Gefördert vom
bmb+f
Bundesministerium für
Bildung, Wissenschaft,
Forschung und Technologie

Verbmobil
Verbundvorhaben

Semantic Construction from Parse Forests

Michael Schiehlen

Universität Stuttgart

Vm
Bericht 145
Juli 1996

Juli 1996

Michael Schiehlen

Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Azenbergstraße 12
D – 70174 Stuttgart

Tel.: (0711) 121 - 1363
Fax: (0711) 121 - 1366
e-mail: `mike@ims.uni-stuttgart.de`

Gehört zum Antragsabschnitt: 8.4 Flache Analyse

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

Semantic Construction from Parse Forests

Michael Schiehlen[1]

Institute for Computational Linguistics, University of Stuttgart,

Azenbergstr. 12, 70174 Stuttgart
mike@adler.ims.uni-stuttgart.de

Abstract

The paper describes a system which uses packed parser output directly to build semantic representations. More specifically, the system takes as input Packed Shared Forests in the sense of Tomita [Tomita, 1985] and produces packed Underspecified Discourse Representation Structures. The algorithm visits every node in the Parse Forest only a bounded number of times, so that a significant increase in efficiency is registered for ambiguous sentences.

1 Introduction

One of the most interesting problems comes about by the tendency of natural language discourse to be ambiguous and open to a wide variety of interpretations. Generating representations for all the interpretations is not feasible in view of the strict computational bounds imposed on NLP systems. Instead, two other routes have been pursued: (1) the integration of further disambiguating knowledge and heuristics into the system or (2) the generation of a single semantic representation that summarizes all the interpretations in the hope that the application task will force a distinction between the interpretations only in few cases. Such a summary representation is called *underspecified* if a procedure is given with it to derive a set of real semantic representations from it. By now, several techniques are known to underspecify quantifier scope ambiguities [Alshawi, 1992], [Reyle, 1993]. In this paper Discourse Representation Structures [Kamp and Reyle, 1993] are employed as underlying semantic representations. For underspecification with respect to scope ambiguities the present approach makes use of Underspecified Discourse Representation Theory [Reyle, 1993]. Another strand of research has looked at compact representations for parse outputs [Earley, 1970], [Tomita, 1985] and efficient parsing algorithms to deliver such representations. Unfortunately, advances made in this area did not have impact on semantic construction. It was still necessary to first unpack the compact parsing representation and derive the individual parse trees from it before going about generating semantic representations. So in this area another application for semantic underspecification is lurking.

Several approaches to underspecification are conceivable. (1) Operational Underspecification: Construction operations that involve arbitrary choices are delayed and carried out only on demand [Alshawi, 1992], [Pinkal, 1995]. (2) Representational

[1]This work was funded by the German Federal Ministry of Education, Science, Research and Technology (BMBF) in the framework of the Verbmobil Project under Grant 01 IV 101 U. Many thanks are due to M. Dorna, J. Dörre, M. Emele, E. König-Baumer, C. Rohrer, C.J. Rupp, and C. Vogel.
Also appears in Proc. of COLING '96

Underspecification: The ambiguities are represented (explicitly or implicitly) in a formalism. A resolution procedure derives the full-fledged semantic representations. This paper opts for the second approach (for motivation see chapter 7). between the parser and the semantic construction component, too.

- Parse forests/charts [Alshawi, 1992].
- Underspecified "trees" with abstract dominance information [Pinkal, 1995].
- Fully specified parse trees [Egg and Lebeth, 1995]. The syntactic ambiguities are obtained by re-ambiguation in the semantic component.

Our choice are parse forests since there are well-known methods of construction for them and it is guaranteed that every syntactic ambiguity can be represented in this way. Furthermore a wide range of existing parsing systems, e.g. [Block and Schachtl, 1992], produce packed representations of this kind.

