Hyper Tableaux and Disjunctive Logic Programming

Peter Baumgartner · Ulrich Furbach
Universität Koblenz
Institut für Informatik
Rheinau 1
56075 Koblenz
Germany
E-mail: {peter,uli}@informatik.uni-koblenz.de

For disjunctive logic programs (DLPs) there are several proposals for defining interpreters, like the nearHorn-Prolog-Family [Loveland, 1987], SLI-Resolution [Lobo *et al.*, 1992], SLO-Resolution[Rajasekar, 1989], Model Tree construction [Fernandez and Minker, 91], Restart Model Elimination [Baumgartner *et al.*, 1995, Baumgartner and Furbach, 1994]. There have also been different approaches to assign least fixpoints to DLPs.

This paper proves that there exist an efficient proof procedure, namely hyper tableaux, which can be understood as a direct implementation of some of the well known fixpoint iteration techniques. We show how a hyper tableaux refutation can be transformed into a restart model elimination refutation. This result links the bottom-up to a top-down semantics for DLPs, and thus generalizes the standard result in [Lloyd, 1987] saying that any finite iteration of the T-operator for *definite* programs can be simulated top-town in a SLD-refutation.

A different approach to obtain a top-down calculus is to replace all literals in the input clause set by their complements. We demonstrate that in this setting hyper tableaux generalize Rajasekar's SLO-Resolution.

In the next section we give the proof theoretical part of this paper, which is based on the hyper tableaux calculus from [Baumgartner *et al.*, 1996]. In the following two sections we compare this calculus to fixpoint iteration techniques: there is one seminal paper by Minker and Rajasekar [Minker and Rajasekar, 1990] which introduces a consequence operator to define a semantics for positive disjunctive logic programs by fixpoint iteration over states. We will relate hyper tableaux to this iteration. Another approach by Fernandez and Minker ([Fernandez and Minker, 91]), gives a bottom up evaluation of hierarchical disjunctive databases. We will demonstrate, that this approach is a special case of hyper tableaux.

In Section 5 we discuss the relation of hyper tableaux to SLO-resolution, and in Section 6 we relate hyper tableaux to restart model elimination.

1 Preliminaries

In what follows, we assume that the reader is familiar with the basic concepts of first-order logic. A *clause* is a multiset of literals, usually written as the disjunction $A_1 \vee \cdots \vee A_m \vee \neg B_1 \vee \cdots \vee \neg B_n$ or the implication $A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$ ($m \geq 0$, $n \geq 0$). As usual, the variables occurring in clauses are considered implicitly as being universally quantified, a clause is considered logically as a disjunction of literals, and a (finite) clause set is taken as a conjunction of clauses. A ground clause is a clause containing no variables. Literal K is an *instance* of literal L, written as $K \geq L$ or $L \leq K$, iff $K = L\gamma$ for some substitution γ. Let $\overline{L}$ denote the complement of a literal L. Two literals L and K are *complementary* if $\overline{L} = K$.

Let X be a literal or a clause. X^g is the set of all ground instances of X (wrt. a given signature which contains at least one constant symbol). Similarly, if $\mathcal{X}$ is a clause set or literal set, then $\mathcal{X}^g := \bigcup_{X \in \mathcal{X}} X^g$.

Definition 1.1 (Literal tree, Clausal Tableau) [Letz et al., 1994])
A *literal tree* is a pair (t, λ) consisting of a finite, ordered tree t and a labeling function λ that assigns a literal to every non-root node of t. The *empty tree* contains one single node only.

The *successor sequence* of a node N in an ordered tree t is the sequence of nodes with immediate predecessor N, in the order given by t.

