE}$.

Save agenda $_I$.

UPDATE EMPTIES :

For all A in INH_EMPTIES: If there exists no empty edge B in vertex $_I$, with $A.\text{rule} = B.\text{rule}$, add $\text{copy}(A)$ to vertex $_I$. Else, if $B.\text{OS} < A.\text{OS}$, replace B by $\text{copy}(A)$.

In order to keep the beam search mechanism working as in the original LR-ACP we have to take care of the inherited edges, when the BEAM is calculated. Furthermore, since the scores of edges inherited may be subject to an updating and since the new maximum in cycle I may differ from the maximum in cycle $I-1$ those pairs with inherited edges, which had been pruned in cycle $I-1$ must be pushed on the new agenda.

Nevertheless, scoring is different from the original version, since we only update an edges acoustic score until a maximum is reached.

In the original version, a sequence of say 5 word hypotheses coming from the same decoder's word copy was represented by 5 edges²⁷. One of those had the maximal acoustic score. Using incremental time mapping, we represent the 5 word hypotheses by only one edge, having 5 possible updates on the acoustic scores. In order to keep the global Viterbi search correct, we stop updating when the maximum is

²⁷Or $5n$ edges, when n lexical entries were found.

reached, so the whole sequence of original hypotheses will be represented by its maximum. The mechanism is similar to the pruning used in Ney’s 91 [15] dynamic programming parsing based on a CYK parser, but constitutes an incremental time synchronous version of the latter.

While the algorithm above works well for BUI and TDPI, in BUITDV some slight modification have to be made. In BUITDV the decoder sends all those hypotheses first which are to be verified, waiting for response, and sends all known hypotheses later. So the decoder can prune based on verified hypotheses only but the parser cannot use the maximum of all scores in the first phase, parsing the new ones. A version of the parser with time mapping using two distinct steps for the new and known hypotheses worked well in experiments although the first step did not have certain knowledge about the maximum in every cycle.

The calculation of predictions in TDPI can be subject to the inheritance technique too. All predictions in a cycle I will be pointed to by the edges they are based on. When the relevant edges are inherited, the predictions coming from them will be inherited with them then.²⁸

7 Selected Experiments

In this section we present a couple of selected experiments with the LR-ACP. The n-gram model used in all of the experiments is word based and has a test set perplexity of 54.28. We used two unification grammars for our experiments. A small and rather restrictive one (GRS) was used in the experiments without the probabilistic grammar model. A larger, less restrictive grammar (GRB) with more structure building features was used with the model *gm* in order to have a more realistic test bed. A corpus of 200 sentences of the train information domain was used for training. Tests were carried out on 10 sentences with the small grammar and 5 sentences with the big one. The acoustic models used were well adapted and led to an acoustic word accuracy of 88.6 on the test sentences.

The experiments should be regarded with care. They are of a small scope they are preliminary in nature. Further testing of the architecture and its variants on a new domain with more realistic grammar and bi-gram are currently at work.

²⁸ Again, details can be found in Weber 94 [20].

The two tables below present the results of Hauenstein & Weber 94 [10, 11] for decoding alone (AC), BUI (BU), BUITDV (BV) and TDPI (TP), using GRS with the bigram only. We measured cpu time (T)²⁹, bottom up hypotheses (BUH), top down hypotheses (TDH), sentence recognition (SR), edges used (E) and maximal active gridpoints (GP).

The first table shows results for narrow beams, the second for wide beams.

	T	BUH	TDH	SR	E	GP
AC	47.4	–	–	0.5	–	790
BU	96.9	385	–	0.7	6195	790
BV	58.2	289	41	0.5	5827	1045
TP	220.5	223	7621	0.7	4954	78

	T	BUH	TDH	SR	E	GP
BU	2911.7	1465	–	0.7	19769	1712
BV	115.8	776	135	0.7	9765	2047
TP	282.7	415	8304	1.0	7519	184

The effect of the top down strategies cannot be overseen here. The amount of bottom up hypotheses is heavily reduced by the restriction supplied to the decoder.³⁰ For the relatively well adapted hmms, the TDPI strategy leads to a sentence recognition of 100 percent using wide beams. BUITDV is superior to BUI only in terms of efficiency.