2 Outline of the System

Let us begin with a rough sketch of the architecture of the system. The semantic construction module works on parse forests and presupposes a semantic grammar of a certain kind (see chapter 6). The semantic grammar must be correlated with the syntactic grammar so that there is a one-to-one mapping between lexical entries and rules.

input string
↓
Parser using: Syntactic Grammar

↓
parse forest
↓
Semantic Construction Module using: Semantic Grammar

↓
packed UDRS

Inside the semantic construction module three processes are distinguished. They are described in turn (see chapter 4 and 6).

parse forest

$\downarrow$

Tree naming

$\downarrow$

disambiguated parse forest

$\downarrow$

Semantic construction proper
using: Semantic Grammar

$\downarrow$

packed UDRS

$\downarrow$

Determining unambiguous arguments

$\downarrow$

packed UDRS

3 Packed Shared Forests

In this section a formal description of packed shared forests in the sense of Tomita [Tomita, 1985] is given.

Let a **context-free grammar** $\mathcal{G}$ be a quadruple $< N, T, R, S >$ where N and T are finite disjoint sets of **nonterminal** symbols and **terminal** symbols, respectively, R is a set of **rules** of the form $A \rightarrow \alpha$ (A is a nonterminal and α a possibly empty string of nonterminal or terminal symbols), S is a special nonterminal, called **start symbol**. An **ordered directed graph** marked according to grammar $\mathcal{G}$ is a triple $< V, E, m >$ so that V is a finite set of **vertices** or **nodes**, E a finite set of **edges** e of the form $(v_1, \langle v_2, \ldots, v_n \rangle)$ ($v_i \in V, n \geq 2$, e starts at v_1, v_1 is the predecessor of $v_2, \ldots, v_n$), m is the **marking function** which associates with each vertex a terminal or nonterminal symbol or the special symbol ϵ. m is restricted so that the vertices on each edge are marked with the symbols of a rule in $\mathcal{G}$, the empty string being represented by the additional symbol ϵ. A **parse tree** is an ordered directed acyclic graph (DAG) satisfying the following constraints.

1. There is exactly one vertex without predecessors, called the top vertex or **root**. The root is marked with the start symbol.

2. For every vertex there is at most one edge starting at the vertex. Vertices that do not begin edges are called **leaves**, such that do are called **inner nodes**.

3. Every vertex except the root has exactly one predecessor.

A DAG satisfying the constraints (1–2) is called **Shared Forest**, a DAG only satisfying (1) is a **Packed Shared Forest** or parse forest (see figure 1). A packed shared forest for an input string σ obeys the further constraint that there must be at most one vertex for each grammar symbol and substring of σ. Thus, if σ consists of n words, there will be at most $k * n^2$ vertices in the parse forest for it (k being constant). Parse forests can be efficiently constructed using conventional parsing algorithms [Tomita, 1985], [Earley, 1970].

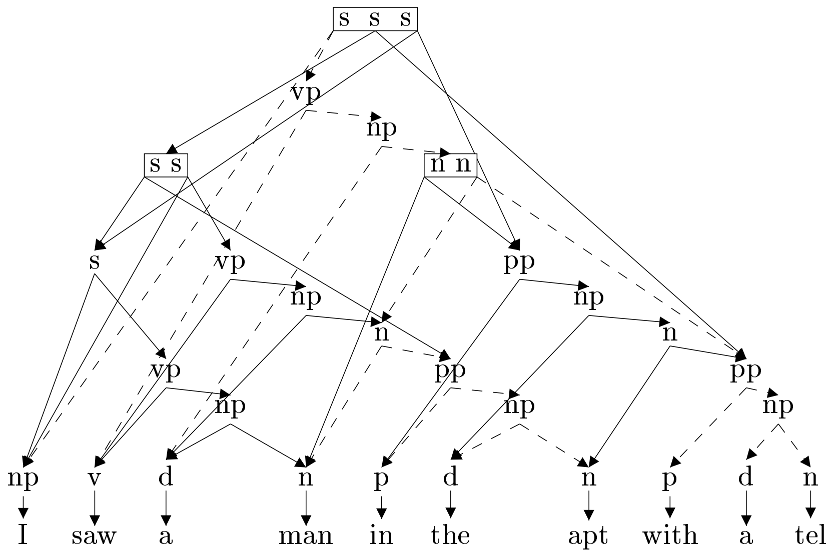

Figure 1: a parse forest with a tree reading d_1: edges used in d_1 are shown as broken lines