A *(clausal) tableau* T of a set of clauses $\mathcal{S}$ is a literal tree (t, λ) in which, for every successor sequence $N_1, \ldots, N_n$ of a node N in t labeled with literals $K_1, \ldots, K_n$, respectively, there is a substitution σ and a clause $\{L_1, \ldots, L_n\} \in \mathcal{S}$ with $K_i = L_i\sigma$ for every $1 \leq i \leq n$. $\{K_1, \ldots, K_n\}$ is called a *tableau clause (below N)* and the elements of a tableau clause are called *tableau literals*. If δ is a substitution, then $T\delta$ denotes the literal tree which is obtained from literal tree T by application of δ to the labels of all nodes of T. That is, if $T = \langle t, \lambda \rangle$ then $T\delta = \langle t, \lambda' \rangle$, where $\lambda'(N) = (\lambda(N))\delta$ for every node N in T.

We define for a given clausal tableau T the extension of branch b with clause C (or, equivalently, we say that b *is extended by* C) to be the tableau T' which is the same as T, except that T' contains the tableau clause C below the leaf node of b. □

Definition 1.2 (Branch, Open and Closed Tableau, Selection Function)
A *branch* of a tableau T is a finite sequence $N_0, \ldots, N_n$ $(n \geq 0)$ of nodes in T such that N_0 is the root of T, N_i is the immediate predecessor of N_{i+1} for $0 \leq i < n$, and N_n is a leaf of T. We say branch $b = N_0, \ldots, N_n$ is a *prefix* of branch c, written as $b \leq c$ or $c \geq b$, iff $c = N_0, \ldots, N_n, N_{n+1}, \ldots, N_{n+k}$ for some nodes $N_{n+1}, \ldots, N_{n+k}$, $k \geq 0$. Branch concatenation is written with ",", as in (b_1, b_2) or (b, N) (throughout this paper the letter b is used for branches, and the letter N is used for nodes).

The *branch literals* of branch $b = N_0, \ldots, N_n$ are the set $\mathrm{lit}(b) = \{\lambda(N_1), \ldots \lambda(N_n)\}$. We find it convenient to use a branch in a place where a literal set is required to mean its branch literals. For instance, we will write expressions like $A \in b$ instead of $A \in \mathrm{lit}(b)$. Further, we will confuse a node with its label and write, for instance "(b, L)", where L is a literal, instead of "(b, N), where $\lambda(N) = L$"; also, we say the "node L" instead of "the node labeled with L".

In order to memorize the fact that a branch contains a contradiction, we allow to label a branch as *closed*; branches which are not labeled as closed are said to be *open*. A tableau is *closed* if each of its branches is closed, otherwise it is *open*.

A *selection function* is a total function f which maps an open tableau to one of its open branches. If $f(T) = b$ we also say that b *is selected in* T *by* f. □

Fortunately, there is no restriction on which selection function to use. For instance, one can use a selection function which always selects the "leftmost" branch.

Definition 1.3 (Branch Semantics)
Let $\mathcal{L}$ be a possibly infinite set of literals. Define $\mathcal{L}^\forall := \{\forall L | L \in \mathcal{L}\}$ as the *clause set* of $\mathcal{L}$, where $\forall F$ denotes the universal closure of formula F. Whenever we take an atom set $\mathcal{A}$ where a set of formulae were required, we implicitly assume its clause set $\mathcal{A}^\forall$. By the *model of an atom set* $\mathcal{A}$ we mean the minimal Herbrand

model of $\mathcal{A}^\forall$ which we denote by $[\![\mathcal{A}]\!]$. Using a previous convention, we thus identify in particular a branch b with the clause set $(\mathrm{lit}(b))^\forall$. Hence, it is meaningful to say that a branch b is unsatisfiable, and also $[\![b]\!] \models C$ is defined (the least Herbrand model of the clause set of b satisfies the clause C). □

2 Hyper Tableaux

We are going to define the calculus of hyper tableaux as given in [Baumgartner *et al.*, 1996]. For this, we need one more preliminary definition.

Definition 2.1 (Pure clause)
A clause $C = A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$ is called *pure* iff variables are not spread over distinct head literals, i.e. iff $\mathrm{Var}(A_i) \cap \mathrm{Var}(A_j) = \emptyset$, for $i, j \in \{1, \ldots, m\}$ and $i \neq j$. A substitution π is a *purifying substitution* for C iff $C\pi$ is pure. □

Obviously, every non-pure clause can be turned into a pure instance thereof by application of an appropriate substitution.

