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

Verbmobil
Verbundvorhaben

Disambiguation of Underspecified Discourse Representation Structures under Anaphoric Constraints

Michael Schiehlen

Universität Stuttgart

Vm Report 188
January 1997

January 1997

Michael Schiehlen

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

Tel.: (0711) 121 - 1352
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.

Disambiguation of Underspecified Discourse Representation Structures under Anaphoric Constraints

Michael Schiehlen[1]
Institute for Computational Linguistics, University of Stuttgart,
Azenbergstr. 12, D-70174 Stuttgart
email: mike@adler.ims.uni-stuttgart.de

Abstract

The paper explores the possibility of performing anaphora resolution and presupposition accommodation in a situation where scope relations are only partially known. For underspecification of scope ambiguity, the approach builds on Reyle's formalism of Underspecified Discourse Representation Structures (UDRSs, (Reyle, 1993)) which is extended to cover arbitrary constraints from anaphora resolution. Wellformedness conditions are formulated for UDRSs to restrict scope readings as required by anaphora resolution. Based on these conditions, a disambiguation algorithm is presented for scope resolution and presupposition accommodation, which has also been implemented.

Keywords: definition and use of underspecified semantic representations, scope resolution, anaphora and presupposition.

1 Introduction

In natural language discourse the phenomenon of ambiguity is pervasive. As the number of interpretations is in general large, an enumeration of all interpretations is apt to slow down a Natural Language Processing system considerably. The traditional approach to the problem (Woods, 1978) consists in applying heuristics (Poesio, 1995) to get a single "preferred" interpretation and resorting to backtracking in case of wrong choice. Unfortunately in the worst case all readings are enumerated. This is why another method, dubbed *underspecification*, has become the focus of attention. The main insight behind this approach is that evaluation processes can very often work on whole classes of interpretations so that disambiguation among the members of these classes is superfluous. Two things are required for underspecification: (1) a formalism for compact representation of interpretation classes (*underspecified* representations) and individual interpretations (*fully specified* representations) and (2) a *disambiguation device*

[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 J. Bos, E. König-Baumer, P. Krause, U. Reyle, and C.J. Rupp.
Also appeared in *Proc. of the Second International Workshop on Computational Semantics*, Department of Computational Linguistics, Tilburg University, 1997.

to connect underspecified representations with fully specified representations. At least the fully specified representations should be interpretable in a model theory. Two roads lead to underspecification.

Operational Underspecification: Operations that typically introduce ambiguity are recorded in the representation formalism and executed only by the disambiguation device. For scope resolution, this approach has been taken e.g. in the Core Language Engine (Alshawi, 1992) where the quantifier raising rules are recorded in quantified terms and in Ambiguous Predicate Logic (van Eijck and Jaspars, 1996) where nested Cooper storages (Cooper, 1983) are explicitly integrated into the language.

Representational Underspecification: Another approach practicable for scope resolution is to reify the structuring objects of a semantic representation formalism. The structural relations between these objects are then appropriately weakened to capture the correct range of ambiguity. This approach is pursued in Underspecified Discourse Representation Theory (UDRT) (Reyle, 1993), (Frank and Reyle, 1992) which has pointers to basic formulae (labels[2]) in its repertory. A traditional semantic representation[3] has a tree structure. UDRT weakens such trees to directed acyclic graphs. Representational Underspecification has the advantage that all binding constraints can be made explicit so that the Free Variable Constraint (Hobbs and Shieber, 1987), (van der Sandt, 1992, 365) plaguing quantifier raising does not need extra enforcement. On the other hand the well-formedness conditions of UDRSs transcend what can be read off the representations' form and are thus more costly to check.

This paper adopts the UDRS formalism. It further assumes that the evaluative processes of anaphora resolution and presupposition binding or accommodation (van der Sandt, 1992) work on UDRSs directly, i.e. with unresolved scope. UDRT is u-deductive (König-Baumer and Reyle, 1996) (i.e. deductions can be done directly on the underspecified structure) and therefore supports the proving required for presupposition binding. In the second section the UDRS formalism with extensions for anaphora binding is introduced. The third section discusses a problem that crops up when scope resolution and donkey anaphora are considered together. The following sections propose a solution in terms of *ambiguity domains* and extend the UDRS well-formedness conditions to this purpose. Finally, an algorithm for scope resolution and presupposition accommodation is presented that obeys the constraints imposed by anaphora resolution and presupposition binding.

[2]The idea of labels and labelled conditions advocated in UDRT has been found useful in applications like theorem proving (Reyle, 1993), (Reyle, 1995), shake-and-bake machine translation (Copestake et al., 1995), (Dorna and Emele, 1996) and underspecification of syntactic ambiguity (Schiehlen, 1996).

[3]That is, a formula of first order predicate logic (PL1) or a Discourse Representation Structure (DRS) of Discourse Representation Theory (DRT, (Kamp and Reyle, 1993)).

2 Underspecified Discourse Representation Structures

This section extends the UDRS formalism with two new sorts of constraints: accessibility and binding constraints. Accessibility constraints are needed to avoid disjunction[4] in the constraint language (see definition 11). $[l_1 \text{ acc } l_2]$ means that l_2 is a member of the A-structure of l_1 (van der Sandt, 1992, 354). Binding constraints record which anaphors have been bound and which are still awaiting accommodation (see definitions 6 and 7). Thus, $[l_1 \hookrightarrow l_2]$ records a transfer of an anaphoric sub-DRS l_1 to its binding site l_2. Anaphora resolution introduces binding constraints and equality constraints for the discourse referents (van der Sandt, 1992, 358), semantic construction all the other constraints.

Definition 1 (UDRS)
Let R be a set of discourse referents, L a set of labels, V a set of predicates and O a set of operators (negation, conditional, disjunction, quantifiers in DRT). Then K is a UDRS confined to R, L, V, O iff K is a finite set consisting of conditions of the following form.

