Labeled Resolution for Discourse Semantics

Christof Monz and Maarten de Rijke
ILLC, University of Amsterdam.
Plantage Muidergracht 24, 1018 TV Amsterdam
E-mail: {christof,mdr}@wins.uva.nl

Abstract

This paper concerns the application of resolution theorem proving to natural language semantics. We focus on the problem of efficient pronoun binding in a discourse, trying to circumvent the enormous computational complexity triggered by natural language ambiguities like pronoun binding. Automated deduction steps and binding of pronouns are interleaved with the effect that only pronouns that are used during a derivation are bound to a possible antecedent. Labels are used to encode structural information which is essential to define admissible bindings. It turns out that the notion of pronoun binding can be reduced to the notion of labeled unification.

1 Introduction

Unlike formal languages natural languages are ambiguous. Ambiguity is especially problematic for automated reasoning, because it is not clear which of the possibly conflicting readings the deduction method has to consider if an ambiguous expression occurs within the derivation. Is it to enough to consider only one reading, or do all of them have to be considered? Should a theorem prover work only on totally disambiguated expressions? Total disambiguation results in an explosion of readings, because of the multiplicative behavior of ambiguity. On the other hand, to prove a conclusion φ from a set of premises Γ it may be enough to use only premises from a subset Δ of Γ, and it may be sufficient, and much more efficient, to disambiguate only Δ instead of the whole set of premises Γ. In general, we do not know in advance which subset of premises might be enough to derive a certain conclusion, but during a derivation often certain (safe) strategies may be applied that prevent some premises from being used since they cannot lead to the conclusion, anyway. Common strategies to constrain the search space in resolution deduction are e.g., the *set-of-support strategy* and *ordered resolution.* Our goal is to constrain the set of premises that have to be disambiguated by interleaving deduction and disambiguation.

Roughly speaking, premises are only disambiguated if they are used by a deduction rule.

Reasoning with ambiguous information is still a matter of ongoing research, see for instance [vD96, vEJ96, Jas97, MdR98], and this paper does not try to give a general answer. Instead, we focus on a particular instance of ambiguity, namely the question of pronoun binding.[1] As exemplified by (1) below.

(1) A man saw a boy. He whistled.

(2) a. A man$_i$ saw a boy. He$_i$ whistled.

b. A man saw a boy$_i$. He$_i$ whistled.

Often, there are lots of possibilities and it is not clear which one to choose. The pronoun *he* in the short discourse in (1) can be resolved in two ways as given in (2), where co-indexation indicates identity. For some cases heuristics are applicable which prefer certain bindings to others, but at present there is no approach making use of heuristics which is general enough to cover all problems.

The main topic of this papers is not to deal with pronoun binding as such, but to give an efficient deduction system for natural language discourses and pronoun binding is only part of the problem. Our strategy is to try and deduce something from a set of premises and bind pronouns only where necessary.

The quest for efficiency suggests that we try to use resolution based methods, but here a problem arises. Most efficient reasoning systems presuppose that their input is in a normal form, e.g., conjunctive normal form (CNF) for resolution deduction. In the process, the original structure of the discourse, which is important for pronoun binding, is totally distorted. In contrast, reasoning systems that do preserve the original structure like tableaux or natural deduction are less efficient. How do code structural information in a different in a resolution based approach? In this paper we will see that *labels* allow us to do so in an appropriate way.

The rest of the paper is structured as follows. Section 2 provides some rudimentary background in dynamic semantics and explains what kind of structural information is necessary to properly restrict pronoun binding. Besides that, the basics of resolution deduction are introduced. Section 3 discusses some of the problems of the (standard) resolution method when applied to natural language. The method of labeled unification and labeled resolution is presented to overcome these problems. Section 4 briefly relates our work to some other approaches to pronoun binding. Section 5 provides some conclusions and prospects for further work.

[1]Throughout this paper we use the term *binding* to express the referential identification of a pronoun and another referential expression occurring in the discourse. Common terms are also *co-indexation* or *pronoun resolution*. We especially did not use *pronoun resolution* to avoid confusion with resolution as a deduction principle.

2 Background

Before we turn to our method of labeled resolution deduction and its applications to discourse semantics, we briefly present the idea of dynamic semantics. The second subsection shortly explains the classic resolution method for (static) first-order logic.

2.1 Dynamic Semantics

Dynamic semantics [Kam81, Hei82] allows to give a perspicuous solution to some problems involving pronoun binding. We consider a combination of Discourse Representation Theory (DRT) [Kam81, KR93], where co-indexation of pronouns is not presupposed, and Dynamic Predicate Logic (DPL) [GS91]. In (3) a semantic representation of the short discourse in (1) is given, where the pronoun *he* is represented by a free variable $\mathbf{u}$. Variables for pronouns are displayed in boldface and are of a different kind than regular variables. If we want to identify this variable with one of the variables bound by one of the existential quantifiers in the first sentence, we have to extend the scope of the existential quantification.

(3) $\exists x(man(x) \wedge \exists y(boy(y) \wedge see(x,y)))\ .\ whistle(\mathbf{u})$

The corresponding construction of a representation for the whole discourse basically consists of two steps:

(4) step 1: $\exists x(man(x) \wedge \exists y(boy(y) \wedge see(x,y))) \oplus whistle(\mathbf{u})$
step 2: $\exists x(man(x) \wedge \exists y(boy(y) \wedge see(x,y))) \wedge whistle(x)$

The $\oplus$ operator merges two formulas by putting them into a conjunction and binding variable pronouns, where we assume for simplicity that both formulas are variable disjoint. A free variable $\mathbf{u}$ is identified with a bound variable x if x is *accessible* from $\mathbf{u}$.

