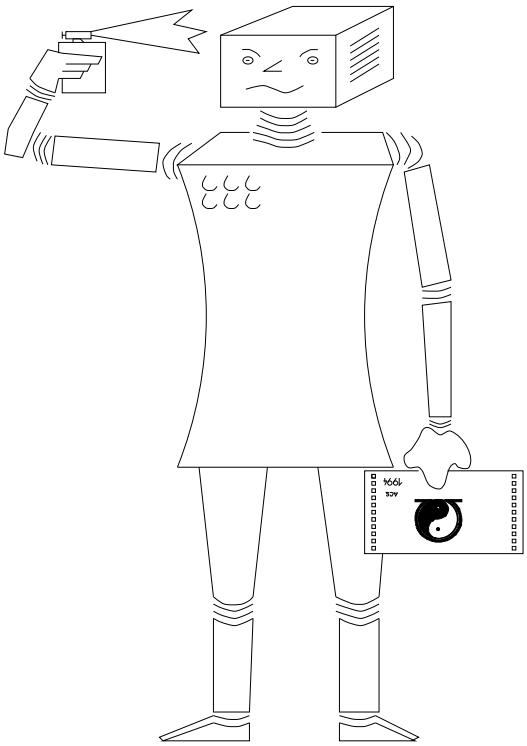

SEKI Report

Higher-Order Automated Theorem Proving for Natural Language Semantics

Michael Kohlhase and Karsten Konrad

SEKI Report SR-98-04

Higher-Order Automated Theorem Proving for Natural Language Semantics

Michael Kohlhase, Karsten Konrad
Dept. of Computer Science,Universität des Saarlandes,Saarbrücken, Germany
{kohlhase,konrad}@cs.uni-sb.de

Abstract

This paper describes a tableau-based higher-order theorem prover HOT and an application to natural language semantics. In this application, HOT is used to prove equivalences using world knowledge during higher-order unification (HOU). This extended form of HOU is used to compute the licensing conditions for corrections.

1 Introduction

Mechanized reasoning systems have many applications in Computational Linguistics. Based on the observation that some phenomena of natural language can be modeled as deductive processes, first-order theorem provers or related inference systems have been used for instance in phonology [2], generation [17] and semantic analysis [22]. [11] describes an abductive framework for natural language understanding that includes world knowledge into the semantics construction process.

Other approaches use higher-order logics and in particular β-reduction and higher-order unification (HOU) as inference procedures. Following Montague [18] who has used the typed λ-calculus as a semantic representation language in order to achieve compositional semantics construction, Dalrymple, Shieber and Pereira [3] model resolution processes by representing semantically underspecified elements (e.g. anaphoric references or ellipses) by free variables whose value is determined by solving higher-order equations (see section 1.1).

Higher-order theorem provers combine logical proof search with HOU. These systems are motivated by the fact that many mathematical problems, such as Cantor's theorem, can be expressed very elegantly in higher-order logic, but lead to an exhaustive and un-intuitive formulation when coded in first-order logic. Recent experiments with set theoretical problems have shown that automated higher-order theorem provers can outperform prominent high-speed first-order theorem provers in such cases.

In this paper, we present the tableau-based higher-order theorem prover HOT and a linguistic application that combines the strengths of the first-order approaches (inclusion of world knowledge) with those of the higher-order ones. Before we present the theorem prover let us get an intuition for our linguistic application.

1.1 HOU and Corrections

Dalrymple, Shieber and Pereira have introduced HOU as an inference procedure for the reconstruction of verb-phrase ellipses in [3]. For the discourse *Dan loves his wife. Peter does too.* they assume the semantic representation (1a) where the value of the predicate variable R is determined by equation (1b)

(1) a. $like(dan, wife\_of(dan)) \wedge R(peter)$
b. $like(dan, wife\_of(dan)) = R(dan)$
c. $R = \lambda X.like(X, wife\_of(X))$
d. $R = \lambda X.like(dan, wife\_of(dan))$

HOU computes the solutions (1c,d), which correspond to the intended **sloppy** reading *peter loves peter's wife* (1c) and the **strict** reading *peter loves dan's wife*. The fact that HO equations can have more than one most general solution is used to account for ambiguity, in this case the so-called sloppy/strict ambiguity.