4 Determining Tree Readings from a Forest

A tree reading of forest F is a tree in F that contains the root and all leaves. Tree readings are treated as objects. An edge is *used* in a tree reading if it is one of the tree's edges. Let us now define a disambiguated parse forest (DPF for short). A DPF $\mathcal{D}$ is a quadruple $< V, D, E', m >$ such that

- V and D are finite disjoint sets. V is the set of vertices and D is the set of tree readings.
- E' is a finite set of edges of the form $(v_1, \langle v_2, \ldots, v_n \rangle, \{d_1, \ldots, d_m\})$. The third element is a set of tree readings ($\subseteq D$) and encodes the tree readings in which the edge is used.
- m is a marking function from vertices to grammar symbols.

To derive a DPF from a parse forest every edge must be assigned a set of tree readings. There is no simple way to determine from a parse forest the number of its tree readings. So instead of postulating a fixed set of readings the present approach uses pointers (implemented as Prolog variables) to refer to sets of tree readings. Two operations *disjoint union* and *multiplication* are defined for these set pointers. Both operations are monotonic in the sense that the pointers are not altered, their value is only specified. Let s_i be a set of tree readings.

- $s_1 \dot{\cup} s_2$
  The operator $\dot{\cup}$ differs from the set-theoretic notion of disjoint union in that it is neither commutative nor associative. This is so because on the implementational level commutativity and associativity would necessitate an abstract data type, thus a costly overhead.

- $s_1 \times s_2$
  In general, s_1 and s_2 correspond to formulae involving atomic sets and $\dot\cup$ operators: $s_1 = s_{11} \;\dot\cup\; \ldots \;\dot\cup\; s_{1m}$ and $s_2 = s_{21} \;\dot\cup\; \ldots \;\dot\cup\; s_{2n}$. The operation $\times$ introduces $m * n$ new atomic sets s'_{ij} and splits the former atomic sets such that $\forall i : 1 \leq i \leq m : s_{1i} = s'_{i1} \;\dot\cup\; \ldots \;\dot\cup\; s'_{in}$ and $\forall j : 1 \leq j \leq n : s_{2j} = s'_{1j} \;\dot\cup\; \ldots \;\dot\cup\; s'_{mj}$. The sets s_1 and s_2 are now equal modulo associativity and commutativity. Consider the following example:

$$
\begin{aligned}
&(s_1 \;\dot\cup\; s_2 \;\dot\cup\; s_3) \times (s_a \;\dot\cup\; s_b) \rightarrow \\
&(s_1 \;\dot\cup\; s_2 \;\dot\cup\; s_3) {=} (s'_{1a} \;\dot\cup\; s'_{1b}) \;\dot\cup\; (s'_{2a} \;\dot\cup\; s'_{2b}) \;\dot\cup\; (s'_{3a} \;\dot\cup\; s'_{3b}) \\
&\qquad (s_a \;\dot\cup\; s_b) {=} (s'_{1a} \;\dot\cup\; s'_{2a} \;\dot\cup\; s'_{3a}) \;\dot\cup\; (s'_{1b} \;\dot\cup\; s'_{2b} \;\dot\cup\; s'_{3b})
\end{aligned}
$$

We begin by associating a particular set pointer s_1 with the root vertex. s_1 refers to the total set of tree readings of the forest since the root vertex figures in all trees derivable from the forest. We then traverse the graph in top-down fashion applying to each new vertex v the following procedure:

> Let e_i be the set of tree readings at edge i ending in v, and b_j the set of tree readings at edge j starting in v. Then the following actions must be performed.
>
> - Apply the procedure to all successors of v. This step yields for each edge j starting in v and for each vertex u at the end of j a set of tree readings b'_{ju}.
> - $b_j = b'_{j1} \times \ldots \times b'_{jn}$ for each edge j starting in v
> - $(b_1 \;\dot\cup\; \ldots \;\dot\cup\; b_n) \times (e_1 \;\dot\cup\; \ldots \;\dot\cup\; e_m)$

If a vertex v has already been encountered the only action required is to connect the edge information on v's predecessor w with the edge information already present on vertex v. In particular, the successors of v need not be checked again.