One of the main characteristics of dynamic semantics is the flexible scope of the existential quantifier. In classical logic the variable x occurs free after the second step in 4, but in dynamic semantics the existential quantifier can also bind variables occurring to the right-hand side of its traditional scope, as long as certain conditions hold. One such condition is that existential quantifiers occurring inside a negation cannot bind variables outside of the negation. The following example illustrates this.

(5) *John doesn't own a car$_i$. It$_i$ is in front of his house.

Both properties (a) existential quantifiers can bind variables occurring to the right-hand side of their traditional scope and (b) negations are barriers for dynamic binding, allow us to define the properties of the other logical connectives $\vee$, $\rightarrow$ and $\forall$.

(6)
$$\begin{array}{rcl} [\![\varphi \vee \psi]\!] & = & [\![\neg(\neg\varphi \wedge \neg\psi)]\!] \\ [\![\varphi \rightarrow \psi]\!] & = & [\![\neg(\varphi \wedge \neg\psi)]\!] \\ [\![\forall x\varphi]\!] & = & [\![\neg\exists x\neg\varphi]\!] \end{array}$$

Given these definitions we see that the disjunction is a barrier both internally and externally, the implication is a barrier externally but internally it allows for flexible binding, and the universal quantifier does not allow for external binding.

The ?-operator is a bit different, because it only binds its argument, but does not quantify over it. Actually, it is not necessary to have a special operator for pronouns, and we introduced it more for the sake of convenience. Unlike the existential quantifier, the ?-operator does not have the property of flexible binding. We get the following equivalence:

$$[\![\neg ?\mathbf{u}\varphi]\!] \quad = [\![?\mathbf{u}\neg\varphi]\!]$$

To define *accessibility* we can now say that a variable x is accessible from a pronoun **u** if no barrier occurs between the quantifier introducing x and $?\mathbf{u}$. The equations in (6) show that $\vee$, $\rightarrow$ and $\forall$ introduce barriers because of the way they are defined in terms of negation. This is exemplified by (7) below.

(7) *Every farmer owns a donkey. It is grey.

Unfortunately we do not have enough space to give a more detailed account of dynamic semantics, but we refer the reader to [Kam81, GS91].

2.2 The Resolution Method

The *resolution method* [Rob65] has become quite popular in automated theorem proving, because it is very efficient and it is easily augmentable by lots of strategies which restrict the search space, see e.g., [Lov78]. On the other hand, the resolution method has the disadvantage of presupposing that its input has to be in *clause form*, where clause form is the same as CNF but a disjunction is displayed as a set of literals (the *clause*) and the conjunction of disjunctions is a set of clauses. Probably the most attractive feature of resolution is that it has only one single inference rule, the resolution rule:

$$\frac{C \cup \{\neg \mathrm{P}_1, \ldots, \neg \mathrm{P}_n\} \quad D \cup \{\mathrm{Q}_1, \ldots, \mathrm{Q}_m\}}{(C \cup D\pi)\sigma} \ (\mathit{res})$$

where
- $\mathrm{Q}_1, \ldots, \mathrm{Q}_m$ are atomic
- π is a substitution such that $C \cup \{\neg \mathrm{P}_1, \ldots, \neg \mathrm{P}_n\}$ and $D\pi \cup \{\mathrm{Q}_1\pi, \ldots, \mathrm{Q}_m\pi\}$ are variable disjoint
- σ is the most general unifier of $\{\mathrm{P}_1, \ldots, \mathrm{P}_n, \mathrm{Q}_1\pi, \ldots, \mathrm{Q}_m\pi\}$

To prove that $\Gamma \models \varphi$ holds we transform $(\bigwedge \Gamma) \wedge \neg\varphi$ in clause form and try to derive a contradiction (the empty clause) from it by using the resolution rule.

For a comprehensive introduction to the resolution method see for instance [Lei97, Lov78].

3 Dynamic Resolution

Applying the classical resolution method to a dynamic semantics causes problems. Below we will first discuss some of them and then see how we have to design our dynamic resolution method to overcome these problems.

3.1 Adapting the Resolution Method

There are mainly two problems that we have to find a solution for. First, transforming formulas to clause form causes a loss of structural information. Therefore, it is sometimes impossible to distinguish between variables that can serve as antecedents for a pronoun and variables than can not. The second problem concerns the duplication of literals which may occur during clause from transformation and the assumption of the resolution method that clauses are variable disjoint. Although the same pronoun may have two occurrences in different clauses, we do not want them to be bound by different antecedents.

Turning to the first problem, e.g. in (8), the pronoun $\mathbf{u}$ cannot be bound by the existential quantifier, whereas the pronoun $\mathbf{z}$ can be bound by it.

(8) a. Every farmer who owns a donkey beats it. It suffers.

b. $\forall x(f(x) \wedge \exists y(d(y) \wedge o(x,y)) \rightarrow ?\mathbf{z}\, b(x,\mathbf{z}))) \wedge ?\mathbf{u}\, s(\mathbf{u})$

(9) $\{\, \{\neg f(x), \neg d(y), \neg o(x,y), b(x,\mathbf{z})\},\ \{s(\mathbf{u})\}\,\}$

How can we tell which identifications are allowed by looking at the corresponding clause form in (9)? How do we know whether a variable or skolem function is accessible?

