Completeness of Hyper-Resolution via the Semantics of Disjunctive Logic Programs*

Linh Anh Nguyen†
Institute of Informatics
University of Warsaw
ul. Banacha 2, 02-097 Warsaw
Poland
nguyen@mimuw.edu.pl

Rajeev Goré
RSISE and NICTA
The Australian National University
Canberra ACT 0200
Australia
Rajeev.Gore@anu.edu.au

Abstract

We present a proof of completeness of hyper-resolution based on the fixpoint semantics of disjunctive logic programs. This shows that hyper-resolution can be studied from the point of view of logic programming.

Keywords: fixpoint semantics, automated theorem proving

1 Introduction

Resolution was introduced by Robinson in his landmark paper [24] in 1965 as a mechanisable method for detecting the unsatisfiability of a given set of formulae of classical first-order logic. It revolutionised the field of automated reasoning, and since then, many refinements of resolution have been proposed by researchers in the field in order to cut down the search space and increase efficiency. One of the most important refinements of resolution is hyper-resolution, which was also introduced by Robinson [23] in the same year 1965. Hyper-resolution constructs a resolvent of a number of clauses at each step. Thus it contracts a sequence of bare resolution steps into a single inference step and eliminates interactions among intermediary resolvents, and interactions between them and other clauses.

Resolution and hyper-resolution have been well studied. There are a number of well-known proofs of completeness of resolution and hyper-resolution. The classical proofs of completeness of hyper-resolution are often based on proofs

*Information Processing Letters, 95(2), 2005, 363-369

†Partially supported by a visiting fellowship from NICTA. National ICT Australia (NICTA) is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

of completeness of resolution. In Leitsch's book on resolution [12], completeness results are proved using the semantic tree technique [11]. Some other methods for proving completeness of resolution and hyper-resolution are Bachmair and Ganzinger's forcing technique [3], Nivelle's resolution game technique [7], Boyer's excess literal technique (see [6]), and a proof-theoretic method by Goubault-Larrecq [9].

There is a close relationship between the theory of logic programming and resolution. A refinement of resolution for the Horn fragment, called SLD-resolution in [1], was first described by Kowalski [10] for logic programming. It is a top-down procedure for answering queries in definite logic programs. On the other hand, a bottom-up method for answering queries is based on fixpoint semantics of logic programs and was first introduced by van Emden and Kowalski [25] using the direct consequence operator T_P. This operator is monotonic, continuous, and has the least fixpoint $T_P \uparrow \omega = \bigcup_{n=0}^{\omega} T_P \uparrow n$, which forms the least Herbrand model of the given logic program P.

In [16], Minker and Rajasekar extended the fixpoint semantics to disjunctive logic programs. Their direct consequence operator, denoted by T_P^I, iterates over model-states, which are sets of disjunctions of ground atoms. This operator is also monotonic, continuous, and has a least fixpoint which is a least model-state characterizing the given program P.

Fixpoint semantics of logic programs are closely related to hyper-resolution. More precisely, the intuition behind hyper-resolution is the same as that behind the fixpoint semantics of disjunctive logic programs, as has been observed by researchers in logic programming. There are, however, the following differences: a disjunctive logic program is a set of *non-negative* clauses, and the direct consequence operator (as defined in [16, 14]) *simultaneously* applies inference steps for all *ground* instances of clauses.

Despite the fact that a number of resolution systems have been implemented in Prolog, the relationship between the theory of logic programming and the theory of resolution has been, in our opinion, mostly one-directional: logic programming has been studied from the point of view of resolution. Our thesis is that hyper-resolution can be studied from the point of view of logic programming. In this paper, we give a proof of completeness of hyper-resolution based on the fixpoint semantics of disjunctive logic programs.

2 Preliminaries

First-order logic is considered in this work and we assume that the reader is familiar with it. We assume we are given a finite set of first-order formulae, which we wish to test for satisfiability. We now give the most important definitions for our work.

2.1 Herbrand Models

Let L be the underlying first order language for the considered set of formulae. Normally, we assume that L is defined by the constants, function symbols and predicate symbols appearing in the considered set of formulae.