- structural information
  - top label constraint
    $[\text{top}(l_1)]$, where $l_1 \in L$
  - subordination constraint
    $[l_1 \leq l_2]$, where $l_1, l_2 \in L$
  - accessibility constraint
    $[l_1 \text{ acc } l_2]$, where $l_1, l_2 \in L$
  - binding constraint
    $[l_1 \hookrightarrow l_2]$, where $l_1, l_2 \in L$
- content information
  - universe
    $[l_1 : x]$, where $l_1 \in L$, $x \in R$
  - atomic condition
    $[l_1 : P(x_1, \ldots, x_n)]$, where P is an n-place predicate in V, $l_1 \in L$, $x_1, \ldots, x_n \in R$
- structural and content information
  - complex condition
    $[l : Q(l_1, \ldots, l_n)]$, where $l, l_1, \ldots, l_n \in L$, Q an n-place operator in O

It is useful to distinguish between overt constraints introduced by semantic construction or anaphora resolution and implicit constraints or relations. Overt constraints are represented in square brackets. For an illustration of a UDRS

[4]In sentences like *Every representative of a big computer company uses a laptop manufactured by it* the *laptop* must be subordinate or equal to the *company* unless the *company* is in the quantifier's restriction in which case the *laptop* must be subordinate or equal to the quantifier's nuclear scope.

enriched with information from anaphora resolution look at the following example.

Every farmer likes his donkey.

$[\mathrm{top}(l_\top)]$	$[l_5 \leq l_3]$	$[l_7 : \mathrm{donkey}(x_2)]$
$[l_1 \leq l_\top]$	$[l_5 : \mathrm{like}(x_1,x_2)]$	$[l_7 : \mathrm{of}(x_2,x_3)]$
$[l_1 : \mathrm{every}(x_1, l_2, l_3)]$	$[l_6 \leq l_\top]$	$[l_7 \text{ acc } l_8]$
$[l_2 : x_1]$	$[l_6 : x_2]$	$[l_8 : x_3]$
$[l_4 \leq l_2]$	$[l_5 \text{ acc } l_6]$	$[l_8 : x_1 = x_3]$
$[l_4 : \mathrm{farmer}(x_1)]$	$[l_7 \leq l_6]$	$[l_8 \hookrightarrow l_2]$

The same UDRS is depicted below in a Hasse diagram. Continuous lines without arrows represent immediate subordination links ($[\prec]$, definition 2), continuous lines with arrows stand for subordination links ($[\leq]$). Dashed lines designate accessibility links ([acc]).

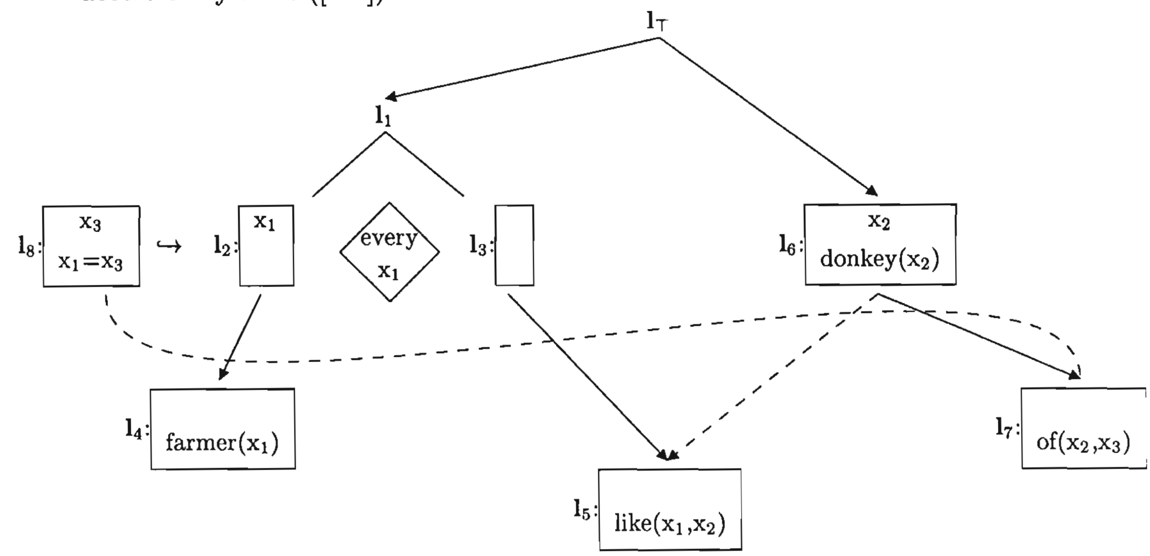


Figure 1: Every farmer likes his donkey

Some useful relations on the UDRS graph structures are defined below. Overt immediate subordination pulls out the structural component of the content of complex conditions.

Definition 2 (overt immediate subordination)
$[l_1 \prec l_2] \leftrightarrow [l_2 : Q(\ldots, l_1, \ldots)]$

Definition 3 (overt subordination)
$l_1 \sqsubseteq l_2 \leftrightarrow [l_1 \leq l_2] \vee [l_1 \prec l_2]$

Definition 4 (leaf [5])
$\mathrm{leaf}_R(l_1) \leftrightarrow \neg\exists l_2 : l_2\ R^*\ l_1$

In UDRT, labels can serve two purposes: Either labels are place holders for sub-DRSs and have a function in the DRS tree. or they are simply used to designate semantic material. Tree labels are special in that they can never be set equal with each other: A well-formedness condition (unique tree labels) prevents inadvertently "unifying" tree labels (that is, sub-DRSs), which would destroy the tree order.

Definition 5 (tree labels)
l_1 is a tree label (written $\mathbf{t}_1$) $\leftrightarrow l_1 = l_\top \ \vee\ \exists l_2 : [l_1 \prec l_2]$.
Labels which are not tree labels are called ambiguity labels (written $\mathbf{a}_1$). The only function of ambiguity labels is to bear some content information.