We use *labels* to carry the information about accessible variables. Each pronoun variable is annotated with a label that indicates the set of accessible variables. Besides the set of first-order or proper variables (*VAR*), first-order formulas (*FORM*), and pronoun variables (*PVAR*), we are going to introduce the sets of labeled pronoun variables (*LPVAR*) and labeled formulas (*LFORM*). Labeled pronoun variables are of the form $V : \mathbf{u}$, where $V \subseteq \mathit{VAR}$ and $\mathbf{u}$ is a pronoun variable. The set of labeled formulas is the set of first-order formulas plus formulas containing labeled pronoun variables.

To see which variables inside of a formula φ can serve as antecedents for pronouns, [GS91] introduced the function AQV which returns the set of *actively quantifying variables* when applied to φ.

Definition 1 (Actively Quantifying Variables) Let *FORM* be the set of classical first-order formulas and *VAR* the set of first-order variables. The function $\mathsf{AQV} : \mathit{FORM} \to \mathit{POW}(\mathit{VAR})$ is defined recursively:

$$\begin{array}{rcl}
\mathsf{AQV}(R(x_1 \ldots x_n)) & = & \emptyset \\
\mathsf{AQV}(\neg\varphi) & = & \emptyset \\
\mathsf{AQV}(\varphi \wedge \psi) & = & \mathsf{AQV}(\varphi) \cup \mathsf{AQV}(\psi) \\
\mathsf{AQV}(\varphi \to \psi) & = & \emptyset \\
\mathsf{AQV}(\varphi \vee \psi) & = & \emptyset \\
\mathsf{AQV}(\forall x\varphi) & = & \emptyset \\
\mathsf{AQV}(\exists x\varphi) & = & \mathsf{AQV}(\varphi) \cup \{x\} \\
\mathsf{AQV}(?\mathbf{u}\varphi) & = & \mathsf{AQV}(\varphi)
\end{array}$$

Using the above definition we can now define the notion of accessible variables.

Definition 2 (Annotation with Accessible Variables) To annotate $\mathbf{u}$ in $?\mathbf{u}\psi$, we drop the binding operator $?\mathbf{u}$ and substitute all occurrences of the pronoun variable in ψ by its annotated counterpart. The annotation function $\mathsf{annot} : \mathit{VAR} \times \mathit{FORM} \to \mathit{LFORM}$ is defined recursively, where $V \subseteq \mathit{VAR}$:

$$\begin{array}{rcl}
\mathsf{annot}(V, R(x_1 \ldots x_n)) & = & R(x_1 \ldots x_n) \\
\mathsf{annot}(V, \neg\varphi) & = & \neg\mathsf{annot}(V, \varphi) \\
\mathsf{annot}(V, \varphi \wedge \psi) & = & \mathsf{annot}(V, \varphi) \wedge \mathsf{annot}(V \cup \mathsf{AQV}(\varphi), \psi) \\
\mathsf{annot}(V, \varphi \to \psi) & = & \mathsf{annot}(V, \varphi) \to \mathsf{annot}(V \cup \mathsf{AQV}(\varphi), \psi) \\
\mathsf{annot}(V, \varphi \vee \psi) & = & \mathsf{annot}(V, \varphi) \vee \mathsf{annot}(V, \psi) \\
\mathsf{annot}(V, \forall x\varphi) & = & \forall x\ \mathsf{annot}(V \cup \{x\}, \varphi) \\
\mathsf{annot}(V, \exists x\varphi) & = & \exists x\ \mathsf{annot}(V \cup \{x\}, \varphi) \\
\mathsf{annot}(V, ?\mathbf{u}\varphi) & = & \mathsf{annot}(V, \varphi[\mathbf{u}/V : \mathbf{u}])
\end{array}$$

The actual annotation takes place in the last case, where the pronoun is substituted. The other cases thread the actively quantifying variables through the formula. To annotate a whole discourse $\varphi_1 \wedge \cdots \wedge \varphi_n$, the variable parameter of annot is initialized with $\emptyset$, $\mathsf{annot}(\emptyset, \varphi_1 \wedge \cdots \wedge \varphi_n)$. Alternatively, we say that a variable x is *accessible from* a pronoun $\mathbf{u}$ iff x is element of the set of the accessible variables of $\mathbf{u}$. It is important to note that the first parameter (the variable parameter) of the annot-function is structure-shared when applied to different subformulas. This is mainly for two reasons: first, to avoid representational redundancy, as the set of accessible variables can become quite large, and second, it allows to handle global instantiation efficiently.

Reconsider the last example, *every farmer who owns a donkey beats it. It suffers.* Applying annotation yields:

$$\begin{array}{l}
\mathsf{annot}(\emptyset, \forall x(f(x) \wedge \exists y(d(y) \wedge o(x,y)) \to ?\mathbf{z}\, b(x, \mathbf{z})) \wedge ?\mathbf{u}\, s(\mathbf{u})) \\
\quad = \forall x(f(x) \wedge \exists y(d(y) \wedge o(x,y)) \to b(x, \{x,y\} : \mathbf{z}))) \wedge s(\emptyset : \mathbf{u})
\end{array}$$

Applying clause form transformation to the annotated formulas yields:

(10) $\{\, \{\neg f(x), \neg d(y), \neg o(x,y), b(x, \{x,y\} : \mathbf{z})\}, \{s(\emptyset : \mathbf{u})\}\, \}$

Here, we can also see that (8.a) is not well-formed because there are no accessible pronouns for the second pronoun *it*, i.e., the label of $\mathbf{u}$ is the empty set.