The *Herbrand universe* U_L for L is the set of all ground terms, which can be formed out of the constants and function symbols appearing in L. If L has no constants, we add some constant, say a, to form ground terms.

An *Herbrand interpretation* for L is an interpretation for L such that:

- The domain of the interpretation is the Herbrand universe U_L;
- Constants in L are assigned themselves in U_L;
- If f is an n-ary function symbol in L, then the mapping from $(U_L)^n$ to U_L defined by $(t_1, \ldots, t_n) \to f(t_1, \ldots, t_n)$ is assigned to f.

Since the assignment to constants and function symbols is fixed in Herbrand interpretations, we can identify an Herbrand interpretation with the set of all ground atoms which are true with respect to that interpretation.

Let S be a set of closed formulae of L. An *Herbrand model* of S is an Herbrand interpretation for L which is a model for S.

2.2 Unification

A *substitution* is a finite set $\theta = \{x_1 := t_1, \ldots, x_n := t_n\}$, where $x_1, \ldots, x_n$ are different variables, $t_1, \ldots, t_n$ are terms, and $t_i \neq x_i$ for all $1 \leq i \leq n$. By ε we denote the *empty substitution*.

An *expression* is either a term or a formula without quantifiers, and a *simple expression* is either a term or an atom.

Let $\theta = \{x_1 := t_1, \ldots, x_n := t_n\}$ be a substitution and E be an expression. Then $E\theta$, the *instance* of E by θ, is the expression obtained from E by simultaneously replacing all occurrence of the variable x_i in E by the term t_i, for $1 \leq i \leq n$.

Let $\theta = \{x_1 := t_1, \ldots, x_n := t_n\}$ and $\delta = \{y_1 := s_1, \ldots, y_m := s_m\}$ be substitutions. Then the *composition* $\theta\delta$ of θ and δ is the substitution obtained from the set $\{x_1 := (t_1\delta), \ldots, x_n := (t_n\delta), y_1 := s_1, \ldots, y_m := s_m\}$ by deleting any binding $x_i := (t_i\delta)$ for which $x_i = (t_i\delta)$ and deleting any binding $y_j := s_j$ for which $y_j \in \{x_1, \ldots, x_n\}$.

If θ and δ are substitutions such that $\theta\delta = \delta\theta = \varepsilon$, then we call them *renaming substitutions* and use θ^{-1} to denote δ (which is unique w.r.t. θ).

A substitution θ is *more general* than a substitution δ if there exists a substitution γ such that $\delta = \theta\gamma$. Note that according to our definition, θ is more general than itself.

Let S be a set of simple expressions. A substitution θ is called a *unifier* for S if $S\theta$ is a singleton. If $S\theta = \{E\}$ then we say that θ unifies S (into E). A unifier θ for S is called a *most general unifier* (mgu) for S if θ is more general than every unifier of S.

There is an effective algorithm, called the *unification algorithm*, for checking whether a set S of simple expressions is unifiable (i.e. has a unifier) and computing an mgu for S if S is unifiable (see, e.g., [13]).

Lemma 1 *Let $\mathcal{S}$ be a collection of sets $S_1, \ldots, S_n$ of simple expressions for some fixed integer n. Suppose there exists a substitution which unifies each S_i for $1 \leq i \leq n$. Then there exists a substitution θ, called an mgu of $\mathcal{S}$, such that:*

- *θ unifies each S_i for $1 \leq i \leq n$;*
- *θ is more general than any substitution that unifies each S_i for $1 \leq i \leq n$.*

Proof. By induction on the number of sets n.

Base Case: If $n = 1$ and δ unifies S_1, then let θ be the mgu of S_1. By definition, it unifies S_1 and is more general than any other unifier of S_1 including δ.

Induction Hypothesis: Assume the lemma holds for all integers $0 < n \leq k$.

Induction Step: Suppose $n = k + 1$ and suppose that δ unifies each S_i for $1 \leq i \leq k + 1$. The collection of sets $S_1, \ldots, S_k$ satisfies the conditions of the lemma, so by the induction hypothesis we know there is a substitution θ_k which unifies each S_i for $1 \leq i \leq k$ and that θ_k is more general than δ.