Not all anaphoric material must be bound. According to van der Sandt's (1992) proposal definite descriptions can optionally be accommodated. Since by assumption the binding process is already over, having worked on the underspecified representation, and only accommodation still needs to be done, we can distinguish accommodation from binding by the fact that no binding constraints are introduced.

Definition 6 (resolved anaphor)
l_1 racc $l_2 \leftrightarrow \exists l_3 : [l_1 \text{ acc } l_3] \ \wedge\ [l_3 \hookrightarrow l_2]$

Definition 7 (anaphor awaiting accommodation[6])
l_1 aacc $l_2 \leftrightarrow [l_1 \text{ acc } l_2] \ \wedge\ \neg\exists l_3 : [l_2 \hookrightarrow l_3]$

Definition 8 (anaphoric link)
l_1 acc $l_2 \leftrightarrow l_1$ racc $l_2 \ \vee\ l_1$ aacc l_2

The auxiliary definitions introduced so far allow us to formulate a set of well-formedness conditions on UDRSs. One of the well-formedness conditions of UDRSs (which Reyle (1993) calls goodness) prohibits alignment of both restriction and scope of complex conditions on a single branch. The purpose of this condition is to ensure that every DRS corresponds to a tree structure, i.e. a structure where every path from a label to the root is linear. We prefer to call the condition Linear Branches, the name by which it is known in the tree literature (Backofen et al., 1995).

Constraint 1 (Well-Formedness Conditions)

- Existence of a root
  $\forall l_1 : l_1 \sqsubseteq^* \mathbf{t}_\top$
- Tree labels are unique[7].
  $\forall \mathbf{t}_1, \mathbf{t}_2 : \mathbf{t}_1 \sim \mathbf{t}_2 \rightarrow \mathbf{t}_1 = \mathbf{t}_2$
- Linear Branches
  $\forall l_1, l_2, l_3 : l_1 \sqsubseteq^* l_2 \ \wedge\ l_1 \sqsubseteq^* l_3 \ \wedge\ l_2 \neq l_3 \rightarrow \neg\exists l_4 : [l_2 \prec l_4] \ \wedge\ [l_3 \prec l_4]$
- Acyclicity
  $\forall l_1 : \neg l_1 \ (\sqsubseteq \cup \text{acc})^+ \ l_1$

[5] The symbols * and $^+$ denote transitive closure as usual.
[6] l_1 aacc l_2 means that l_2 is still in the A-structure of l_1 and has not been moved out yet.

Another useful definition singles out the links produced by the semantic construction component and thus ultimately motivated by syntax.

Definition 9 (syntactic link)
$l_1 \sqsubseteq_a l_2 \leftrightarrow l_1 \sqsubseteq l_2 \lor l_1 \text{ aacc } l_2$

In DRT some operators are barriers to anaphoric relations, some are not. This classification is used in the definition of accessibility (Kamp and Reyle, 1993).

Definition 10 (operators licensing accessibility)
$\mathbf{t}_1 \Rightarrow \mathbf{t}_2 \leftrightarrow \exists l_1 : [l_1 : Q(\mathbf{t}_1, \mathbf{t}_2)]$, where Q is a suitable complex condition (quantifiers and conditional in DRT). $\mathbf{t}_1$ is the restriction and $\mathbf{t}_2$ the nuclear scope of Q.

Definition 11 (accessibility[8])
l_1 is accessible to $l_2 \leftrightarrow l_1 \leq l_2 \lor \exists l_3 : (l_2 \Rightarrow l_3 \land l_1 \leq l_3)$

3 Problems with Scope and Anaphora

Let us have a second look at the UDRS in Figure 1. Why can the definite l_6 not get wide scope? The box with the donkey l_7 must be subordinate to either the restriction l_2 or the scope l_3 of the universal quantifier l_1 for l_2 to be accessible from l_7. In any case l_1 along with l_7 would be in the nuclear scope of the definite if the definite had wide scope. Since l_7 is also in the definite's restriction the configuration violates Linear Branches (restriction and scope cannot be aligned). Figure 2 is an example of multi-sentence discourse.

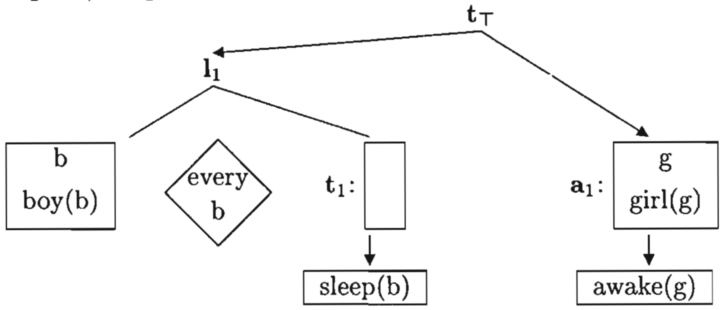


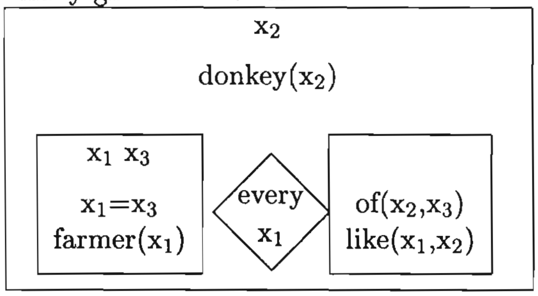
Figure 2: Every boy was sleeping. A girl was awake.

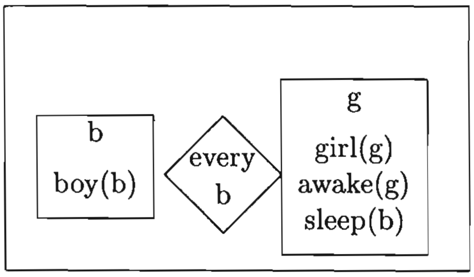
The box $\mathbf{a}_1$ cannot go into the nuclear scope $\mathbf{t}_1$ of the universal since in this case $\mathbf{a}_1$ would be subordinate to both the first and second conjunct of $\mathbf{t}_\top$.