Definition 2.2 (Hyper tableau)
Let S be a finite set of clauses and f be a selection function. *Hyper tableaux* for S are inductively defined as follows:
Initialization step: The empty tree is a hyper tableau for S. Its single branch is marked as "open".
Hyper extension step: If

1. T is an open hyper tableau for S, $f(T) = b$ (i.e. b is selected in T by f), where b is an open branch, and
2. $C = A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$ is a clause from S ($m \geq 0$, $n \geq 0$), called *extending clause* in this context, and
3. σ is a most general substitution[1] such that $[\![b]\!] \models \forall(B_1 \wedge \cdots \wedge B_n)\sigma$ (referred to as *hyper condition*), and
4. π is a purifying substitution for $C\sigma$,

then the literal tree T' is a hyper tableau for S, where T' is obtained from T by extending b by C, and then marking every new branch $(b, (A_1\sigma\pi)), \ldots, (b, (A_m\sigma\pi))$ with positive leaf as "open", and marking every new branch $(b, (\neg B_1\sigma\pi)), \ldots, (b, (\neg B_n\sigma\pi))$ with negative leaf as "closed".

We will write the fact that T' can be obtained from T by a hyper extension in the way defined as $T \vdash_{b,C,\sigma,\pi} T'$, and say that C is *applicable* to b (or T). Note that the selection function does not appear explicitly in this relation; instead we prefer to let f be given implicitly by the context. □

Note that in the hyper extension step we do *not* take new variants, and that the substitution $\sigma\pi$ is *not* applied to the whole tableau but only to the extending clause. Condition 3, the *hyper condition*, expresses that *all* (instantiated) body literals have to be satisfied by the

[1] Here, "most general" means that whenever $[\![b]\!] \models \forall(B_1 \wedge \cdots \wedge B_n)\delta$ for some substitution δ, then $\sigma \leq \delta\ [\mathrm{Var}(B_1 \wedge \cdots \wedge B_n)]$. The notation $\sigma \leq \delta\ [V]$ means the restriction of the "more general" relation $\leq$ to the variables V. See [Siekmann, 1989].

branch to be extended. This similarity to hyper *resolution* [Robinson, 1965] coined the name "hyper tableaux".

Expressing the hyper condition slightly differently, we mark a branch as "closed" if and only if it is unsatisfiable. For instance, a branch containing literals $P(x)$ and $\neg P(y)$ is closed. In the standard tableaux with rigid variables (e.g. in [Fitting, 1990]) a branch is considered as closed if it contains a complementary pair of literals (notice that $P(x)$ and $\neg P(y)$ are not complementary). Of course, these notions coincide in the ground case.

The need for a purifying substitution in condition 4 in the hyper extension step will guarantee the soundness of hyper tableaux calculi (i.e. if a clause set S admits a refutation then S is unsatisfiable). The underlying property is the easily provable observation that $\forall(A \vee B) \equiv (\forall A \vee \forall B)$ holds if clause $A \vee B$ is pure. The substitutions σ and π have to be applied in this order because if applied in the other order, there is no guarantee that the resulting instance of the extension clause is pure. This would destroy soundness.

Example 2.3 For illustration consider the single-literal branch $b = r(f(X))$ and the clause $C = p(X), q(X,Y) \leftarrow r(X)$. Then, $[\![b]\!] \models \forall r(X)\sigma$, where $\sigma = \{X \leftarrow f(X')\}$. The head $(p(X), q(X,Y))\sigma = p(f(X')), q(f(X'),Y)$ is impure. Taking e.g. a purifying substitution $\pi = \{X' \leftarrow a\}$ enables a hyper extension step, yielding the hyper tableau whose two open branches are $b_1 = (r(f(X)), p(f(a)))$ and $b_2 = (r(f(X)), q(f(a),Y))$. Now, the intended model candidates for the input clause set are just $[\![b_1]\!]$ or $[\![b_2]\!]$. It is important to note that the models are derived "locally" from the paths *alone*, but not from the whole tableaux. However, for this construction to be sound we have to require that $\forall b_1 \vee \forall b_2$ is a logical consequence of $\forall b$, which indeed holds due to the application of π. □