> Let k be the edge over which the vertex v was reached from another vertex w in the top-down traversal. Let e_{kw} be the set of tree readings determined for edge k at vertex w and e_{kv} the set of tree readings determined for the edge at vertex v.
>
> - $e_{kv} \times e_{kw}$

5 Packed Underspecified Discourse Representation Structures

In this section an extension to UDRSs [Reyle, 1993] to express referentially underspecified semantic representations isp resented. First a definition of UDRSs is given. A UDRS $\mathcal{U}$ is a quadruple $< L, R, C, \leq>$ where L and R are disjoint finite sets of labels and discourse referents, respectively. The order relation $\leq$ forms a semilattice over L with one-element $l_\top$. C is a set of conditions of the following form

- $l : x$, where $l \in \mathcal{L}, x \in \mathcal{R}$.
- $l : p(x_1, \ldots, x_n)$, where $l \in \mathcal{L}, x_1, \ldots, x_n \in \mathcal{R}$, and p is an n-place predicate

- $l : l_1 \Rightarrow l_2$, where $l, l_1, l_2 \in \mathcal{L}$
- $l : \neg l_1$, where $l, l_1 \in \mathcal{L}$
- $l : l_1 \vee l_2$, where $l, l_1, l_2 \in \mathcal{L}$
- $l_1 \leq l_2$, where $l_1, l_2 \in \mathcal{L}$

In UDRSs $\mathcal{L} = L$ and $\mathcal{R} = R$.

To get packed UDRSs the UDRS language is extended by adding reified *contexts* (semantic readings) to it. The idea of using context variables to represent ambiguous structures originally stems from the literature on constraint-based formalisms [Dörre and Eisele, 1990]. A packed UDRS is a quintuple $< L, R, D, C', \leq >$ where L, R, $\leq$ are the same as in UDRSs, D is a finite set of contexts which is disjoint from L and R. C' is defined as in UDRSs except that (1) any condition may also be prefixed by a context set, and (2) label arguments may not only be labels but also functions from contexts to labels ($\mathcal{L} = L \cup (D \rightarrow L)$), and the same holds for discourse referents ($\mathcal{R} = R \cup (D \rightarrow R)$). If a function $\{A \rightarrow x_1, B \rightarrow x_2\}$ replaces a discourse referent in a packed UDRS, this intuitively means that the argument slot is filled by x_1 in reading A and by x_2 in reading B. As an example for a packed UDRS consider the following representation for *I saw every man with a telescope.*

$l_\top : i$
$anchor(i, speaker)$
$l_2 : see(e_1, i, x_1)$
$l_2 \leq l_\top$
$l_3 \leq l_\top$
$l_2 \leq l_4$
$l_3 : every(x_1, l_5, l_4)$
$l_5 : man(x_1)$
$\{A \rightarrow l_2, B \rightarrow l_5\} : with(\{A \rightarrow e_1, B \rightarrow x_1\}, x_2)$
$\{A \rightarrow l_2, B \rightarrow l_5\} \leq l_6$
$l_6 \leq l_\top$
$l_6 : x_2$
$l_6 : telescope(x_2)$

In the implementation contexts are represented by Prolog variables. In this way disambiguation is ensured to be monotonic[2]: A context d can be cancelled by grounding the Prolog variable representing d to a specific atom "no". The formalism also allows any kind of partially disambiguated structures since the variables for the readings do not interact.

In the above version of UDRS packing, disjuncts are reified. Another way to represent referential ambiguities is to reify argument slots using additional variable names (L and X below, not to be mistaken as discourse referents). Disjunctions are then represented directly.

$l_\top : i$
$anchor(i, speaker)$

[2]Another way to see that the resolution procedure is monotonic is to assume a semi-lattice over context sets with respect to the subset relation. Cancelling a context from a set makes it more specific in the semi-lattice.