But wait a minute. Both these explanations refer to PL1 concepts unknown in DRT (Kamp and Reyle, 1993): restriction and scope of a definite or indefinite

[7] $\sim$ is the relation of scope equivalence.
[8] $<$ is DRS subordination.

article, conjuncts of a conjunction operator. In DRT neither definites nor indefinites have a restriction or a nuclear scope, since they lack quantificational force. Donkey anaphora (*Every farmer who owns a donkey beats it.*) shows that the indefinite inherits both quantificational force and nuclear scope from the universal since retention of the original existential force would make the representation uninterpretable. Likewise, conjunctions are no obstacles to the scope extension of definites and indefinites, witness cross-sentential anaphora. So there are some good reasons to discard the surplus structure. But exactly this structure is apparently needed to exclude the illicit readings. Without further provisos the following perfectly well-formed DRSs can be constructed for the wrong readings, the one with the common donkey and the one that asserts many girls to be awake.





The literature does not help very much with the problem. Most authors have focused on scope resolution alone and simply ignored donkey anaphora, intersentential anaphora and other DRT phenomena. So Bos (1995) treats indefinites just like ordinary quantifiers. Reyle (1993) uses σ-conditions to keep material from different sentences apart. The interaction of σ-conditions with accessibility remains opaque. Another technique discussed by Reyle (1993), dependency constraints, unfortunately is lacking in detail. It is not clear how these constraints could be automatically derived from parse trees.

4 Ambiguity Domains

In order to discard the inadmissible readings discussed above we are left with finding a surrogate for Linear Branches. The main idea is to draw a distinction between leaf labels[9] (standing for predicates) and inner labels (which represent PL1 operators). Using this dichotomy a restriction on subordination relations is formulated as an additional well-formedness condition on UDRSs: Connectedness states that a subordination relation ($\leq$) between two labels l_1 and l_2 can only be introduced if l_1 and l_2 are *connected*, i.e. if they are members to the same extended *ambiguity domain*. Put differently, ambiguity domains are defined as the sets of labels which can take scope over each other.

[9]The leaf labels of the UDRS in Figure 1 are l_4, l_5, and l_7.

Constraint 2 (Well-Formedness Condition)

- Connectedness
  $\forall l_1, l_2 : l_1 \leq l_2 \rightarrow \exists l_3 : l_1 \in EAD(l_3) \wedge l_2 \in EAD(l_3)$.

At least those operators can take scope over each other which depend on the same predicate. This insight gives us a first definition of ambiguity domains. Note that the definition only considers links introduced by semantic construction ($\sqsubseteq_a$).

Definition 12 (ambiguity domain)
$l_1 \in AD(l_2) \leftrightarrow l_2 \sqsubseteq_a^* l_1$ where $leaf_{\sqsubseteq_a}(l_2)$

According to definition 12 there is a one-to-one relation between leaf labels and ambiguity domains such that leaf labels identify ambiguity domains. In the course of scope resolution ambiguity domains can be extended: The subordination links which are introduced by scope resolution ($\leq$) admit new members to ambiguity domains. This is where recursion enters the picture: Subordination and extended ambiguity domains depend on each other.

Definition 13 (extended ambiguity domain)
$l_1 \in EAD(l_2) \leftrightarrow l_2(\sqsubseteq_a \cup \leq)^* l_1$ where $leaf_{\sqsubseteq_a}(l_2)$

Theorem 1 states that the term "connected" as it is used here (sharing an extended ambiguity domain) is interchangeable with the term "comparable" known from order theory (standing in a subordination relation).

Theorem 1
$\forall l_1, l_2 : \exists l_3 : l_1 \in EAD(l_3) \wedge l_2 \in EAD(l_3) \leftrightarrow l_1 \leq l_2 \vee l_2 \leq l_1$.
Proof: ($\rightarrow$) If $\neg l_1 \leq l_2 \wedge \neg l_2 \leq l_1$ we infer from the tree axioms that $\exists l_3, l_4, l_5 : [l_3 \prec l_4] \wedge [l_5 \prec l_4] \wedge l_3 \neq l_5 \wedge l_1 \leq l_3 \wedge l_2 \leq l_5$. Suppose l_1 and l_2 are in $EAD(l_6)$. Then l_6 is subordinate to both l_3 and l_5 violating Linear Branches. ($\leftarrow$) Connectedness.

5 Conjoining Chains

Definitions 14 – 16 serve to single out the relevant PL1 two-place operators. Every PL1 two-place operator is superordinate to two leaf labels which are guaranteed to end up in its two argument places. Hence every conjoining label is a PL1 two-place operator.

Definition 14 (conjoining label)
l_1 conjoins $AD(l_2)$ and $AD(l_3) \leftrightarrow l_1 = \min\{l | l \in AD(l_2) \wedge l \in AD(l_3)\}$ where $l_2 \neq l_3$

Definition 15 (conjoining sequence)
$\langle \mathbf{l}_1, \ldots, \mathbf{l}_n \rangle$ conjoins two ambiguity domains A_1 and $A_{n+1} \leftrightarrow \exists A_2, \ldots, A_n \forall i \in \{1, \ldots, n\} : \mathbf{l}_i$ conjoins A_i and A_{i+1}, where none of the ambiguity domains $A_1, \ldots, A_{n+1}$ is equal to another and no label occurs twice in the sequence.

A well-formedness condition (unique conjoining labels) forbids incomparable conjoining sequences: Whenever there are two conjoining sequences between particular ambiguity domains, a member of one of them subordinates all the other labels. As a side effect, the condition guarantees that conjoining labels are unique.