Let γ_k be a substitution so that $\delta = \theta_k\gamma_k$, making γ_k a unifier for $S_{k+1}\theta_k$. Let σ_k be an mgu for $S_{k+1}\theta_k$. So γ_k and σ_k are both unifiers for $S_{k+1}\theta_k$, but σ_k is an mgu, hence there is a substitution γ_{k+1} such that $\gamma_k = \sigma_k\gamma_{k+1}$.

Since θ_k unifies each S_i for $1 \leq i \leq k$, and σ_k is an mgu for $S_{k+1}\theta_k$, the substitution $\theta = \theta_k\sigma_k$ must unify each S_i for $1 \leq i \leq k + 1$. Moreover, $\delta = \theta_k\gamma_k = \theta_k\sigma_k\gamma_{k+1} = \theta\gamma_{k+1}$, so θ is more general than δ.

Finally, our choice of δ was arbitrary, and θ does not actually depend upon δ, so the lemma must hold for any such δ.

□

2.3 Hyper-Resolution

A *clause* is a formula of the form

$$\forall x_1 \ldots \forall x_h(A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_m)$$

where $x_1, \ldots, x_h$ are all the variables occurring in the rest of the formula, $n \geq 0$, $m \geq 0$, and each A_i and B_j are *atoms*. We write such a clause in the form

$$A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$$

If $n = 0$ and $m = 0$ then the clause is *empty*. If $n = 0$ and $m > 0$ then the clause is *negative* and called a *goal*. If $m = 0$ and $n > 0$ then the clause is *positive* and treated as the set of its atoms. If $n \geq 1$ then we call the clause a (disjunctive) *program clause* or a *non-negative* clause.

A *disjunctive logic program* is a finite set of disjunctive program clauses.

It is well known that any set of first-order formulae can be transformed into a set of clauses such that the transformation preserves satisfiability. So we can assume that our initial set of first-order formulae are actually clauses.

Let $\perp$ be a special atom not belonging to the primitive language: that is, $\perp$ does not occur in the set of clauses given to us. We do not assign any formal meaning to $\perp$, but just treat it as a positive literal. We use it to transform every negative clause $\leftarrow B_1 \wedge \ldots \wedge B_m$ into the program clause $\perp \leftarrow B_1 \wedge \ldots \wedge B_m$. Thus we translate our given set of clauses to a disjunctive logic program. If $\perp$ is derivable from the obtained program, then the original given set of clauses is unsatisfiable.

Let $\varphi = A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ with $m \geq 0$, and let $\psi_1, \ldots, \psi_m$ be positive clauses. Let $\psi_i = \xi_i \vee \zeta_i$ for $1 \leq i \leq m$, where ξ_i is a non-empty set of so called *selected atoms* of ψ_i. If there exists an mgu θ which unifies each $\xi_i \cup \{B_i\}$ for $1 \leq i \leq m$, then we call the positive clause $(A_1 \vee \ldots \vee A_n \vee \zeta_1 \vee \ldots \vee \zeta_m)\theta$, written in the compact form with no atom duplicates, a *hyper-resolvent* of φ and $\psi_1, \ldots, \psi_m$. Note that "factoring" is hidden in this definition. Also note that if $m = 0$ then θ is empty and φ is a hyper-resolvent of itself.

Let Γ be a set of clauses. A *derivation* from Γ is a sequence $\varphi_1, \ldots, \varphi_n$ of positive clauses such that, for each $1 \leq i \leq n$, φ_i is a hyper-resolvent of a clause of Γ (called the *input* clause) and standardized variants of some clauses from $\varphi_1, \ldots, \varphi_{i-1}$, where a *standardized variant* is a renaming of all the variables in the original clause so that it does not contain variables of the input clause and the other involved variants. A *refutation* of Γ is a derivation from Γ with the empty clause as the last clause of the sequence.