$l_2 : see(e_1, i, x_1)$
$l_2 \leq l_\top$
$l_3 \leq l_\top$
$l_2 \leq l_4$
$l_3 : every(x_1, l_5, l_4)$
$l_5 : man(x_1)$
$L : with(X, x_2)$
$L \leq l_6$
$l_6 \leq l_\top$
$l_6 : x_2$
$l_6 : telescope(x_2)$
$(L = l_2 \ \wedge \ X = e_1) \vee (L = l_5 \ \wedge \ X = x_1)$

6 Building Semantic Representations

UDRS construction [Frank and Reyle, 1992], [Bos, 1995] is different from conventional semantic construction in that embedding is not represented directly but by means of labels. The only semantic composition operation is concatenation. In addition labels and discourse referents are matched as specified in the semantic part of the grammar rules (the "semantic grammar"). In the semantic grammar every nonterminal is assigned a list of arguments. For every operator (e.g. an NP) a lower label and a series of upper labels must be given. The lower label points to material which must be in the scope of the operator (e.g. the verb). The upper labels refer to the minimal scope domain the operator must occur in. This domain differs for indefinite NPs and quantifier NPs since these types of NPs are subject to different island constraints (only indefinites can be raised over clause boundaries). Furthermore, the semantic grammar specifies the UDRS conditions introduced by lexical items and rules and determines the arguments to be matched in rules and lexical items. It also gives the direction of this matching by fixing in which lexical item an argument originates (see last slot of lexical entries). If an argument originates in an item (because it is e.g. its instance discourse referent or label) then the value of this argument is unambigous for the item[3]. In adjunction structures, the modified constituent assigns and the modifier receives the shared discourse referent. Consider the following example grammar[4].

```
startsymbol(s/[_Event,_VerbL,Top,Top],
            [Top] ). % originating argument

s/[Event,VerbL,DomL,TopL] --->
            np/[X,VerbL,DomL,TopL],
            vp/[Event,X,VerbL,DomL,TopL].
```

[3] A similar train of thought lies behind the notion of "focus" proposed by Tomita [Tomita, 1985]. A "focus" in a rule is the constituent which gets assigned an argument from the "background" constituents of the rule. In general this notion of focus must be relativised to individual arguments. Constituent 1 can be focus with respect to argument i while constituent 2 is focus for argument j in a rule.

[4] The Prolog symbol **leq** represents the UDRS subordination relation $\leq$.

```
vp/[Event,X,VerbL,DomL,TopL] --->
          vt/[Event,X,Y,VerbL,DomL],
          np/[Y,VerbL,DomL,TopL].

np/[X,VerbL,DomL,TopL] --->
          det/[X,NounL,VerbL,DomL,TopL],
          n/[X,NounL,DomL,TopL].

lex(a,    det/[X,Lab,VerbL,_DomL,TopL],
          [       leq(VerbL,Lab),
                  leq(Lab,TopL),
                  Lab:X
          ], [X] ).  % originating argument
lex(every, det/[X,ResL,VerbL,DomL,_TopL],
          [       leq(+Lab,DomL),
                  leq(VerbL,ScopeL),
                  Lab:every(X,ResL,ScopeL)
          ],
          [X,Lab,ScopeL] ).

lex(man,  n/[X,Lab,_DomL,_TopL],
          [       Lab:man(X)
          ],
          [Lab] ).

lex(saw,  vt/[Event,X,Y,Lab,DomL],
          [       Lab:see(Event,X,Y),
                  leq(Lab,DomL)
          ],
          [Lab,Event] ).
```

Let us turn now to the semantic construction component. The tree readings of the DPF correspond to the contexts of the packed UDRS. The motivation behind this layout is that in most cases syntactic ambiguity has some impact on the semantic readings[5]. The construction algorithm traverses the DPF and assigns to each vertex the argument list associated with its category in the semantic grammar. The arguments on this list are not arguments proper as they would be if only parse trees were considered, but *functions* from contexts to arguments proper. These functions are total only for the root and the leaves, for inner nodes v they are restricted to the union D_1 of the context sets at the edges starting at v. A predicate **match1** matches arguments proper as given in the lexical entries and the startsymbol declaration onto functions as used in the rules.