Constraint 3 (Well-Formedness Condition)

- Unique Conjoining Labels[10]
  If a label $\mathbf{l}_0$ and a sequence $\langle \mathbf{l}_1, \ldots, \mathbf{l}_n \rangle$ conjoin A_1 and A_2 then $\exists \mathbf{l} \in \{\mathbf{l}_0, \ldots, \mathbf{l}_n\} : \mathbf{l}_0 \sqsubseteq^* \mathbf{l} \wedge \ldots \wedge \mathbf{l}_n \sqsubseteq^* \mathbf{l}$

A conjoining chain is a "minimal" conjoining sequence. Conjoining chains exist and are unique, since according to condition 3 conjoining sequences are comparable.

Definition 16 (conjoining chain[11])
A sequence of labels $\langle \mathbf{l}_1, \ldots, \mathbf{l}_n \rangle$ is a conjoining chain from $\mathbf{l}$ to $\mathbf{l}' \leftrightarrow$ it is the shortest sequence that conjoins some ambiguity domains A_1 and A_2 such that $\mathbf{l} \in A_1 \wedge \mathbf{l}' \in A_2$ and $\forall \langle \mathbf{k}_1, \ldots, \mathbf{k}_m \rangle$ that conjoin A_1 and A_2 $\forall \mathbf{l}_j \exists \mathbf{k}_i : \mathbf{l}_j \sqsubseteq^* \mathbf{k}_i$.

Theorem 2
$\mathbf{l}_1 \leq \mathbf{l}_2 \rightarrow \forall\, \mathbf{l}_i$ in the chain from $\mathbf{l}_1$ to $\mathbf{l}_2$: $\mathbf{l}_i \leq \mathbf{l}_2$
Proof: By Connectedness $\mathbf{l}_1$ and $\mathbf{l}_2$ must be in the same extended ambiguity domain $\mathbf{l}_3$. By the recursion assumption there must be a conjoining chain from $\mathbf{l}_3$ to $\mathbf{l}_1$ and one from $\mathbf{l}_3$ to $\mathbf{l}_2$ such that all chain members are subordinate to $\mathbf{l}_1$ or $\mathbf{l}_2$, respectively. But if a label is subordinate to $\mathbf{l}_1$ it is also subordinate to $\mathbf{l}_2$.

Theorem 2 retraces Linear Branches for PL1 operators. If $\mathbf{l}_1$ is subordinate to $\mathbf{l}_2$ and the operator $\mathbf{l}_3$ conjoins the ambiguity domains A_1 of $\mathbf{l}_1$ and A_2 of $\mathbf{l}_2$, then $\mathbf{l}_2$ must not be subordinate to $\mathbf{l}_3$, since $\mathbf{l}_3$'s argument places, A_1 and A_2, are both subordinate to $\mathbf{l}_2$. We can now explain the examples of Figure 1 and 2. In Figure 1 $\mathbf{l}_2$ is accessible to $\mathbf{l}_7$, hence $\mathbf{l}_7$ subordinate to $\mathbf{l}_3$. By theorem 2 the definite $\mathbf{l}_6$, which is on the chain from $\mathbf{l}_7$ to $\mathbf{l}_3$, is also subordinate to $\mathbf{l}_3$. In Figure 2 the possibility of $\mathbf{a}_1$ being subordinate to $\mathbf{t}_1$ is discussed. The chain from $\mathbf{a}_1$ to $\mathbf{t}_1$ consists of the single element $\mathbf{t}_\top$ which would have to be subordinate, by the root condition equal, to $\mathbf{t}_1$. Tree labels cannot be set equal.

[10]By constraint 3 semantic construction must not produce a configuration like $P(a,b), Q(b,c), R(c,a)$.

[11]In Figure 1 $\mathbf{l}_1$ and $\langle \mathbf{l}_6, \mathbf{l}_\top \rangle$ conjoin AD($\mathbf{l}_5$) and AD($\mathbf{l}_4$), $\mathbf{l}_6$ and $\langle \mathbf{l}_\top, \mathbf{l}_1 \rangle$ conjoin AD($\mathbf{l}_7$) and AD($\mathbf{l}_5$), $\mathbf{l}_\top$ and $\langle \mathbf{l}_6, \mathbf{l}_1 \rangle$ conjoin AD($\mathbf{l}_7$) and AD($\mathbf{l}_4$). The conjoining chain from AD($\mathbf{l}_7$) to AD($\mathbf{l}_5$) is $\langle \mathbf{l}_6 \rangle$, the conjoining chain from AD($\mathbf{l}_7$) to AD($\mathbf{l}_4$) is $\langle \mathbf{l}_6, \mathbf{l}_1 \rangle$.

6 Computed Subordination

The subordination relation in a DRS tree structure is partly given by syntax (overt subordination, $\sqsubseteq$), partly determined in the course of scope resolution (computed subordination, $\ll$, see a functional definition below).

Definition 17 (computed subordination, functional)
$l_1 \ll l_2 \leftrightarrow l_1 \leq l_2 \;\wedge\; \neg l_1 \sqsubseteq^* l_2$

To define computed subordination for a tree label $\mathbf{t}$ constructively we make an induction assumption that all scope relations involving tree labels superordinate to $\mathbf{t}$ have already been resolved. l_1 is then computable to be subordinate to $\mathbf{t}$ if l_1 could have been assigned to some ancestor $\mathbf{t}_1$ of $\mathbf{t}$ ($l_1 \sqsubseteq^+ |_a \mathbf{t}_1$) but was not ($\neg \mathbf{t} < l_1$). Any label l_1 overtly subordinate ($\sqsubseteq^+$) to $\mathbf{t}_1$ can be equal to $\mathbf{t}_1$ except for tree labels, so we restrict $\sqsubseteq$ to ambiguity labels ($\sqsubseteq^+ |_a$). For l_1 to be subordinate to $\mathbf{t}$, l_1 has to be connected with $\mathbf{t}$ (constraint 2). Since connectedness implies comparability (theorem 1), the only place where we have to look for potentially subordinate labels l_1 that are not overtly subordinate is $\mathbf{t}$'s ancestor line.