We return to Definition 2.2. The hyper condition in hyper extension step is — intentionally – given in a pure semantical way. With view to a proof procedure it is mandatory to decide (and not only to semi-decide) whether a clause C and most general substitution σ as required exist. Fortunately, this is possible:[2]

Proposition 2.4 (Implementing the Hyper Condition)
For every finite atom set $\mathcal{A}$ and conjunction of atoms $C = B_1 \wedge \cdots \wedge B_n$: if there is a substitution γ for C such that $[\![\mathcal{A}]\!] \models \forall(B_1 \wedge \cdots \wedge B_n)\gamma$ then there is a SLD resolution refutation of the clause set $P = \mathcal{A} \cup \{\neg B_1 \vee \ldots \vee \neg B_n\}$ with computed answer $\sigma \leq \gamma\ [Var(C)]$ and using exactly $|C|$ resolution steps. If there is no such γ, then each of the finitely many SLD derivations of P finitely fails.

Notice that the input clause set for SLD resolution is very simple: it consists of only one negative clause and some positive unit clauses. We prefer this formulation over the unit hyper resolution procedure in [Chang and Lee, 1973] because its answer completeness result gives us immediately that σ is a *most general* substitution as required in the hyper condition.

The hyper extension step has the property that a branch is closed if and only if it ends in a negative literal. Thus it holds:

Proposition 2.5
Every hyper tableau is a clausal tableau where every inner node is labeled with a positive literal. The converse does in general not hold.

[2] The missing proofs for this section are contained in the long version of [Baumgartner *et al.*, 1996]. It can be obtained in the WWW using the URL `http://www.uni-koblenz.de/universitaet/fb4/publications/GelbeReihe/RR-8-96.ps.gz`

Definition 2.6 (Hyper Tableaux Derivation)
Let $\mathcal{S}$ be a finite clause set, called the *set of input clauses*, and let f be a selection function. A (possible infinite) sequence $T_1, \ldots, T_n, \ldots$ of hyper tableaux for $\mathcal{S}$ is called a *hyper tableaux derivation from* $\mathcal{S}$ iff T_1 is obtained by an initialization step, and for $i > 1$, $T_{i-1} \vdash_{b_{i-1}, C_{i-1}, \sigma_{i-1}, \pi_{i-1}} T_i$ for some clause $C_{i-1} \in \mathcal{S}$, and some substitutions σ_{i-1} and π_{i-1}. This is also written as

$$T_1 \vdash_{b_1, C_1, \sigma_1, \pi_1} T_2 \cdots T_n \vdash_{b_n, C_n, \sigma_n, \pi_n} T_{n+1} \cdots$$

A hyper derivation is called a *hyper tableaux refutation* if it contains a closed tableau. □

Note that extension steps are no longer applicable to a closed hyper tableau. Figure 1 shows an example refutation.

(1) $r(a), r(f(Z)) \leftarrow$
(2) $p(X), q(X,Y) \leftarrow r(X)$
(3) $p(X) \leftarrow q(X,Y)$
(4) $\leftarrow p(a)$
(5) $\leftarrow p(f(b))$