Let Γ be a set of clauses. A *ground derivation* from Γ is a sequence $\varphi_1, \ldots, \varphi_n$ of positive ground clauses such that, for each $1 \leq i \leq n$, φ_i is a hyper-resolvent of a ground instance of a clause of Γ and some clauses from $\varphi_1, \ldots, \varphi_{i-1}$.

3 Fixpoint Semantics of Disjunctive Logic Programs

In this section, we present the fixpoint semantics of disjunctive logic programs invented by Minker and Rajasekar [16] (see also [14]). Our formulation is slightly different than that of [16, 14].

Let P denote a disjunctive logic program. We shall define the direct consequence operator T_P as a function that maps a set I of positive ground clauses to another set of positive ground clauses which can be "directly" derived from P and I. It is formally defined as follows: $T_P(I)$ is the set of all positive ground clauses which are a hyper-resolvent of a ground instance of a clause of P and some clauses from I.

Lemma 2 *The operator T_P is monotonic, compact, and hence also continuous.*

Hence it has the least fixpoint $T_P \uparrow \omega = \bigcup_{n=0}^{\omega} T_P \uparrow n$, where $T_P \uparrow 0 = \emptyset$ and $T_P \uparrow (n+1) = T_P(T_P \uparrow n)$.

The first assertion of the above lemma clearly holds. The second assertion immediately follows from the first one by Kleene's theorem (see, e.g., [13]).

The following theorem is a consequence of the results of [14].

Theorem 3 *Every minimal Herbrand model of $T_P \uparrow \omega$ is a minimal Herbrand model of P.*

Proof. Let M be a minimal Herbrand model of $T_P \uparrow \omega$. Since $T_P \uparrow \omega$ contains only consequences of P, it suffices to prove that M is an Herbrand model of P. Let $\varphi = A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ be a ground instance of some clause of P and suppose that $M \vDash B_1 \wedge \ldots \wedge B_m$. It suffices to show that $M \vDash A_1 \vee \ldots \vee A_n$.

Since each B_i is a ground atom and $M \vDash B_1 \wedge \ldots \wedge B_m$, we must have $B_i \in M$ for all $1 \leq i \leq m$. Since M is a minimal Herbrand model of $T_P \uparrow \omega$, it follows that for every $1 \leq i \leq m$, there exists $\psi_i \in T_P \uparrow \omega$ such that ψ_i can be written as $B_i \vee \zeta_i$ and ζ_i is false in M. We have that $\psi = A_1 \vee \ldots \vee A_n \vee \zeta_1 \vee \ldots \vee \zeta_m$ is a hyper-resolvent of φ and $\psi_1, \ldots, \psi_m$. Hence $\psi \in T_P \uparrow \omega$. Since $M \vDash T_P \uparrow \omega$ and ζ_i is false in M for all $1 \leq i \leq m$, we have $M \vDash A_1 \vee \ldots \vee A_n$. □

Corollary 4 *For every Herbrand model M of $T_P \uparrow \omega$, there exists a minimal Herbrand model M' of P such that $M' \subseteq M$.*

Proof. It is easily seen that if $M_1 \supseteq M_2 \supseteq \ldots$ is a chain of Herbrand models of $T_P \uparrow \omega$ then their intersection is also an Herbrand model of $T_P \uparrow \omega$. By the Zorn-Kuratowski lemma, it follows that for every Herbrand model M of $T_P \uparrow \omega$ there exists a minimal Herbrand model M' of $T_P \uparrow \omega$ such that $M' \subseteq M$. By Theorem 3, such a model M' is also a minimal Herbrand model of P. □

4 Completeness of Hyper-Resolution

As usual, to prove completeness of a first-order resolution system we need a lifting lemma.