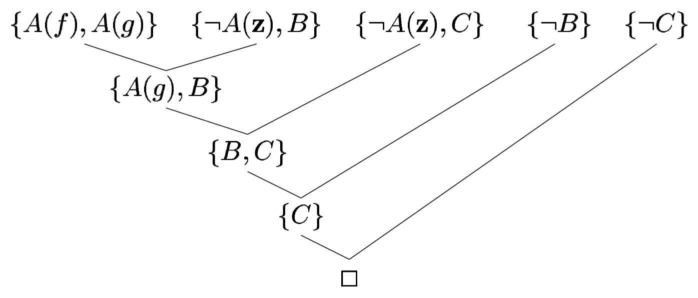
Although clause form transformation does not manipulate the labels directly, it does in an indirect way, because skolemization substitutes variables by skolem terms. Therefore, the substitution has to be applied to the labels, which carry a set of variables, also.

Now we turn to the second problem: how do we make sure that the same pronoun, occurring in different clauses, is bound to the same antecedent? As we said earlier, we do not want to assume pronouns to be bound in a set of premises when we apply resolution. The reason is that pronoun binding is highly ambiguous and often it is not necessary to bind all pronouns in a set of premises to derive a certain conclusion from it. Another issue, which we briefly hinted at in Section 2, is that pronouns should be treated as free variables of a special kind, not to be dealt with in the same manner as universally quantified variables (which also happen to be represented by free variables). This is illustrated by the following example, which shows an invalid entailment.

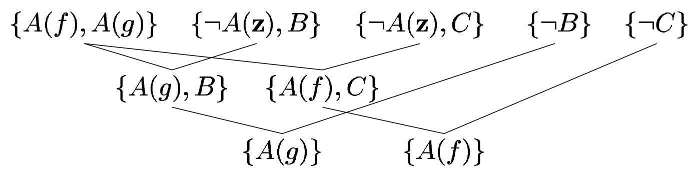
(11) a. $\exists x \exists y((A(x) \vee A(y)) \wedge (?\mathbf{z}A(\mathbf{z}) \rightarrow (B \wedge C))) \not\models B \vee C$

b. $\{\, \{A(f), A(g)\},\ \{\neg A(\mathbf{z}), B\},\ \{\neg A(\mathbf{z}), C\},\ \{\neg B\},\ \{\neg C\}\,\}$

The transformation in (11) causes a duplication of the literal $\neg A(\mathbf{z})$, and we have to make sure that the pronoun is instantiated the same way in both cases.

(12)


In (12) $\mathbf{z}$ is instantiated with f in the first resolution step and then with g in the second. The resolution rule as it was stated in the preceding section assumes that clauses to be resolved are variable disjoint. We have to modify the resolution rule such that the same pronoun variable is allowed to occur in both clauses. Additionally, the instantiation of a pronoun variable for constructing the most general unifier in a resolution step is applied globally, i.e. to all clauses.

(13)


Global instantiation correctly prevents us from deriving a contradiction in (13).

3.2 Labeled Resolution

Unification is a fundamental technique in the resolution method. Since we are also dealing with labeled variables, we have to think how the unification mechanism has to be adapted. In the course of this subsection, it will turn out that pronoun binding can be reduced to unification.

3.2.1 Labeled Unification

We use the unification algorithm of Martelli and Montanari [MM82] as a basis and adapt it in such a way that it can deal with labeled pronoun variables.

What does it mean to unify a set of equations $E = \{s_1 \approx t_1, \ldots, s_n \approx t_n\}$, where s_i or t_i can also be a labeled pronoun variable? We have to distinguish three possible cases: (i) neither s_i nor t_i is a labeled pronoun variable, then labeled unification and normal unification are the same thing, (ii) one of them is a pronoun and the other is not, and (iii) both are pronouns. Case (ii) is the normal pronoun binding, where a try to identify pronoun with a proper variable. Case (iii) is not an instance of pronoun binding, but an identification of two pronouns, i.e., whatever is the antecedent of the first pronoun, it is also the antecedent of the other one.

Definition 3 (Labeled Unifier) A substitution σ is a *labeled unifier* or *unifier** of a set of equations $E = \{s_1 \approx t_1, \ldots, s_n \approx t_n\}$ iff

1. $s_1\sigma = t_1\sigma, \ldots, s_n\sigma = t_n\sigma$
2. if $(V\!:\!\mathbf{u})\sigma = t$ and $t \notin \mathit{LPVAR}$ then $t \in V$
3. if $(V\!:\!\mathbf{u})\sigma = V'\!:\!\mathbf{v}$ then $V' \subseteq V$

We use $\approx$ to express equality in our object language, whereas $=$ denotes equality in the meta language.

Condition 1 is the normal condition of unifiability, namely that the terms of an equation have to be identical after substitution. The second condition says that unifiers have to obey accessibility, for instance $\sigma := [\{x, f\}\!:\!\mathbf{u}/g]$ is not a unifier of $\{\{x, f\}\!:\!\mathbf{u} \approx g\}$, because g is not accessible from $\mathbf{u}$, as $g \notin \{x, f\}$. To ensure that identification of pronouns always restricts the set of accessible antecedents, we need condition 3.