The effect of the time mapping can be seen in the following table, which changes the overall impression given by the first results. All parameters besides the time mapping are as those used in the wide beam case. The table shows BUI, BUITDV and TDPI with incremental time mapping. TDPI has been tested with standard time mapping only (TM) and with time mapping for the computation of predictions (TMP). The comparison with time mapping has been done on five of the test sentences, which were well recognized by all the couplings.

	Time in Sek.	Edges
BUI, TM	68.8	6613
BUITDV, TM	51.4	4662
TDPI, TM	172.6	5004
TDPI, TM & TMP	58.2	4226

²⁹ Allegro Common Lisp on a SUN Sparc10, parsing time.

³⁰ BUI without time mapping often exhausted the machines 48mb RAM causing swapping which explains the huge increase in cpu time.

We can see, that in terms of efficiency the strategies do not lead to similar results, when redundant completer operations are omitted. The TDPI case with standard time mapping gives a good impression of the enormous overhead caused by the calculation of predictions.

The first results with a larger grammar (GRB) and the model gm are presented below. The first column shows the weights for *n-gram* and *gm*. The single weights were optimized on the test sentences by hand, maximizing sentence recognition rate. The rows show *gm* only vs *n-gram* only vs a combination of both models. PR and TR stand for pruned and tried agenda items. T, SR and E are defined as above. The table shows a version of GRB without rules for performance phenomena and a BUI coupling.

ng/gm	PR	TR	T	SR	E
0.0 0.4	19920	2075	103	4/5	7774
0.2 0.0	7031	25796	137	3/5	8874
0.1 0.2	26381	2486	108	4/5	8192

The full GRB grammar and TDPI coupling with n-gram value supplied to the decoder with the prediction, led to the following results:

ng/gm	PR	TR	T	SR	E
0.0 0.4	69052	530	145	5/5	6627
0.2 0.0	22588	5343	171	5/5	6648
0.1 0.2	69858	599	157	5/5	6646

The most prominent observation is, that when the influence of *gm* is set to zero, much more agenda items are tried. The reason for this is, that on the basis of *gm* a lot of unifications can be pruned, that would fail with a type mismatch if tried. So these agenda items do not lead to additional edges. They are responsible for a growth of processing time, since the processing of the scores is cheaper than unification. Astonishing there is a slight increase in processing time and edges from *gm* alone to the combination of the models. We had expected the opposite effect. The weights, which were optimized by hand, might be the reason for that. A better tuning of the weights by an optimizing procedure, taking smaller steps than our hand tuning, should correct this.

8 Conclusions

We presented an active chart parser which combines different statistical knowledge sources in

parsing speech in a time synchronous fashion. The beam search implemented on the agenda takes global scores into account.

On the basis of this parser, different couplings with a decoder in the style of Hauenstein & Weber 94 were explicated. The time mapping technique presented here leads to the effect, that differences in efficiency between the couplings observed by Hauenstein & Weber 94 become small.

TDPI is superior to the other couplings with respect to recognition rate in the experiments performed. This does still hold, when a statistical model of the unification grammar's derivations is included.

We introduced a simple method to supply a typed unification grammar with a statistical model of its derivations by observing type shifts on the one hand and type-rule pairs on the other hand. The method generalizes to most known type systems used for such applications.

Further experiments, larger in scale have to be performed in the future.