Lemma 5 (Simplified Lifting Lemma) *Let φ^g be a ground instance of a clause φ, and for each $1 \leq i \leq m$, let ψ_i^g be a ground instance of a positive clause ψ_i. Suppose that ψ^g is a hyper-resolvent of φ^g and $\psi_1^g, \ldots, \psi_m^g$. Let $\delta_1, \ldots, \delta_m$ be renaming substitutions for standardising apart $\psi_1, \ldots, \psi_m$, with respect to φ as an input clause. Then there exists a hyper-resolvent ψ of φ and $\psi_1\delta_1, \ldots, \psi_m\delta_m$ such that ψ^g is a (ground) instance of ψ.*

Proof. Let $\varphi^g = \varphi\sigma$ and $\varphi = A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$. We have that $\varphi^g = A_1^g \vee \ldots \vee A_n^g \leftarrow B_1^g \wedge \ldots \wedge B_m^g$, where for each $1 \leq j \leq n$, $A_j^g = A_j\sigma$, and for each $1 \leq i \leq m$, $B_i^g = B_i\sigma$.

Let $\psi_i^g = \psi_i\sigma_i$, for $1 \leq i \leq m$. Without loss of generality, for each $1 \leq i \leq m$, we can assume that $\psi_i^g = B_i^g \vee \zeta_i^g$, where B_i^g is a selected atom of ψ_i^g and is not an atom of ζ_i^g. Thus $\psi^g = A_1^g \vee \ldots \vee A_n^g \vee \zeta_1^g \vee \ldots \vee \zeta_m^g$.

For each $1 \leq i \leq m$, let ξ_i be the set of all atoms $B_{i,k}$ of ψ_i such that $B_i^g = B_{i,k}\sigma_i$, and let ζ_i be the set of the remaining atoms of ψ_i. We have that $\psi_i = \xi_i \vee \zeta_i$, $\xi_i\sigma_i = B_i^g$, and $\zeta_i\sigma_i = \zeta_i^g$, for $1 \leq i \leq m$.

Let $\mu = \sigma \cup \delta_1^{-1}\sigma_1 \cup \ldots \cup \delta_m^{-1}\sigma_m$. Thus μ is a well-defined substitution which unifies each $\xi_i\delta_i \cup \{B_i\}$ into B_i^g for $1 \leq i \leq m$: we have that $\xi_i\delta_i\mu = \xi_i\delta_i\delta_i^{-1}\sigma_i = \xi_i\sigma_i = B_i^g$ and $B_i\mu = B_i\sigma = B_i^g$. By Lemma 1, there exists an mgu θ which unifies each $\xi_i\delta_i \cup \{B_i\}$ for $1 \leq i \leq m$ and is more general than μ. Let $\mu = \theta\gamma$.

Let ψ be the hyper-resolvent of $\varphi = A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ and $\psi_1\delta_1 = (\xi_1 \vee \zeta_1)\delta_1, \ldots, \psi_m\delta_m = (\xi_m \vee \zeta_m)\delta_m$ using the mgu θ with each $\xi_i\delta_i$ as the set of selected atoms of $\psi_i\delta_i$. Then $\psi = (A_1 \vee \ldots \vee A_n \vee \zeta_1\delta_1 \vee \ldots \vee \zeta_m\delta_m)\theta$.

For every $1 \leq j \leq n$, we must have $A_j\theta\gamma = A_j\mu = A_j\sigma = A_j^g$, and for every $1 \leq i \leq m$, we must have $\zeta_i\delta_i\theta\gamma = \zeta_i\delta_i\mu = \zeta_i\delta_i\delta_i^{-1}\sigma_i = \zeta_i\sigma_i = \zeta_i^g$. Hence $\psi\gamma = \psi^g$, which completes the proof. □

Theorem 6 (Completeness) *Every finite and unsatisfiable set* Γ *of clauses has a hyper-resolution refutation.*

Proof. Replace every negative clause $\leftarrow B_1 \wedge \ldots \wedge B_m$ in Γ by $\perp \leftarrow B_1 \wedge \ldots \wedge B_m$. Denote the resulting set by P. It is clear that $P \cup \{\leftarrow \perp\}$ is also unsatisfiable. Hence $P \models \perp$: that is, every model for P is a model for $\perp$.