This approach has been successfully been extended to other parallelism-based phenomena [7], focus [8, 21] and finally corrections [10], which combine focus and parallelism. Since we will use corrections as the prime example for this paper, let us discuss the approach of [10] in more detail.

Corrections are utterances such as (2b–d) where a discourse participant corrects the utterance of some other discourse participant.

(2) a. *A: Jon likes Mary.*
b. *B: No, PETER likes Mary.*
c. $\star$ *B: No, PETER likes Sarah.*
d. *B: No, PETER likes [the woman with the red hat]*

Although there is much AI literature on corrections, a thorough investigation of their linguistic properties is still outstanding. But it is clear that the utterance in (2c) does not form a well–formed correction for (2a), since the source utterance and the target are not parallel. In [10] we model the requirements corrections place on context, or in other words, the relationship between correction and correctum with a variant of the equations given above.

(3) a. $X(a_1, \ldots, a_n) = S$ *and* $X(b_1, \ldots, b_n) = T$

The pair of higher-order equations in (3) states that the semantics of the source utterance (Speaker A) and that of the must be contrastively parallel to that of the target utterance (Speaker B), i.e both have a common part (modeled by the predicate variable X in the equations) and can differ only in the contrastive elements a_i and b_i (The target contrastive elements b_i are given by prosodic markings that we represent by capitalization). For (2) we have the equations

(4) a. $X(jon) = likes(jon, mary)$
b. $X(peter) = likes(peter, mary)$;
solution: $X = \lambda X.likes(X, mary)$
c. $X(peter) = likes(peter, sarah)$;
no solution
d. $X(peter) = \exists^1 x(w(x) \wedge wrh(x) \wedge like(j, x))$;
solution ???

(4a) is the source equation and (4b-d) are the target equations corresponding to the corrections (2b-d). The HOU analysis predicts that (2b) is a correct correction to (2a) – the topic (computed as the value for X) is *liking Mary* – while (2c) is not. An interpretation with topic *liking* with additional contrastive element *Sarah* is impossible, since it is not prosodically marked. Finally, the analysis predicts that (2d) is incorrect, since HOU fails – even in a situation, where *Mary* is the unique woman with a red hat – as it is too weak to identify the semantic equivalence of the two descriptions of *Mary*.

Even though HOU considers $\beta\eta$-equality of formulae, it does not take into account the semantics of the logical connectives and quantifiers contained in the logical representation of natural language utterances. This is exactly the problem that prevents HOU from finding a solution in (4d). Therefore [10] contends that the correct notion of identity is given by **Higher–Order Unification with Equivalence** (HOUE), a form of unification which takes into account not only syntactic identity, but also denotational equivalence.

Obviously, HOUE has to generalize theorem proving methods for higher-order logic, since the task of unifying an equation modulo equivalence $(A \vee \neg A) = T$, where T is a sentence, is equivalent to proving the validity of the theorem T[1].

[1] The formula $(A \vee \neg A)$ must be true in all models, thus T can only be equivalent to it, if it is a theorem.

2 The Theorem Prover HOT

The HOT system (see [16] for details) is based on a free variable tableau calculus $\mathcal{HTE}$ for classical higher-order logic [15]. It is a generalization of the first-order tableau method [6] for automated theorem proving, which refutes a negated theorem by analyzing the connectives in an and/or tree and finding instantiations that close each branch of the tree by finding elementary contradictions on it.

2.1 The calculus $\mathcal{HTE}$

Instead of a formal recapitulation of the tableau method, we discuss the example of the logical theorem $(p(a) \vee p(b) \Rightarrow \exists x.p(x))$. The negation of this is equivalent (in addition to the de Morgan laws we use the identity $\exists x.A = \neg\forall x.\neg A$) to the formula at the root of the following tableau.

$$
\begin{array}{c}
p(a) \vee p(b) \wedge \forall x.\neg p(x) \\
p(a) \vee p(b) \\
\forall x.\neg p(x) \\
\begin{array}{c|c}
p(a) & p(b) \\
\neg p(y) & \neg p(z) \\
*\,[y = a] & *\,[z = b]
\end{array}
\end{array}
$$

Here we see that conjuncts are simply added to the branch, whereas disjunctions are analyzed in separate branches of the tree. The scopes of universal quantifications (with new variables) can be inserted at the end of branches, the same is possible with the scopes of existential quantifications (with the bound variables replaced by Skolem[2] terms). Finally, both branches of the tableau are closed, i.e. the last formula can be instantiated (by the substitution in brackets) so that it contradicts a formula in the branch above.