References

- [1] X. Aubert, C. Dugast, H. Ney, and V. Steinbiss. Large vocabulary continuous speech recognition of wall street journal data. In *ICASSP*, 1994.
- [2] J.K. Baker. Trainable grammars for speech recognition. In *Speech Communication Paper, 97th Meetingh of the Acoustical Society of America*, Cambridge, Mass., 1979. MIT Press.
- [3] Ted Briscoe and John Carroll. Generalized probabilistic lr parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59, 1993.
- [4] Bob Carpenter. *Typed Feature Structures: Inheritance (In)equality and Extensionality*. CMU, 1991.
- [5] Lee-Feng Chien, Keh-Jiann Chen, and Lin-Shan Lee. A best-first language processing model integrating the unification grammar and markov language model for speech recognition applications. In *IEEE Transactions on Speech and Audio Processing*, volume 1,2, pages 221–240, 1993.

- [6] Lee-Feng Chien, K.J. Chen, and Lin-Shan Lee. An augmented chart data structure with efficient word lattice parsing scheme in speech recognition applications. In *Proceedings of COLING*, pages 60–65, 1990.
- [7] Martin Emele and Rémi Zajac. A fixed-point semantics for feature type systems. In *Proc. of the 2nd International Workshop on Conditional and Typed Rewriting Systems (CTRS)*, Montreal, June 1990.
- [8] T. Fujisaki, F. Jelinek, J. Cocke, E. Black, and T. Nishino. A probabilistic parsing method for sentence disambiguation. In Masura Tomita, editor, *Current Issues in Parsing Technology*, pages 139–148, Norvell, Mass., 1991. Kluwer Academic Publishers.
- [9] Günther Görz. *Strukturanalyse natürlicher Sprache*. Addison Wesley, Bonn, 1988.
- [10] A. Hauenstein and H. Weber. An investigation of tightly coupled time synchronous speech language interfaces using a unification grammar. In Paul McKeivitt, editor, *Proceedings of the Workshop on Integration of Natural Language and Speech Processing at AAAI 94*, pages 42–49, Seattle, August 1994.
- [11] A. Hauenstein and H. Weber. An investigation of tightly coupled time synchronous speech language interfaces. In *Proceedings of the CONVENTS 94*, Wien, September 1994.
- [12] Charles Hemphill and Joseph Picone. Chart parsing of stochastic spoken language models. In *DARPA Workshop on Speech and Natural Language*, Philadelphia, Pennsylvania, February 1989. DARPA.
- [13] F. Jelinek. Basic methods of probabilistic context free grammars. In P. Laface and R. De Mori, editors, *Speech Recognition and Understanding: Recent Advances*, volume NATO ASI Series, F 75, pages 345–360, Berlin Heidelberg, 1992. Springer Verlag.
- [14] D.M. Magermann and M.P. Marcus. Pearl: A probabilistic chart parser. In *Proc. of the European ACL*, March 1991.
- [15] Hermann Ney. Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Transactions on Signal Processing*, 39(2):336–340, February 1991.
- [16] Hermann Ney. *Architecture and Search Strategies for Large-Vocabulary Continuous-Speech Recognition.*, pages 59–84. (NAT)-ASI Bubi3n, 1993.
- [17] Annedore Paeseler. Modification of earleys algorithm for speech recognition. *NATO ASI Series: Recent Advances in Speech Understanding*, F46:465–472, 1988.
- [18] S. Seneff. Tina: A probabilistic syntactic parser for speech understanding systems. In *DARPA Speech and Natural Language Workshop*, Philadelphia, February 1989.
- [19] Stuart M. Shieber. Using restriction to extend parsing algorithms for complex-feature-based formalisms. In *Proc. of the ACL 1985*, volume 23, pages 145–152, 1985.
- [20] Hans H. Weber. *LR-inkrementelles probabilistisches Chartparsing von Worthypothesenmengen mit Unifikationsgrammatiken: Eine enge Kopplung von Suche und Analyse*. PhD thesis, Submitted to Universitat Hamburg, FB Informatik, Dezember 1994.
- [21] M. Wir3n. *Studies in Incremental Natural-Language Analysis*. Number Dissertation No. 292 in Link3ping Studies in Science and Technology. Link3ping University, 1992.