We show that $\perp \in T_P \uparrow \omega$. For a contradiction, suppose that $\perp \notin T_P \uparrow \omega$. By definition, $T_P \uparrow \omega$ contains only positive ground clauses, each of which we can treat as the set of its atoms. Thus no clause contains duplicated atoms. Since $\perp \notin T_P \uparrow \omega$, every such clause of $T_P \uparrow \omega$ contains some atom different from $\perp$. By choosing these atoms, and assigning them "true", we can obtain an Herbrand model M of $T_P \uparrow \omega$ not containing $\perp$. By Corollary 4, there exists an Herbrand model M' of P such that $\perp \notin M'$, which contradicts $P \models \perp$.

Therefore $\perp \in T_P \uparrow n$ for some n and there exists a ground derivation from P with $\perp$ as the last clause. By Lemma 5, it follows that there exists a derivation from P with $\perp$ as the last clause. Simulate that derivation for Γ as follows: whenever $\perp \leftarrow B_1 \wedge \ldots \wedge B_m$ is used in the derivation, replace it by $\leftarrow B_1 \wedge \ldots \wedge B_m$. The resulting derivation is a refutation of Γ. □

5 Discussion

We started to think about this work when developing a hyper-resolution calculus for first-order modal logics. A number of resolution systems had been previously developed for first-order modal theorem proving, but all of them are based on translation to classical logic [22, 2, 21]. We wanted to use a "direct" approach. The idea is that, because there is a close relationship between hyper-resolution and fixpoint semantics of disjunctive logic programs, one can extend the fixpoint semantics of positive modal logic programs given in [17, 18] for disjunctive modal logic programs and modify it to obtain a hyper-resolution calculus for first-order modal logics. When proving completeness of our calculus, we first tried to apply the technique in Leitsch's book [12] but then met a difficulty with the semantic tree technique. So we decided to try a direct proof using the fixpoint semantics of disjunctive modal logic programs. This paper is written with a hope that our method may be useful for others who have a similar problem when developing resolution calculi for non-classical logics.

Our proof itself is just another proof for a well-known fact. However, it may be intuitive for some people. For example, as the simplified lifting Lemma 5 is more or less standard and Theorem 6 has a short and understandable proof, we expect that our proof will be welcomed by researchers of logic programming.

As mentioned earlier, the relationship between hyper-resolution and fixpoint semantics of disjunctive logic programs has been observed by other researchers. However, it was not explicitly studied as done here: hyper-resolution is not mentioned in the semantics of disjunctive logic programs from [16, 14].

On technical matters, our proof uses the saturation technique, which is a common technique for proving completeness of logical calculi. Roughly speaking, saturation methods apply some operator to derive as many consequences of the considered formula set as is possible until the set is "saturated". The important questions then are the choice of operator and how to apply it systematically. In the theory of logic programming, the direct consequence operator T_P is taken and it is well-known that, since T_P is monotonic and continuous, it suffices to apply this operator ω times to reach the least fixpoint. The operator T_P is continuous because it is monotonic and compact, which is clear to see.

In [3, 4], Bachmair and Ganzinger also use the saturation technique to prove completeness of various refinements of resolution, including hyper-resolution. In spite of the similarity in using the saturation technique, our proof is essentially different from the proofs of Bachmair and Ganzinger [3, 4]. Given a *total* and *well-founded* ordering on the set of ground atoms, the completeness proofs given in [3, 4] are based on an induction on the ordering. This means that the systematic way used in [3, 4] for creating a saturation is related to the ordering. Completeness proofs in Leitsch's book [12] also use saturation, but they are based also on the semantic tree technique.

In automated model building, hyper-resolution can be used to extract minimal models (from the ground instances of hyper-resolvents of the considered clause set). This has been often studied for classes of inputs for which resolution decision procedures can be established and finite minimal models can be extracted (see, e.g., [12, 8]). In this paper, extraction of minimal models for disjunctive logic programs is formulated for the general case. (Theorem 3 is a consequence of the results of Lobo et al [14], but not explicitly stated there.)

The problem of generating minimal models has been also studied implicitly by Manthey and Bry [15] and explicitly by Bry and Yahya [5]. In their work, minimal models are generated by positive unit hyper-resolution tableaux, which differ from traditional hyper-resolution in two respects: first they require a splitting rule, and second they allow a hyper-resolvent to be derivable from a clause and positive *unit* clauses only.