Definition 18 (computed subordination, constructive)
$l_1 \ll \mathbf{t} \leftrightarrow \exists \mathbf{t}_1 : \mathbf{t} < \mathbf{t}_1 \;\wedge\; l_1 \sqsubseteq^+ |_a \mathbf{t}_1 \;\wedge\; \neg \mathbf{t} < l_1 \;\wedge\; (l_1 \text{ conn}_s\, \mathbf{t} \;\vee\; (l_1 \text{ conn}_a\, \mathbf{t} \;\wedge\; \neg \exists \mathbf{t}_2 : l_1 \text{ assigned } \mathbf{t}_2) \;\vee\; l_1 \text{ assigned } \mathbf{t})$

For the implementation of connectedness we make use of the insight recorded in theorem 2 that membership to extended ambiguity domains can be determined along conjoining chains. Let AD(l_2) be the ambiguity domain by virtue of which l_1 is connected with $\mathbf{t}$, i.e. l_2 is the leaf that l_1 shares with the first link l_3 of the chain from l_1 to $\mathbf{t}$. By theorem 2 l_3 is also subordinate to $\mathbf{t}$ and if we induce over the length of chains[12] this subordination relation is computed ($l_3 = \mathbf{t} \vee l_3 \ll \mathbf{t}$). Two cases must be distinguished: Either l_2 is subordinate to l_1 over subordination links alone (connectedness over subordination) or there is at least one accessibility link in the series (connectedness over accessibility).

Definition 19 (connectedness over subordination)
$l_1 \text{ conn}_s\, \mathbf{t} \leftrightarrow \exists l_2 : \text{leaf}_{\sqsubseteq_a}(l_2) \wedge l_2 \sqsubseteq^* l_1 \wedge (l_2 \sqsubseteq_a^* \mathbf{t} \vee (\exists l_3 : l_2 \sqsubseteq^* l_3 \wedge l_3 \ll \mathbf{t}))$

Definition 20 (connectedness over accessibility)
$l_1 \text{ conn}_a\, \mathbf{t} \leftrightarrow \exists l_2 : \text{leaf}_{\sqsubseteq_a}(l_2) \;\wedge\; l_2 \sqsubseteq_a^* l_1 \;\wedge\; \neg l_2 \sqsubseteq^* l_1 \;\wedge\; (l_2 \sqsubseteq_a^* \mathbf{t} \vee \exists l_3 : l_2 \sqsubseteq^* l_3 \;\wedge\; l_3 \ll \mathbf{t})$

If l_1 is connected with $\mathbf{t}$ over accessibility links and $\mathbf{t}$ is the nuclear scope of a suitable operator, there is the further option of assigning l_1 to the operator's restriction (which realizes intermediate accommodation). If this option is taken we record it with a relation *assigned*.

[12] As no label occurs twice in a chain all conjoining chains are finite.

Definition 21 (assignment to restriction)
$\mathbf{l}_1$ assigned $\mathbf{t}_1 \rightarrow \exists \mathbf{t} : \mathbf{t}_1 \Rightarrow \mathbf{t} \ \wedge \ \mathbf{l}_1$ conn$_a\mathbf{t}$

The full algorithm for enumerating the readings of a UDRS enriched with constraints from anaphora resolution is given in the appendix. It can also be used as a consistency checker for UDRSs. If it produces no solution but simply fails, the UDRS is inconsistent.

7 Conclusion

The proposed algorithm disambiguates UDRSs with constraints from anaphora resolution and handles accommodation in the way of van der Sandt (1992). It can also be used as a consistency checker for UDRSs. Some new well-formedness conditions for UDRSs are directly applied in the algorithm. They can not only be used for complete disambiguation but also for an early detection of inconsistency in case of partial disambiguation or for supplying a set of discourse referents accessible in some of the readings. Van der Sandt (1992, 365) requires that anaphoric DRSs may only be resolved if they do not contain anaphors themselves. In this approach no such requirement is necessary. A theorem prover (Reyle, 1995), (König-Baumer and Reyle, 1996) is used for matching anaphoric and antecedent content in presupposition binding. Embedded anaphors must only be resolved if the theorem prover discovers it cannot work without knowledge about the antecedent. Van der Sandt (1992, 362,367) also specifies some further criteria for accommodation sites, viz. consistency, informativity, and preference for global accommodation. The latter constraint can only be tested if the whole projection line is known, in contrast to the first two criteria. Since the algorithm works top-down, the projection lines of presupposed sub-DRSs have already been determined when they are met. Generation of readings is stopped as soon as the algorithm finds out that the readings lead to inconsistent or uninformative DRSs or in case a higher accommodation level for a presupposition has been found in some previously generated reading. Further work might be to extend the coverage to plural noun phrase ambiguities. Most interesting is here the possibility to surmount quantifier boundaries by set formation (called abstraction in Kamp and Reyle (1993)). Another extension to be envisaged is the treatment of further types of ambiguities such as collective and distributive readings (Frank and Reyle, 1995) and syntactic ambiguity in general (Schiehlen, 1996).

References

Hiyan Alshawi. 1992. *The Core Language Engine.* MIT Press, Cambridge, MA, USA.

Rolf Backofen, James Rogers, and K. Vijay-Shanker. 1995. A First-Order Axiomatization of the Theory of Finite Trees. *Journal of Logic, Language and Information*, 4(1):5–39, April.

Johan Bos. 1995. Predicate Logic Unplugged. In *Proceedings of the 10th Amsterdam Colloquium*, pages 133–142, Amsterdam, Holland, December. ILLC/Department of Philosophy, University of Amsterdam. also appeared as Verbmobil-Report 103.

Robin Cooper. 1983. *Quantification and Syntactic Theory.* Reidel, Dordrecht, Holland.

Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. 1995. Transfer and Minimal Recursion Semantics. In *Proceedings of the 6th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI '95)*, Leuven, Belgium.

Michael Dorna and Martin C. Emele. 1996. Semantic-Based Transfer. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*, Copenhagen, Denmark.

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

Anette Frank and Uwe Reyle. 1995. Principle Based Semantics for HPSG. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL '95)*, pages 9–16, Dublin, Ireland, March.