These instantiations are computed by unification, and in the case of higher-order logic by HOU, which solves the problem of finding substitutions σ that for a given equation $A = B$ make both sides equal in the theory of $\beta\eta$-equality ($\sigma(A) =_{\beta\eta} \sigma(B)$). Huet's well-known algorithm [13] solves the problem by recursively decomposing formulae and binding function variables to most general formulae of a given type and given head for details we refer the reader to [24].

Instead of HOU, the $\mathcal{HTE}$ calculus uses HOUE for computing instantiations, enabling it to prove, e.g., tautologies with embedded equivalent formulae like $\mathbf{A} = c(a) \vee \neg c(\neg\neg a)$. With this measure $\mathcal{HTE}$ removes a source of incompleteness that all earlier higher-order machine-oriented calculi [12, 1] exhibited.

The distinguishing feature of the HOUE algorithm is that intermediate equations $(A = B)$ of type t (generated either by unifying two formulae on the branch to make them contradictory or by processing other unification problems) can be transformed into negated equivalences (which can then be treated by the theorem proving component). Actually, tableau development for the negated equivalence $\neg(A \Leftrightarrow B)$ contains trivial branches, so we use the following (optimized) rule, which splits an equation of type t into two tableau branches

$$
\begin{array}{c}
A = B \\
\begin{array}{c|c}
A & B \\
\neg B & \neg A
\end{array}
\end{array}
\qquad (5)
$$

This way, HOU and tableau theorem proving recursively call each other in HOUE, until a refutation is found (all branches of the tableau are closed). In particular, we can also consider the $\mathcal{HTE}$ calculus as a HOU algorithm with a built-in theorem prover for equivalences. Since the theorem $\mathbf{A}$ consists only of two unit literals, it is equivalent to unifying $c(a)\vee$ and $c(\neg\neg a)$ modulo the embedded equivalence $a \Leftrightarrow \neg\neg a$. We have the following closed tableau for $\mathbf{A}$:

[2] Skolem terms serve as witnesses for the objects whose existence is claimed by the existential formula A. Since this object may depend on the values of free variables $x_1, \ldots, x_n$ occurring in A, they have the form $f(x_1, \ldots, x_n)$ where f is a new function.

$$
\begin{array}{c}
(\neg(cb) \vee c(\neg\neg b))^{\mathtt{F}} \\
\neg(cb)^{\mathtt{F}} \\
c(\neg\neg b)^{\mathtt{F}} \\
cb^{\mathtt{T}} \\
cb \neq^{?} c(\neg\neg b) \\
b \neq^{?} \neg\neg b \\
\hline
\begin{array}{c|c}
\neg\neg b^{\mathtt{F}} & \neg\neg b^{\mathtt{T}} \\
b^{\mathtt{T}} & b^{\mathtt{F}} \\
b^{\mathtt{F}} & b^{\mathtt{T}} \\
b \neq^{?} b & b \neq^{?} b
\end{array}
\end{array}
$$

2.2 Implementation: Concurrent Higher-Order Tableaux

We can conceptualize tableaux implementations as Blackboard architectures [4] where tableau agents, equipped with abilities from their underlying calculus, manipulate a Blackboard-like data structure, the tableau. The proof search space is defined by each agent's possible nondeterministic decisions. An agent may implement search strategies by choosing an order of rule applications or restricting the location of rule applications.

Tableau agents are equipped with a set of structural rules which recursively build up the tableau tree by decomposing the logical structure of formulae and adding new nodes and branches. In order to close a tableau, an agent will try to find for each branch a pair of complementary formulae A and B and a global variable substitution such that $\sigma(A) = \sigma(\neg B)$ holds. The substitution σ is computed using unification, which can be efficiently decided for first-order terms. For higher-order terms, unification is undecidable in general and we can not use it as a decision procedure. The $\mathcal{HTE}$ calculus instead embeds the unification within the structural rules and unification becomes a part of the tableau construction. By choosing a complete search strategy, e.g., iterative deepening on the depth of the tableau, we can make sure that proof search is complete.