> Let D_1 be a context set $\{d_1, \ldots, d_n\}$, let LexArg be an argument as provided by a lexical item or startsymbol declaration I, let Arg be an argument as occurring attached to a nonterminal on the right-hand side of a grammar rule.

[5] If several tree readings correspond to a single context (semantic reading) this is recognised in the last step (determining unambiguous arguments) where the tree readings are merged.

> Then the predicate **match1** unifies LexArg with Arg if LexArg does not originate in I. If LexArg does, Arg is unified with the function $\{d_1 \rightarrow LexArg, \ldots, d_n \rightarrow LexArg\}$.

Let us assume a bottom-up traversal of the parse forest and let e be the edge from v to one of its successors w. Then the arguments already present[6] at w must be matched with the arguments predicted for w by the semantic rule corresponding to e (predicate **match2**). Let D_2 be the context set assigned to e. Then only the argument values of the contexts in D_2 are unified. In this way it is guaranteed that argument matching is done as it would be done in the underlying trees: The contexts clearly separate the information flow.

> Let D_2 be the context set $\{d_1, \ldots, d_n\}$ at e, let UpperArg be an argument as provided by the semantic rule corresponding to edge e, let LowerArg be an argument as attached to the vertex w.
> Then the predicate **match2** unifies UpperArg with the restriction of the function LowerArg to the contexts in D_2 $\{d_1 \rightarrow v_1, \ldots, d_n \rightarrow v_n\}$ (a subset of LowerArg).

In the final step the packed UDRS is traversed and functions where all contexts point to a single value are replaced by this value.

7 Comparison with Other Approaches

This section discusses two evaluation criteria for approaches to semantic underspecification. The present proposal is measured against the criteria, and so are the Minimal Recursion Semantics approach [Egg and Lebeth, 1995], the Radical Underspecification approach [Pinkal, 1995], and the Core Language Engine approach [Alshawi, 1992]. The first criterion is coverage. Several types of syntactic ambiguities can be distinguished.

- adjunction ambiguities (arising from attachment of PPs, adjectives, adverbial subclauses, and other modifiers)
- coordination ambiguities
- θ-role assignment ambiguities (arising from scrambling)
- ambiguities arising from multi-part-of-speech words (A subcase of this type of ambiguity is the treatment of unknown input words.)

The MRS approach is restricted to adjunction ambiguities, while the other approaches are applicable to all the kinds of ambiguities mentioned. A drawback of the MRS approach might be that it generates semantic readings which are not licensed by the syntactic structure. To give an example consider the sentence *I saw a man in the apartment with a telescope.* MRS produces a spurious reading in which the PP *with a telescope* adjoins to the NP *a man* while the PP *in the apartment* modifies the full sentence. Remember that MRS does not use a parse forest as input structure but an arbitrary parse tree, i.e. one specific syntactic reading. MRS

[6] The bottom-up assumption makes sure that vertex w has been treated.

PPs	Readings	U-Nodes	U-Time (per reading)		S-Time (per reading)	
n=1	2	16	75 msec	(37.5 msec)	15 msec	(7.5 msec)
n=2	5	28	180 msec	(36.0 msec)	70 msec	(14.0 msec)
n=3	14	43	430 msec	(30.7 msec)	355 msec	(25.4 msec)
n=4	42	61	1115 msec	(26.5 msec)	2225 msec	(53.0 msec)
n=5	132	82	3145 msec	(23.8 msec)	16895 msec	(128.0 msec)
n=6	429	106	10505 msec	(24.5 msec)	176930 msec	(412.4 msec)
n=7	1430	133	32195 msec	(22.5 msec)	441630 msec	(308.8 msec)
n=8	4862	163	131125 msec	(27.0 msec)	4331120 msec	(890.8 msec)

Table 1: Result of Experiment

re-ambiguates the parse tree only afterwards within semantic construction. At this point information about positions in the input string is lost.