Definition 4 (Most General Labeled Unifier) A labeled unifier σ of a set of equations $E = \{s_1 \approx t_1, \ldots, s_n \approx t_n\}$ is the *most general labeled unifier* or *mgu** of E iff

1. if θ is a unifier* of E then there is substitution τ such that $\theta = \sigma\tau$
2. if $(V\!:\!\mathbf{u})\sigma = V_1\!:\!\mathbf{v}$, $(V\!:\!\mathbf{u})\theta = V_2\!:\!\mathbf{v}$, $V_1, V_2 \subseteq V$, and $V_1, V_2 \neq \emptyset$ then $V_2 \subseteq V_1$

Again, the first condition is standard in regular unification. Condition 2 says that the most general unifier* has to restrict the set of accessible antecedents as little as possible when identifying pronouns. To unify $V_1 : \mathbf{u}$ and $V_2 : \mathbf{v}$ it suffices to take any non-empty subset of the intersection of V_1 and V_2, but this fact may prohibit some antecedents from being accessible, although they are in fact accessible for both pronouns.

Definition 5 (The Labeled Unification Algorithm) The unification function unify^* is first applied to a pair of atoms, and then it tries to unify the set of corresponding argument pairs. The algorithm terminates successfully if it did not terminate with failure and no further equations are applicable.

1. $\mathsf{unify}^*(R(s_1...s_n), R(t_1..t_n))$
   $= \mathsf{unify}^*(\{s_1 \approx t_1...s_n \approx t_n\})$
2. $\mathsf{unify}^*(\{f(s_1...s_n) \approx f(t_1...t_n)\} \cup E)$
   $= \mathsf{unify}^*(\{s_1 \approx t_1...s_n \approx t_n\} \cup E)$
3. $\mathsf{unify}^*(\{f(s_1...s_n) \approx g(t_1...t_m)\} \cup E)$, $f \neq g$ or $n \neq m$
   $=$ terminate with failure
4. $\mathsf{unify}^*(\{x \approx x\} \cup E$
   $= \mathsf{unify}^*(E)$
5. $\mathsf{unify}^*(\{t \approx x\} \cup E)$, $t \notin \mathit{VAR}$
   $= \mathsf{unify}^*(\{x \approx t\} \cup E)$
6. $\mathsf{unify}^*(\{x \approx t\} \cup E)$, $x \neq t$, $t \notin \mathit{LPVAR}$, x in t
   $=$ terminate with failure
7. $\mathsf{unify}^*(\{x \approx t\} \cup E)$, $x \neq t$, $t \notin \mathit{LPVAR}$, x not in t, x in E
   $= \mathsf{unify}^*(\{x \approx t\} \cup E[x/t])$
8. $\mathsf{unify}^*(\{V : \mathbf{u} \approx t\} \cup E)$, $t \notin \mathit{LPVAR}$, $t \in V$, $V : \mathbf{u}$ in E
   $= \mathsf{unify}^*(\{V : \mathbf{u} \approx t\} \cup E[V : \mathbf{u}/t])$
9. $\mathsf{unify}^*(\{V_1 : \mathbf{u} \approx V_2 : \mathbf{v}\} \cup E)$, $V_1 \cap V_2 \neq \emptyset$, $V_1 \cap V_2 \subset V_2$
   $= \mathsf{unify}^*(\{V_1 : \mathbf{u} \approx V_1 \cap V_2 : \mathbf{v}, V_2 : \mathbf{v} \approx V_1 \cap V_2 : \mathbf{v}\}$
   $\cup E[V_1 : \mathbf{u}/V_1 \cap V_2 : \mathbf{v}, V_2 : \mathbf{v}/V_1 \cap V_2 : \mathbf{v}])$

The first equations of the algorithm are the same as in [MM82], except for additional side conditions which make sure that t is not a labeled variable. The interesting cases are 8 and 9. In 8 a pronoun is bound to an antecedent and in 9 two pronouns are identified, i.e., they have the same possible antecedents, namely those which are accessible for both of them.

Identification of pronouns underlies different constraints than binding a pronoun to a proper antecedent. To identify a pronoun $\mathbf{u}$ with another pronoun $\mathbf{v}$, it is not required that $\mathbf{u}$ is accessible from $\mathbf{v}$, or the other way around. But they can only be identified if they have at least one proper accessible antecedent in common.

(14) Buk is a poet. For every man there is a woman who hates him.
$\models$ There is a woman who hates him.

(15) $p(b) \wedge \forall x(w(x) \rightarrow \exists y(w(y) \wedge ?\mathbf{u}\, h(y, \mathbf{u})))$
$\models \exists z(w(z) \wedge ?\mathbf{v}\, h(z, \mathbf{v}))$

For instance, in (14) the conclusion is only valid if the first and the second occurrence of *him* are identified. In Section 2 it was said that universal quantification is a barrier for flexible binding, and therefore the second occurrence of *him* cannot be bound to the first one. On the other hand, both of them have a proper antecedent in common, namely the constant b representing the proper name *Buk*. In addition, the first occurrence of *him* has the variable x as an accessible antecedent, which is introduced by the universal quantification *every man*. If one wants to identify them, one has to take the intersection of both sets of accessible antecedents and hence drop x as a possible antecedent. Observe that identification of pronouns still leaves some space for underspecification, because the intersection of two pronouns does not have to be a singleton. Of course, identifying two pronouns, where more than one antecedent is accessible for both, forces them to be bound to the same element of the intersection. Both can be bound to one or the other element of the intersection, but it has to be the same one for both pronouns.