The HOT system is implemented in OZ[3], a constraint programming language based on a new computation model providing a uniform foundation for higher-order functional programming, constraint logic programming, and concurrent objects with multiple inheritance [23]. OZ is a concurrent programming language, i.e., a procedure may start sub-processes, called **threads**, which are executed concurrently in a fair way. HOT uses this feature to implement a Blackboard architecture for higher-order tableaux where multiple tableau agents work together in order to construct a proof.

Tableau agents analyze disjunctions in separate branches of the tree. Because tree expansion is potentially infinite, each agent has to respect fairness conditions for branch expansion. The tableaux theorem prover presented in [6] uses an approach where branches are stored in a queue, and tree expansion follows a breadth-first strategy. The LEANTAP theorem prover [20] traverses branches in a left-right, depth-first order and uses an increasing depth bound in order to be complete.

LEANTAP performs a very useful pre-processing step that orders formulae parts, moving subformulae in front which have a smaller, less branching structure. This optimization can not be fully performed for higher-order tableaux because higher-order terms may change their propositional structure by instantiation. We therefore have implemented a **concurrent tableau expansion**. Instead of a single tableau agent that has to decide on the order of branches to visit, we analyze disjunctive branches by multiple agents, each one working autonomously on its own part of the tableau. The first agent which is able close a tableau branch decides on the important choice of the next global variable substitution to explore, hopefully inhibiting unnecessary unification attempts in other parts of the tableau. In this way, the agents communicate with each other by manipulating the variable substitutions that are part of the Blackboard. As long as the concurrent execution of the agents is fair, we emulate a weak form of breadth-first expansion by concurrency.

The concurrent architecture gives us an efficient method to cope with a certain class of HOU constraints. So-called **flex/flex** pairs are unification problems with variable functional heads,

[3] see `http://www.ps.uni-sb.de/ns3/oz/`

e.g., $X(a) = Y(b)$. A flex/flex pair is always solvable, but can have infinitely many incomparable solutions, e.g., $X = Y = \lambda x.a$ for an arbitrary constant a. It is useful to treat flex/flex problems as solved until one of the heads becomes determined by a variable substitution. Each branch of a unification problem is part of the tableau, and therefore a unique agent deals with it. In the case of a branch ending in a flex/flex pair, the agent related to the branch simply suspends and waits for one of the flexible heads to become determined. An instantiation of one of the flexible heads will reactivate the agent, and the extension/closing cycle of the branch continues.

2.3 HOT and the CHOLI System

While HOT can also be used as a stand-alone theorem prover (see [16] for details and examples), its main purpose is the use as an inference module within a natural language system. The CHOLI system [5] includes a graphical user interface for textual input and control, a natural-language parser, an HPSG-like [19] grammar and a semantic construction module that uses higher-order theorem proving and several forms of HOU. All modules have been programmed in OZ.

CHOLI uses HOT as a submodule to construct semantics for natural language utterances that can be processed by HOUE. CHOLI's parser and grammar create a Montague style representation in the simply typed λ-calculus for each syntactic reading. For these representations, the system offers different inference modules, e.g., plain HOU for the treatment of VP-ellipsis (following [3]) or HOT for corrections (following [10]). CHOLI translates correction phenomena into proof problems where necessary additional information (world knowledge) can be added by the user. HOT signals a successful computation back to the main system which then extracts the linguistic information from the proof data structure.