Another test is the usefulness of the representation for further processing. Such processes are

- disambiguation by sort hierarchies
- theorem proving
- transfer and generation

All these processes can successfully handle scopally underspecified structures (for sortal disambiguation and transfer see the Core Language Engine [Alshawi, 1992], for theorem proving see the Underspecified DRS formalism [Reyle, 1993]). In the Core Language Engine approach to syntactic underspecification the representation must be unpacked to perform disambiguation by sorts. This seems to be true for any approach relying on delay of semantic construction operations: In order to apply the sortal restrictions of, e.g., a verb to one of its argument discourse referents it must be known which discourse referents could possibly fill the argument slot. Moore and Alshawi [Alshawi, 1992] explain their reluctance to apply sort restrictions already in the packed structure with the maintenance overhead in associating semantic records with vertices of the forest. In the packed UDRS approach the problem is handled by explicitly enumerating all possible readings. Then, the maintenance effort is reduced to the effort of extrapolating the tree readings from the parse forest. None of the compared approaches makes any claims about theorem proving and transfer. In the packed UDRS approach it is conceivable to delay actual disambiguation as long as possible: Apart from the potential representation of referential ambiguities by functions packed UDRSs look exactly like UDRSs. So if only referentially unambiguous conditions must be consulted in a proof, a UDRS theorem prover may be used.

8 Efficiency

This section reports on an experiment in which the efficiency of the proposed underspecified construction mechanism was measured against the cost of generating all UDRSs separately. Table 1 compares the time behaviour of constructing one underspecified structure (U-Time) with the time needed for constructing of the whole bunch of specified structures (S-Time). The experiment was conducted on a SPARCstation 20 using input sentences of the form *I saw a man (with a telescope)*^{n}.

Visibly the time needed per reading remains approximately constant in the construction of the underspecified representation whereas it grows sharply when the ambiguities are enumerated.

References

[Alshawi, 1992] Hiyan Alshawi, ed. 1992. The Core Language Engine. MIT Press, Cambridge, Massachusetts.

[Block and Schachtl, 1992] Hans Ulrich Block and Stefanie Schachtl. 1992. Trace and Unification Grammar. In *Proceedings of the fifteenth International Conference on Computational Linguistics*, pages 87–94, Nantes, France, August.

[Bos, 1995] Johan Bos. 1995. Predicate Logic Unplugged. In *Proceedings of the Tenth Amsterdam Colloquium*, ILLC/Department of Philosophy, University of Amsterdam, Amsterdam, Holland, December.

[Dörre and Eisele, 1990] Jochen Dörre and Andreas Eisele. 1990. Feature Logic with Disjunctive Unification. In *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki, Finland.

[Earley, 1970] Jay Earley. 1970. An Efficient Context-Free Parsing Algorithm. In *Communications of the ACM*, 13(2), pages 94–102, February.

[Egg and Lebeth, 1995] Markus Egg and Kai Lebeth. 1995. Semantic underspecification and modifier attachment ambiguities. In *James Kilbury, Richard Wiese (ed.), Integrative Ansätze in der Computerlinguistik. Beiträge zur 5. Fachtagung der Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft (DGfS)*, pages 19–24, Düsseldorf, Germany.

[Frank and Reyle, 1992] Anette Frank and Uwe Reyle. 1992. How to Cope with Scrambling and Scope. In *Görz, G. (ed.) KONVENS '92. Reihe Informatik aktuell, Springer Berlin*, pages 121–130, Nürnberg, Germany.

[Kamp and Reyle, 1993] Hans Kamp and Uwe Reyle. 1993. From Discourse to Logic. Kluwer, Dordrecht, Holland.

[Pinkal, 1995] Manfred Pinkal. 1995. Radical Underspecification. In *Proceedings of the Tenth Amsterdam Colloquium*, ILLC/Department of Philosophy, University of Amsterdam, Amsterdam, Holland, December.

[Reyle, 1993] Uwe Reyle. 1993. Dealing with Ambiguities by Underspecification: Construction, Representation and Deduction. In *Journal of Semantics*, 10, 2, pages 123–179.

[Tomita, 1985] Masaru Tomita. 1985. Efficient Parsing for Natural Language. Kluwer, Dordrecht, Holland.