If the unification algorithm terminates successfully for a pair of literals P,Q, the solved set determines a substitution σ that is the mgu* of P,Q:

$$\sigma := \{s/t \mid s \approx t \in \mathsf{unify}^*(\mathrm{P}, \mathrm{Q})\}.$$

A set of equations $\{s_1 \approx t_1, \ldots, s_n \approx t_n\}$ is called *solved* iff

1. $s_i \in \mathit{VAR} \cup \mathit{LPVAR}$ and the s_i are pairwise disjoint
2. no s_i occurs in a term t_j $(1 \leq i, j \leq n)$.

Lemma 6 (Correctness of the Unification* Algorithm) *Let E be a set of equations and* $\mathsf{unify}^*(E) = E'$*, then*

(i) E is unifiable iff E' is unifiable**

(ii) σ is the mgu of E iff σ is the mgu* of E'*

Proof. (i) We have to show that actions 2, 4, 5, 7, 8, and 9 preserve unifiability*, when unify^* is applied to a unifiable* set E. For 2, 4, and 5, this is obvious. To show it for 7, note that $\tau := [x/t]$ is a unifier* of x and t. If σ is a unifier* of $\{x \approx t\} \cup E$ then σ is of the form $\tau\rho$. Because $\tau\tau = \tau$, it holds that $\sigma = \tau\rho = \tau\tau\rho = \tau\sigma$. Therefore σ unifies* $\{x \approx t\} \cup E$ iff σ unifies* $\{x \approx t\} \cup E[x/t]$. 8 is analogous to 7, plus the additional side condition that $t \in V$. The last case is 9. If $\{V_1 : \mathbf{u} \approx V_2 : \mathbf{v}\} \cup E$ is unifiable*, then it is with a unifier* σ of the form $\tau\rho$ with

$$\tau := [V_1 : \mathbf{u} / V_1 \cap V_2 : \mathbf{v}, V_2 : \mathbf{u} / V_1 \cap V_2 : \mathbf{v}].$$

Again, $\sigma = \tau\rho = \tau\tau\rho = \tau\sigma$ and then σ also unifies*

$$\{V_1 : \mathbf{u} \approx V_1 \cap V_2 : \mathbf{v}, V_2 : \mathbf{v} \approx V_1 \cap V_2 : \mathbf{v}\} \cup E[V_1 : \mathbf{u}/V_1 \cap V_2 : \mathbf{v}, V_2 : \mathbf{v}/V_1 \cap V_2 : \mathbf{v}].$$

(ii) The actions 2, 4, 5, 7, and 8 turn a set of equations into an equivalent one. For σ to be the mgu* of $\{V_1 : \mathbf{u} \approx V_2 : \mathbf{v}\} \cup E$ means according to our definition that σ has to be of the form $\tau\rho$, where

$$\tau := [V_1 : \mathbf{u}/V_1 \cap V_2 : \mathbf{v}, V_2 : \mathbf{u}/V_1 \cap V_2 : \mathbf{v}].$$

But then σ is also the mgu* of

$$\{V_1 : \mathbf{u} \approx V_1 \cap V_2 : \mathbf{v}, V_2 : \mathbf{v} \approx V_1 \cap V_2 : \mathbf{v}\} \cup E[V_1 : \mathbf{u}/V_1 \cap V_2 : \mathbf{v}, V_2 : \mathbf{v}/V_1 \cap V_2 : \mathbf{v}].$$

Lemma 7 (Termination of the Unification* Algorithm) *The unification* algorithm terminates for each finite set of equations.*

Proof. If rules 3 and 6 are applied, we are done. Otherwise, rule 7 can be applied only once, because after application the side condition is no longer fulfilled. In 9 it is presupposed that $V_1 \cap V_2$ is a proper subset of V_2; this ensures that an application of 9 really reduces the set of possible antecedents. Because 9 can be applied only a finite number of times, it can reintroduce a term $V : \mathbf{u}$ only finitely often, therefore rule 8 can also be applied only finitely many times. Rules 1, 5, and 6 are only applied once, and the number of possible applications of rule 2 is finite as well, because terms contain only finitely many symbols. Therefore all rules can be applied only finitely many times, and termination follows.

Proposition 8 (Total Correctness of the Unification* Algorithm) *The unification* algorithm computes for each finite set of equations E a solved set, that has the same mgu* as E in finitely many steps iff E is unifiable*.*

Proof. The fact that the unification* algorithm preserves unifiability* and that it terminates has been proven in Lemma 1 and Lemma 2, respectively. It remains to be shown that the set of equations computed by the algorithm is a solved set. In 7, 8, and 9, the left side of the equation is always substituted in E by the right side of the equation. If the left side is identical to the right side, the equation is erased by rule 4. Therefore, no left side of an equation occurs somewhere else.