3 The woman with a red hat

As illustrated, we now sketch the main steps of the unification process for example (2d) with equations (which would have been presented to HOT by the CHOLI system):

$$\begin{array}{ll} X(p) & = like(p,s) \\ X(j) & = \exists x(w(x) \wedge wrh(x) \wedge unique(x) \wedge like(j,x)) \end{array}$$

These are solved in a context, where Sarah is the only woman with a red hat. The HOUE method is given access to the hypotheses $unique(s)$, $w(s)$ and $wrh(s)$ by adding them to the initial tableau. In a first step, we solve the second equation to $X = \lambda z.like(x,s)$ and obtain the following tableau:

$$\begin{array}{c} unique(s) \\ w(s) \\ wrh(s) \\ An(p) = like(p,s) \\ \vdots \\ like(j,s) = \exists x(w(x) \wedge wrh(x) \wedge unique(x) \wedge like(j,x)) \end{array}$$

The HOUE rule (2.1) splits the initial equation into two branches. The first one has the form

$$\begin{array}{c} like(j,s) \\ \neg\exists x(w(x) \wedge \ldots \wedge like(j,x)) \\ \begin{array}{c|c|c|c} \neg w(z) & \neg wrh(z) & \neg unique(z) & \neg like(j,z) \\ *[z=s] & *[z=s] & *[z=s] & *[z=s] \end{array} \end{array}$$

and contains the formulae $like(j,s)$ and $(\neg\exists x(w(x) \wedge wrh(x) \wedge unique(x) \wedge like(j,x)))$. The latter is universally quantified[4] and can therefore be developed into four branches $\neg w(z)$, $\neg wrh(z)$, $\neg unique(z)$, and $\neg like(j,z)$. The first branches can be closed using the hypotheses on Sarah and the last with the first formula, all by binding the new variable z to j.

[4] We use that $\neg\exists x.A$ is equivalent to $\forall x.\neg A$ here

The second branch consists of the formulae $\neg like(j,s)$ and $\exists x(w(x) \wedge wrh(x) \wedge unique(x) \wedge like(j,x))$, which is developed into the single branch containing the conjuncts $w(c)$, $wrh(c)$, $unique(c)$, and $like(j,c))$, where c is a Skolem constant for x:

$$\begin{array}{c} \neg like(j,s) \\ \exists x(\ldots \wedge unique(x) \wedge like(j,x)) \\ unique(c) \\ like(j,c)) \\ \vdots \\ c = s \\ like(j,s) \\ * \, [] \end{array}$$

Here an expansion of the definition of uniqueness

$$unique(x) \Leftrightarrow \forall z_{\blacksquare}\, w(z) \wedge wrh(z) \Rightarrow x = z$$

closes the branch (if Sarah and c are unique, then $s = c$).

By now, it should be clear that the HOT/ChoLi-system will also encounter no particular problem in dealing with examples such as (7) and (6) below. The first example relies on the world-knowledge, that marrying is a symmetric relation (both partners have to say "yes I do"), whereas the second relies on the fact that getting wounded is synonymous to being hurt by someone/thing. Once these equivalences are taken into account, the HOUE analysis of corrections will correctly predict that these examples are well–formed.

(6) a. *A: Jon married Sarah.*
b. *B: No, Sarah married PETER.*

(7) a. *A: Sarah hurt Paul.*
b. *B: No, PETER was wounded by Sarah.*

We have seen that a deaccented anaphor must either have a semantic representation which syntactically unifies with that of its antecedent, or be semantically equivalent to this antecedent. To show that this is a necessary condition, we need to provide some ill–formed examples in which neither condition holds. Such examples are given when the correction contains a destressed pronoun whose source parallel element is an indefinite (8).

(8) a. *Jon eats an*$_1$ *apple.*
b. $*$ *No, PETER eats it*$_1$.

In this example the semantic representation of the pronoun in the correction fails to syntactically unify with the semantic representation of its antecedent. Therefore it cannot be proved that *it* is semantically equivalent to *an apple* and thus unification fails correctly ruling out (8). The logical reason for this is that while the second equation $An(p) = eat(p,y)$ in 8 can be solved to $An = \lambda x.eat(x,y)$ yielding the propositional unification problem $eat(j,y) = \exists x.ap(x) \wedge eat(j,x)$, which is equivalent to refuting $\neg(eat(j,y) \Leftrightarrow \exists x.ap(x) \wedge eat(j,x))$ by rule (2.1). This is impossible, since this is satisfiable.

As we have seen in section 3, in the case where there is a unique apple, and this argument breaks down, HOUE finds the correct solution.