Jerry R. Hobbs and Stuart M. Shieber. 1987. An Algorithm for Generating Quantifier Scopings. *Computational Linguistics*, 13:47–63, June.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic: An Introduction to Modeltheoretic Semantics of Natural Language.* Kluwer Academic Publishers, Dordrecht, Holland.

Esther König-Baumer and Uwe Reyle. 1996. A General Reasoning Scheme for Underspecified Representations. In Hans Jürgen Ohlbach and Uwe Reyle, editors, *Logic and its Applications. Festschrift for Dov Gabbay.* Kluwer Academic Publishers. to appear.

Massimo Poesio. 1995. Semantic Ambiguity and Perceived Ambiguity. HCRC Research Paper 68, Human Communications Research Centre, University of Edinburgh, Edinburgh, UK, May.

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

Uwe Reyle. 1995. On Reasoning with Ambiguities. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL '95)*, pages 1–8, Dublin, Ireland.

Michael Schiehlen. 1996. Semantic Construction from Parse Forests. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*, Copenhagen, Denmark.

Rob A. van der Sandt. 1992. Presupposition Projection as Anaphora Resolution. *Journal of Semantics*, 9(4):333–377.

Jan van Eijck and Jan Jaspars, 1996. *Underspecification and Reasoning*, pages 268–289. Number D15 in FraCaS Deliverable. FraCaS, LRE 62-051, University of Edinburgh, January.

W.A. Woods. 1978. Semantics and Quantification in Natural Language Question Answering. In M.C. Yovits, editor, *Advances in Computers*, volume 17. Academic Press, New York, New York.

A Description of the Algorithm

The algorithm traverses the UDRS graph from top down beginning with the root label $\mathbf{t}_\top$. In the following description, sets gained by abstraction are abbreviated by omission of the label abstracted over (e.g. $\{[\mathtt{t} \leq]\} = \{\mathtt{l} | [\mathtt{t} \leq \mathtt{l}]\}$). The algorithm starts with a call to `disamb(t⊤)`. It is nondeterministic in the sense that it yields several solutions (i.e. UDRS readings or DRSs) on backtracking. The individual steps are explained in turn below.

$\mathrm{DRS} := \mathrm{DRS}(\mathbf{t}_\top)$
$\{\ll \mathbf{t}_\top\} := \emptyset$
$\{\mathbf{t}_\top \prec^+\} := \emptyset$
$\{\mathbf{t}_\top\ \mathrm{acc}\} := \emptyset$

`disamb( t )` $\leftarrow$
(1) $\{\leq |_a \mathbf{t}\} := \{\sqsubseteq^+ |_a \mathbf{t}\} \cup \{\ll \mathbf{t}\}$,
(2) $\{\sim \mathbf{t}\} :\subseteq \{\leq |_a \mathbf{t}\}$,
(3) $\{< |_a \mathbf{t}\} := \{\leq |_a \mathbf{t}\} \backslash \{\sim \mathbf{t}\}$,

(4) $\{\mathbf{t} \leq\} := \{\mathbf{t} \prec^+\} \cup \{\mathbf{t}\} \cup \{\sim \mathbf{t}\}$,

(5) $\forall \mathrm{l}_1 \in \{\sim \mathbf{t}\} : ($
$\quad \{[\mathrm{l}_1 \leq]\} \subseteq \{\mathbf{t} \leq\} \ \wedge$
$\quad \{[\mathrm{l}_1 \text{ acc}]\} \subseteq \{\mathbf{t} \leq\} \cup \{\mathbf{t} \text{ acc}\} \ )$

(6) $\mathrm{DRS}(\mathbf{t}) := \{ \ \mathrm{Cond}_{\mathbf{t}_i \rightarrow \mathrm{DRS}(\mathbf{t}_i)} | \exists \mathrm{l}_1 : \mathrm{l}_1 \in \{\mathbf{t}\} \cup \{\sim \mathbf{t}\} \ \wedge \ [\mathrm{l}_1 : \mathrm{Cond}]\}$,

(7) $\{\prec \mathbf{t}\} := \{[\prec \mathrm{l}_1]$, where $\mathrm{l}_1 \in \{\mathbf{t}\} \cup \{\sim \mathbf{t}\}\}$,

(8) $\forall \mathrm{l}_1 \in \{\prec \mathbf{t})\} :$
$\quad \{\mathrm{l}_1 \prec^+\} := \{\mathbf{t} \leq\}$,
$\quad \{\mathrm{l}_1 \text{ acc}\} := \{\mathbf{t} \text{ acc}\} \cup (\text{if } \exists \mathrm{l}_2 : \mathrm{l}_2 \Rightarrow \mathrm{l}_1) \ \{\mathrm{l}_2\} \cup \{\sim \mathrm{l}_2\}$,

(9) $\forall \mathbf{t}_1 \in \{\prec \mathbf{t}\} \forall \mathrm{l}_1 \in \{< |_a \mathbf{t}\} :$
$\quad \mathrm{DLL}(\mathbf{t}_1) := (\{\sqsubseteq_a^* \mathbf{t}_1\} \cup \{\sqsubseteq^* \mathrm{l}_2$, where $\mathrm{l}_2 \ll \mathbf{t}_1\}) \cap \mathrm{leaf}_{\sqsubseteq_a}$

(9a) $\quad \{\sqsubseteq^* \mathrm{l}_1\} \cap \mathrm{DLL}(\mathbf{t}_1) \neq \emptyset \rightarrow$ add l_1 to $\{\ll \mathbf{t}_1\}$,

(9b) $\quad \{\sqsubseteq^* \mathrm{l}_1\} \cap \mathrm{DLL}(\mathbf{t}_1) = \emptyset \ \wedge \ \{\sqsubseteq_a^* \mathrm{l}_1\} \cap \mathrm{DLL}(\mathbf{t}_1) \neq \emptyset \rightarrow$
$\quad$ either add l_1 to $\{\ll \mathbf{t}_1\}$ or if $\exists \mathbf{t}_2 : \mathbf{t}_2 \Rightarrow \mathbf{t}_1$, add l_1 to $\{\ll \mathbf{t}_2\}$,

until no more l_1 can be added to $\{\ll \mathbf{t}_1\}$.