3.2.2 The Resolution Method

Having defined labeled unification, it is straightforward to adapt the resolution principle. The only thing we have to change is to make sure that variable disjointness applies only to proper variables (elements of *VAR*). The function *VAR* returns the set of proper variables, when applied to a set of clauses Δ : $VAR(\Delta) = \{x \in VAR \mid x \text{ occurs in } \Delta\}$. The resolution rule accomplishing pronoun binding (res_p) is defined as follows:

$$\frac{C \cup \{\neg \mathrm{P}_1, \ldots, \neg \mathrm{P}_n\} \quad D \cup \{\mathrm{Q}_1, \ldots, \mathrm{Q}_m\}}{(C \cup D\pi)\sigma} \; (res_p)$$

where
- $\mathrm{Q}_1, \ldots, \mathrm{Q}_m$ are atomic
- π is a substitution such that $\mathit{VAR}(C \cup \{\neg \mathrm{P}_1, \ldots, \neg \mathrm{P}_n\}) \cap (\mathit{VAR}(D \cup \{\mathrm{Q}_1, \ldots, \mathrm{Q}_m\}))\pi = \emptyset$
- σ is the mgu* of $\{\mathrm{P}_1, \ldots, \mathrm{P}_n, \mathrm{Q}_1\pi, \ldots, \mathrm{Q}_m\pi\}$

3.2.3 An Example

We will only give a very short, and therefore very simple example of a labeled resolution derivation. We hope that it illustrates some of the aspects of labeled resolution mentioned before.

Consider example (14) again, here repeated as (16), where (17) is the corresponding semantic representation.

(16) Buk is a poet. For every man there is a woman who hates him.
$\models$ There is a woman who hates him.

(17) $p(b) \wedge \forall x(w(x) \rightarrow \exists y(w(y) \wedge ?\mathbf{u}\; h(y, \mathbf{u})))$
$\models \exists z(w(z) \wedge ?\mathbf{v}\; h(z, \mathbf{v}))$

Our proof algorithm prf consists of three steps:

1. annotate the conjunction of the premises and the negation of the conclusion;
2. apply clause form transformation; and
3. apply the resolution rule until a contradiction can be derived, or no new resolvents can be generated.

annotating (17):

$$\mathsf{annot}(\emptyset, p(b) \wedge \forall x(w(x) \rightarrow \exists y(w(y) \wedge ?\mathbf{u}\; h(y, \mathbf{u}))) \wedge \neg\exists z(w(z) \wedge ?\mathbf{v}\; h(z, \mathbf{v}))) =$$
$$p(b) \wedge \forall x(w(x) \rightarrow \exists y(w(y) \wedge h(y, \{b, x\} : \mathbf{u}))) \wedge \neg\exists z(w(z) \wedge h(z, \{b\} : \mathbf{v})))$$

clause form transformation:

$$\{p(b)\}, \{m(h)\}, \{\neg m(x), w(f)\}, \{\neg m(x), h(f, \{b, x\} : \mathbf{u})\}, \{\neg w(z), \neg h(z, \{b\} : \mathbf{v})\}$$

where the additional clause $\{m(h)\}$ stems from the assumption that the domain of men is nonempty.

resolution:

$$\begin{tikzcd}
\{p(b)\} & \{m(h)\} \arrow[dr, dash] \arrow[dd, dash] & \{\neg m(x), w(f)\} \arrow[d, dash] & \{\neg m(x), h(f, \{b, x\} : \mathbf{u})\} \arrow[d, dash] & \{\neg w(z), \neg h(z, \{b\} : \mathbf{v})\} \arrow[dl, dash] \\
 & & \{w(f)\} \arrow[dd, dash] & \{\neg m(x), \neg w(f)\} \arrow[dll, dash] & \\
 & \{\neg w(f)\} \arrow[dr, dash] & & & \\
 & & \square & &
\end{tikzcd}$$

Actually, the only remarkable step in the derivation is resolving

$$\{\neg m(x), h(f, \{b, x\} : \mathbf{u})\} \text{ and } \{\neg w(z), \neg h(z, \{b\} : \mathbf{v})\}$$

with $\{\neg m(x), \neg w(f)\}$ as the resolvent. Here, the two labeled pronoun variables can be identified, because the intersection of their accessible antecedents is nonempty. The corresponding mgu* of

$$\{\neg m(x), h(f, \{b, x\} : \mathbf{u}), \neg w(z), \neg h(z, \{b\} : \mathbf{v})\}$$

is $\sigma := [x/z, z/f, \{b, x\} : \mathbf{u}/\{b\} : \mathbf{v}]$.

Note also, that although $p(b)$ introduced the antecedent b, it is not used in the derivation because all information that is necessary to derive the contradiction is captured by the labels. This is the advantage of using labels; it allows us to express non-local dependency relations in our framework, which is essential for dealing with pronoun binding in dynamic semantics where a pronoun and its antecedent can occur in different formulas.

3.2.4 Evaluation from a Linguistic Point of View

In general, it is of course not enough if one gives just the information that there is a binding that allows to derive a conclusion, but one also wants to know *which* binding. It is straightforward to augment our method in a way such that it accomplishes this simply by memorizing the substitutions of pronoun variables that occur during a derivation.

From a linguistic point of view, one is also interested in comparing different bindings. If we force the proof procedure to backtrack every time it has found a binding which allows to derive a contradiction, we can generate all possible bindings. Probably some of the bindings are preferable to others by taking linguistic heuristics for pronoun resolution into account, see for instance [GJW95, JK96], but this is beyond the scope of the present paper.

3.3 Results

Before we prove completeness and soundness of our method, we have to explain what these notions mean in our setting.

First, we need to explain the notion of *dynamic consequence* $\models_D$. We cannot possibly give full details here, and refer the reader to [Dek93, GS91]. Basically, the dynamic meaning of a formula φ is a function from states to states, where a state is a collection of assignment function. Then $\psi_1, \ldots, \psi_n \models_D \varphi$ means that for every state $s[\![\psi_1]\!] \circ \cdots \circ [\![\psi_n]\!] \subseteq s[\![\varphi]\!]$, where $[\![\chi]\!]$ is the dynamic meaning of χ and $\circ$ denotes functional composition.