4 Experimental Results

We have encoded the examples from this papers as proof problems for HOT[5] and measured HOT's performance on a Pentium Pro 200-based LINUX workstation. Figure 1 shows the plain runtime (Run), the time spend for copying data structures (Copy) and the total time for computing a proof, including garbage collection (Total). Each value is given in msec.

[5] see `http://www.ags.uni-sb.de/~konrad/software/corr.oz`

Problem	Run	Copy	Total
(2b)	20	0	20
(2d)	1850	12900	19610
(5b)	660	2640	4260
(6b)	60	50	110

Figure 1: Corrections as proof problems.

In order to suppress an excessive use of the extensionality rule, we have restricted the number of applications of this rule within each branch of the tableau. The examples were solved with an extensionality depth limit of 4, i.e., a very moderate restriction.

All proofs except "*the woman with the red hat*" (2d) can be found automatically by using the standard parameter settings. Problem (2d) requires an additional restriction on the minimum size of the proof tree. Basically, we have to force HOT to skip over all proof trees which are too shallow to contain a solution. (2d) can be solved with an extensionality depth of 1 in 12.5 seconds. For problem (2c), HOT correctly does not find a proof within the given boundaries of the proof length.

The HOT system uses an object-oriented programming paradigm in order to realize special-purpose and experimental proof procedures which inherit most of their properties from the standard algorithm. The so-called **proof engines** put different heuristics to work and permit us to evaluate different proof strategies. One of these engines implements an incomplete but decidable proof procedure that does not use instances of universally quantified formulae more than once within each branch of the tableau. While one can not prove many interesting mathematical problems using this restricted engine, all examples from above fall under this category of "obvious" problems. Using the Obvious proof engine, we can solve for instance (2d) in less than 8 seconds with an extensionality depth of 4. We expect that further experimentation will reveal more adequate restrictions that eventually lead to an inference engine that is especially useful for semantics construction.

5 Conclusion

We have presented a tableau-based higher-order theorem prover HOT, which allows to take into account world knowledge during the semantic analysis for parallelism constructs like corrections. Even though we have not used any linguistic information for guiding the search for proofs, the problems studied so far remain tractable for the HOT theorem prover if we only add the knowledge needed for a successful analysis.

So far, there exists no theorem prover that is optimized for inferences in natural language processing. We are interested in proof techniques and heuristics which use linguistic knowledge to guide the proof search. One of these techniques is higher-order **colored** unification (HOCU) [14] which can be used to guide the search for unifiers in natural language semantics [9] based on knowledge coming from natural language syntax. HOT's unification algorithm developed by Martin Müller and the second author already includes the constraint propagation needed for computing colored unifiers.

Analytical proof procedures like tableaux have the advantage that they can enumerate proofs, giving different answers for the same problem. This feature is far more important for natural language processing than for mathematical theorem proving where we want to map different readings for a natural language utterance to different proofs. While we can create a proof for each reading with the current implementation, we will usually be confronted with several proofs which differ only in their structure, but contain identical linguistic knowledge. The proof procedure over-generates solutions, and we need a closer cooperation between CHOLI and HOT in order to filter out interesting proofs.

Finally, we want to use Abduction to synthesize new knowledge from utterances. Consider the correction[6]

[6]The example has been proposed by Noor van Leusen[25].

John called Gerald a Fascist.
No, PAUL insulted him.

Given the correction, we can infer that John calling Gerald a Fascist is an insult to Gerald and that not John but Paul insulted Gerald. The information that being called a Fascist is an insult need not to be stated explicitly in an abductive framework where we can assume this information if it leads us to a proof. In this sense, the linguistic knowledge about corrections will provide us with additional information about the utterance's context.

Concretely, we intend to integrate the HOUE framework with Hobbs' et al. abductive approach to natural language understanding in a higher-order abductive process which uses weighted abduction [11] to infer the new information in an utterance (that, which has to be assumed to explain the semantics of a the new utterance with respect to the background knowledge). Such a higher-order framework would alleviate some of the problems caused by the first-order framework (dealt with by "Ontological Promiscuity" in [11]) and include HOU as a special case (abduction of the unifying instantiations).

References

[1] Peter B. Andrews, 'On Connections and Higher Order Logic', *J. of Automated Reasoning*, **5**, 257–291, (1989).