Recently, Nguyen has applied our method to prove answer completeness of negative hyper-resolution semantics of disjunctive logic programs [19]. In [20], he successfully adapted our method to extend [19] to ordered hyper-resolution.

Acknowledgements

We are grateful to Professor Jan Chomicki and the anonymous reviewers for helpful comments and suggestions.

References

[1] K.R. Apt and M.H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM*, 29(3):841–862, 1982.

[2] Y. Auffray and P. Enjalbert. Modal theorem proving: An equational viewpoint. *Journal of Logic and Computation*, 2(3):247–297, 1992.

[3] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

[4] L. Bachmair and H. Ganzinger. Resolution theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 19–99. Elsevier, 2001.

[5] F. Bry and A.H. Yahya. Positive unit hyperresolution tableaux and their application to minimal model generation. *Journal of Automated Reasoning*, 25(1):35–82, 2000.

[6] C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.

[7] Hans de Nivelle. Resolution games and non-liftable resolution orderings. In *L. Pacholski and J. Tiuryn, editors, Proceedings of CSL 1994, LNCS 933*, pages 279–293. Springer, 1995.

[8] C. Fermüller, A. Leitsch, U. Hustadt, and T. Tammet. Resolution decision procedures. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 1791–1849. Elsevier, 2001.

[9] J. Goubault-Larrecq. A note on the completeness of certain refinements of resolution. Technical Report LSV-02-8, Lab. Specification and Verification, ENS de Cachan, Cachan, France, July 2002.

[10] R.A. Kowalski. Predicate logic as a programming language. In J.L. Rosenfeld, editor, *Information Processing 74, Proceedings of IFIP Congress 74*, pages 569–574, 1974.

[11] R.A. Kowalski and P.J. Hayes. Semantic trees in automatic theorem-proving. *Machine Intelligence*, 4:87–101, 1969.

[12] A. Leitsch. *The Resolution Calculus*. Springer, 1997.

[13] J.W. Lloyd. *Foundations of Logic Programming, Second Edition*. Springer-Verlag, 1987.

[14] J. Lobo, A. Rajasekar, and J. Minker. Semantics of Horn and disjunctive logic programs. *Theoretical Computer Science*, 86(1):93–106, 1991.

[15] R. Manthey and F. Bry. SATCHMO: A theorem prover implemented in Prolog. In *E.L. Lusk and R.A. Overbeek, editors, Proceedings of CADE 1988, LNCS 310*, pages 415–434. Springer, 1988.

[16] J. Minker and A. Rajasekar. A fixpoint semantics for disjunctive logic programs. *Journal of Logic Programming*, 9(1):45–74, 1990.

[17] L.A. Nguyen. A fixpoint semantics and an SLD-resolution calculus for modal logic programs. *Fundamenta Informaticae*, 55(1):63–100, 2003.

[18] L.A. Nguyen. Multimodal logic programming and its applications to modal deductive databases. Manuscript, available on Internet at `http://www.mimuw.edu.pl/~nguyen/papers.html`, 2003.

[19] L.A. Nguyen. Negative hyper-resolution as procedural semantics of disjunctive logic programs. In J.J. Alferes and J.A. Leite, editors, *Proceedings of JELIA 2004, LNCS 3229*, pages 565–577. Springer, 2004.

[20] L.A. Nguyen. Negative ordered hyper-resolution as a proof procedure for disjunctive logic programming. Manuscript, available on Internet at `http://www.mimuw.edu.pl/~nguyen/papers.html`, 2004.

[21] A. Nonnengart. *A Resolution-Based Calculus for Temporal Logics*. PhD thesis, Universität des Saarlandes, Saabrücken, Germany, 1995.

[22] H.J. Ohlbach. A resolution calculus for modal logics. In E.L. Lusk and R.A. Overbeek, editors, *Proceedings of CADE-88, LNCS 310*, pages 500–516. Springer, 1988.

[23] J.A. Robinson. Automatic deduction with hyper-resolution. *International Journal of Computer Mathematics*, 1:227–234, 1965.

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

[25] M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.