We say that $\Gamma \models \varphi$ iff there is a total disambiguation or binding function $\delta : \mathit{LPVAR} \to \mathit{VAR}$ such that $\Gamma\delta \models_D \varphi\delta$, where $\models_D$ is the dynamic entailment relation. Our resolution method returns partial pronoun bindings, when applied to $\bigwedge \Gamma \wedge \neg\varphi$.

Theorem 9 (Soundness and Completeness) *For every set of first-order formulas the following hold:*

1. *If* prf *produces the empty clause on input* Γ*, then there is a disambiguation* δ *of* Γ *such that* $\Gamma\delta$ *is unsatisfiable.*

2. *If there is a disambiguation* δ *of* Γ *such that* $\Gamma\delta$ *is unsatisfiable, then* prf *produces the empty clause on input* Γ*.*

The proof of the soundness half of Theorem 9 can be reduced to the soundness of standard resolution. As to completeness, this may be proved as follows. Assume that for some disambiguation δ, the (disambiguated) set of formulas $\Gamma\delta$ is unsatisfiable. Then, by completeness of the standard resolution method, there is a resolution proof of $\square$ from $\Gamma\delta$. The idea, then, is to turn this proof into a labeled resolution proof of $\square$ from the original set Γ by repeating the resolution steps and inserting the required substitutions (i.e., partial disambiguations) just before any steps where they used in the original proof. Although the idea of this proof is simple, the details are too cumbersome to be included here.

4 Related Work

Most work in the area of ambiguity and discourse semantics focuses on representational issues. Approaches that deal with pronoun binding are mostly trying to bind pronouns by applying some heuristics. The work that is closest to ours is the approach of Kohlhase and Konrad [KK98] who deal with pronoun binding in the setting of natural language corrections by using higher-order unification, and a higher-order tableaux method [Koh95] to reason about possible bindings.

5 Conclusion

In this paper we have presented a resolution calculus for reasoning with ambiguities triggered by pronouns and the different ways to bind them. Deduction steps and pronoun bindings are interleaved with the effect that only pronouns that are used during a derivation are bound to a possible antecedent. Labels allow us to capture relevant structural information of the original formula on a very local level, namely by annotating variables. Therefore structural manipulation, a prerequisite of any efficient proof method, does no harm.

Our ongoing work focuses on two aspects. First, we have to see how our resolution method behaves when additional strategies to restrict the search space are added; these include set-of-support strategy, ordered unification, or subsumption checking. Second, we are in the process of implementing the annotation and unification* algorithms and are trying to integrate them into a resolution theorem prover.

References

[Dek93] P. Dekker. *Transsentential Meditations.* PhD thesis, University of Amsterdam, 1993.

[GJW95] B. Grosz, A. Joshi, and S. Weinstein. Centering: A framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2), 1995.

[GS91] J. Groenendijk and M. Stokhof. Dynamic Predicate Logic. *Linguistics and Philosophy*, 14:39–100, 1991.

[Hei82] I. Heim. *The Semantics of Definite and Indefinite Noun Phrases.* PhD thesis, University of Massachusetts, Amherst, 1982.

[Jas97] J. Jaspars. Minimal logics for reasoning with ambiguous expressions. CLAUS-Report 94, University of Saarbrücken, 1997.

[JK96] J. Jaspars and M. Kameyama. Preferences in dynamic semantics. In M. Stokhof and P. Dekker, editors, *Proceedings of the 10th Amsterdam Colloquium*, pages 445–464. ILLC, Amsterdam, 1996.

[Kam81] H. Kamp. A theory of truth and semantic representation. In J. Groenendijk et al., editor, *Formal Methods in the Study of Language.* Mathematical Centre, Amsterdam, 1981.

[KK98] M. Kohlhase and K. Konrad. Higher-order automated theorem proving for natural language semantics. SEKI-Report SR-98-04, University of Saarbrücken, 1998.

[Koh95] Michael Kohlhase. Higher-order tableaux. In P. Baumgartner et al., editor, *Proceedings TABLEAUX'95*, pages 294–309. Springer-Verlag, New York, 1995.

[KR93] H. Kamp and U. Reyle. *From Discourse to Logic.* Kluwer Academic Publishers, 1993.

[Lei97] A. Leitsch. *The Resolution Calculus.* Texts in Theoretical Computer Science. Springer-Verlag, 1997.

[Lov78] D. W. Loveland. *Automated Theorem Proving: A Logical Bases.* North-Holland, Amsterdam, 1978.

[MdR98] C. Monz and M. de Rijke. A tableaux calculus for ambiguous quantification. In H.C.M. de Swart, editor, *Proceedings TABLEAUX'98*, LNAI. Springer-Verlag, Berlin, 1998.

[MM82] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Trans. on Prog. Languages and Systems*, 4:258–282, 1982.

[Rob65] J. A. Robinson. A machine oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[vD96] K. van Deemter. Towards a logic of ambiguous expressions. In S. Peters and K. van Deemter, editors, *Semantic Ambiguity and Underspecification.* CSLI Lecture Notes, Stanford, Ca., 1996.

[vEJ96] J. van Eijck and J. Jaspars. Ambiguity and reasoning. Technical Report CS-R9616, Centrum voor Wiskunde en Informatica, Amsterdam, 1996.