(10) $\forall \mathbf{t}_1, \mathbf{t}_2 \in \{\prec \mathbf{t}\}, \mathbf{t}_1 \neq \mathbf{t}_2 : \{\ll \mathbf{t}_1\} \cap \{\ll \mathbf{t}_2\} = \emptyset$

(11) $\forall \mathrm{l}_1 \in \{< |_a \mathbf{t}\} \exists \mathbf{t}_1 \in \{\prec \mathbf{t}\} : \mathrm{l}_1 \in \{\ll \mathbf{t}_1\}$

(12) $\forall \mathbf{t}_1 \in \{\prec \mathbf{t}\} :$ `disamb(` $\mathbf{t}_1$ `)`

(1) First we determine the labels that could be equal to the tree label $\mathbf{t}$. By def. 17 we have to look at overt and computed subordination. The set $\{ \mathrm{l}_1 \mid \mathrm{l}_1 \ll \mathbf{t} \}$ is computed at the mother label of $\mathbf{t}$ and is known by top-down assumption.

(2) Next we arbitrarily choose some of the potentially equal labels ($\leq |_a$) to also be equal in the currently generated reading ($\sim$). This step is the main source of nondeterminism ($\subseteq$ instead of $=$) in the algorithm. So this is the point to invoke additional heuristics.

(3) All labels under $\mathbf{t}$ not equal to $\mathbf{t}$ are strictly subordinate to $\mathbf{t}$ ($< |_a$). These labels must be set equal to some tree label below (see def. 18).

(4) With scope equivalence we can define the tree order of DRS subordination ($\leq$). Since the disambiguation process works from top down, the part of the graph above $\mathbf{t}$ ($\prec^+$) is always fully specified. This step merely computes a reflexive order ($\leq$) out of an irreflexive one ($<$) and equivalence ($=$).

(5) The two constraints check the overt constraint information ($[\leq]$ and [acc]) against the tree being built. The set $\{\mathrm{l}_1 | \mathbf{t} < \mathrm{l}_1\}$ is known according to the top-down assumption[13].

(6) In order to generate a DRS from a UDRS it suffices to determine the relation of scope equivalence ($\sim$) which assigns to each ambiguity label a tree label. With scope equivalence, we can define a function DRS(.) from tree labels to sub-DRSs. $\mathrm{Cond}_{a \rightarrow b}$ stands for a version of Cond where all a's are replaced by b's. The sub-DRS associated with the top label ($\mathrm{DRS}(\mathbf{t}_\top)$) is the fully specified DRS that expresses the currently generated reading.

(7) This step determines all the daughter labels of the tree label $\mathbf{t}$.

(8) Here the daughter labels $\mathbf{l}_1$ are assigned their superordinate labels ($\prec^+$) and the labels that are accessible from them but not superordinate (acc). The latter relation is computed according to the formula $\mathbf{t}$ acc $\mathbf{l}_1 \leftrightarrow \exists \mathbf{t}_1, \mathbf{t}_2 : \mathbf{l}_1 \sim \mathbf{t}_1 \;\wedge\; \mathbf{t}_1 \Rightarrow \mathbf{t}_2 \;\wedge\; \mathbf{t} \leq \mathbf{t}_2$ (see def. 11).

(9) This step distributes the labels $\mathbf{l}_1$ below $\mathbf{t}$ among $\mathbf{t}$'s daughter labels $\mathbf{t}_1$. The sets DLL are the sets of leaves that are syntactically linked to the respective daughter labels ($\sqsubseteq_a$ is defined in def. 9). Note that these sets grow as more and more labels $\mathbf{l}_1$ are distributed. Fortunately conjoining chains are finite.

(9a) If $\mathbf{l}_1$ is connected with $\mathbf{t}_1$ over subordination links only (def. 19) $\mathbf{l}_1$ is subordinate or equal to $\mathbf{t}_1$.

(9b) If $\mathbf{l}_1$ is connected with $\mathbf{t}_1$ over at least one accessibility link (def. 20) $\mathbf{l}_1$ is either subordinate to $\mathbf{t}_1$ itself or, if $\mathbf{t}_1$ is nuclear scope, to the corresponding restriction.

(10) If $\mathbf{l}_1$ shares ambiguity domains with two daughter labels $\mathbf{t}_1$ and $\mathbf{t}_2$ of $\mathbf{t}$, then $\mathbf{l}_1 \geq \mathbf{t}$, since theorem 1 demands that $\mathbf{l}_1$ be on a branch with both $\mathbf{t}_1$ and $\mathbf{t}_2$.

(11) If $\mathbf{l}_1$ shares ambiguity domains with a label $\mathbf{t}$ but with none of $\mathbf{t}$'s daughters, then $\mathbf{l}_1 \geq \mathbf{t}$, see Connectedness (constraint 2). Checking (10) and (11) runs interleaved with (9).

(12) Top down recursion step.

Since all $\sqsubseteq$ relations are recorded (in step (1) and (7) by definition 5) the root axiom guarantees completeness: Every label in the graph is visited. Inconsistency results when step (5) requires that a label $\mathbf{l}_1$ be subordinate to $\mathbf{t}$ while steps (10) or (11) necessitate $\mathbf{l}_1$'s being equal to $\mathbf{t}$.

[13]In the implementation checking constraint (5) and determining the set of scope equivalent labels (2) is an interleaved process. This is possible since the subordinated labels $\mathbf{l}_1$ can be ordered according to the following metric: Assign to each subordinated label $\mathbf{l}_1$ the length n of the longest chain of $[\leq] \cup$ acc-links involving only labels $\leq |_a \mathbf{t}$. The number n is always finite since a UDRS is an acyclic graph. Check the labels with least n first.