[2] Denis Bechet and Philippe de Groote, 'Constructing different phonological brackettings from a proof net', in LACL96, (1996).

[3] Mary Dalrymple, Stuart Shieber, and Fernando Pereira, 'Ellipsis and higher-order unification', *Linguistics and Philosophy*, **14**, 399–452, (1991).

[4] *Blackboard Systems*, eds., R. Engelmore and T. Morgan, Addison-Wesley, 1988.

[5] Markus Egg et al., 'ChoLi: A Natural Language System for Semantic Construction and Evaluation', CLAUS Report 103, Universität des Saarlandes, (January 1998). `http://coli.uni-sb.de/claus/claus97.html`.

[6] Melvin Fitting, *First-Order Logic and Automated Theorem Proving*, Springer Verlag, 1990.

[7] Claire Gardent, 'Sloppy identity', in *Logical Aspects of Computational Linguistics*, Springer-Verlag, (1997).

[8] Claire Gardent and Michael Kohlhase, 'Focus and higher–order unification', in *Proc. COLING'96*, (1996).

[9] Claire Gardent, Michael Kohlhase, and Karsten Konrad, 'Higher-order coloured unification: a linguistic application'. Submitted for Publication, 1997.

[10] Claire Gardent, Michael Kohlhase, and Noor van Leusen, 'Corrections and Higher-Order Unification', in *Proceedings of KONVENS96*, 268–279, De Gruyter, Bielefeld, Germany, (1996).

[11] J. Hobbs, M. Stickel, D. Appelt, and P. Martin, 'Interpretation as abduction', *Artificial Intelligence*, **63**, 69–142, (1993).

[12] Gérard P. Huet, 'A mechanization of type theory', in *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, eds., Donald E. Walker and Lewis Norton, pp. 139–146, (1973).

[13] Gérard P. Huet, 'An unification algorithm for typed λ-calculus', *Theoretical Computer Science*, **1**, 27–57, (1975).

[14] Dieter Hutter and Michael Kohlhase, 'A coloured version of the λ-calculus', in *Proceedings of the 14th Conference on Automated Deduction*, ed., William McCune, number 1249 in LNAI, pp. 291–305, (1997). Springer Verlag.

[15] Michael Kohlhase, 'Higher-Order Tableaux', in *Theorem Proving with Analytic Tableaux and Related Methods*, eds., P. Baumgartner, R. Hähnle, and J. Posegga, volume 918 of *LNAI*, pp. 294–309, (1995).

[16] Karsten Konrad, 'HOT: Implementing Higher-Order Tableaux', Seki Report SR-98-03, Fachbereich Informatik, Universität Saarbrücken, (1998).

[17] Josep Maria Merenciano and Glyn Morrill, 'Generation as deduction', in Proc. LACL96, (1996).

[18] R. Montague, 'The proper treatment of quantification in ordinary english', in *Formal Philosophy. Selected Papers*, ed., R. Montague, Yale University Press, New Haven, (1974).

[19] Carl Pollard and Ivan Sag, *Head-Driven Phrase Structure Grammar*, Lecture Notes 13, Stanford, 1987.

[20] Joachim Posegga and Peter H. Schmitt, 'Implementing Semantic Tableaux', Interner bericht, Fakultät für Informatik, Unversität Karlsruhe, (1996).

[21] Steve G. Pulman, 'Higher-order unification and the semantics of focus', in *Third Nordic Conference on Text Comprehension in Man and Machine*, pp. 113–127, Linköping University, (1993).

[22] Uwe Reyle and Dov M. Gabbay, 'Direct deductive computation on discourse representation structures', *Linguistics and Philosophy*, **17**, 343–390, (1994).

[23] Gert Smolka, 'The Oz Programming Model', in *Computer Science Today*, ed., Jan van Leeuwen, volume 1000 of *LNCS*, pp. 324–343. Springer Verlag, (1995).

[24] Wayne Snyder, *A Proof Theory for General Unification*, Progress in Computer Science and Applied Logic, Birkhäuser, 1991.

[25] Noor van Leusen, 'The Role of Inference in the Resolution of Corrections', CLAUS Report 93, Universität des Saarlandes, (December 1997). `http://coli.uni-sb.de/claus/claus97.html`.