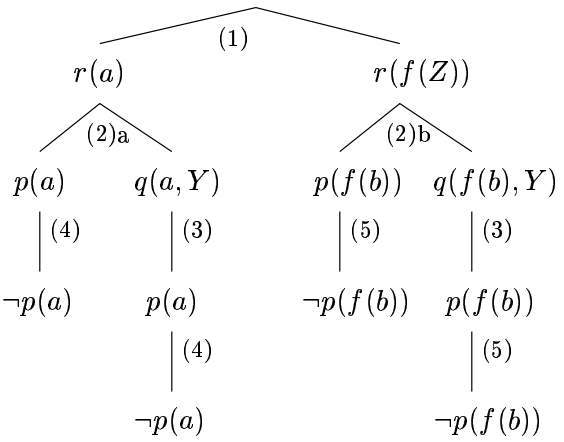

Figure 1: A sample hyper tableaux refutation. The clause set is given by clauses (1)–(5). Variables are written in capital letters. The usage of the clauses in extension steps is indicated at the edges. The initial tableaux is set up with clause (1) (there is no other choice). Extension at $r(a)$ with clause (2) uses $\sigma = \{x \leftarrow a\}$; since Y is pure in the resulting head, we can choose $\pi = \varepsilon$ which leaves us Y as a universal variable. This extension step is indicated as "(2)a" (the body literals are not depicted). The further refutation of the left subtree should be obvious. For the right subtree we can extend $r(f(Z))$ with clause (2) again: first compute $\sigma = \{x \leftarrow f(Z)\}$. The resulting head $p(f(Z)) \vee q(f(Z), Y)$ of clause (2) is not pure; we guess $\pi = \{Z \leftarrow b\}$ in order to find the refutation immediately.

For a further discussion of this calculus and especially to its relation to resolution we refer the reader to [Baumgartner *et al.*, 1996].

3 States

In this section we relate hyper tableaux to the fixpoint semantics from [Lobo *et al.*, 1992]. Let P be a set of ground clauses which contains no purely negative clause; we call P a set of program clauses.

DHB, the disjunctive Herbrand base for P is the set of disjunctions, that can be formed by atoms of the Herbrand base of P. The transformation $\Gamma^S : 2^{DHB} \to 2^{DHB}$ is given by[3]

$$\Gamma^S(D) = \{C \in DHB \mid C' \leftarrow B_1, \ldots, B_n \in P, \quad \forall 1 \leq i \leq n : \; B_i \vee C_i \in D, \\ C'' = C_1 \vee \ldots \vee C_n \vee C', \quad C \text{ is the smallest factor of } C''\}$$

[3]Obviously, this operator is dependent of the program P; we assume this will be clear from the context.

A *state* for P is a subset of DHB. An *expanded state* ST for P is a state, such that $ST = exp(ST)$, where

$$exp(ST) = \{C \in DHB \mid C \in ST \text{ or } \exists C' \in ST : C' \text{ is a subclause of } C\}$$

In [Lobo *et al.*, 1992] it is shown that the operator Γ^S is continuous, hence its least fixpoint exists and the *lfp*-Operator yields $\Gamma^S\uparrow\omega$.

We do not introduce model states explicitly, moreover we use their characterization as fixpoints and logical consequences:

Theorem 3.1 (Lobo et al. 92)
Let P be a set of program clauses and $C \in DHB$. Then $P \models C$ iff $C \in exp(lfp(\Gamma^S))$.

In the following we assume S to be a minimal unsatisfiable set of ground clauses and T to be a hyper tableau for S. S^+ is this set without the purely negative clauses. Then S^+ only consists of program clauses and is consistent. Furthermore we assume that S is in goal normal form. By this we mean the transformation of every clause of the form $\leftarrow B_1, \ldots, B_n$ into $G \leftarrow B_1, \ldots, B_n$, where G is a new predicate symbol, and furthermore adding the clause $\neg G$ as the only goal.

A clause C is a *cut of* T iff

$$\forall b \text{ branch of } T : \exists N \in b : \exists L \in C \text{ such that } \lambda(N) = L, \text{ and} \tag{1}$$

$$\forall L \in C : \exists b \text{ branch of } T : \exists N \in b \text{ such that } \lambda(N) = L. \tag{2}$$

The following lemma relates hyper tableaux derivations to Γ^S-iterations.

Lemma 3.2
For every i and $C \in \Gamma^S\uparrow i$ there is a hyper tableau T such that there is a cut C' of T were C is the smallest factor of C'.

Proof. Induction on i.

Induction start $i = 1$: The set $\Gamma^S\uparrow 0$ contains all smallest factors of (disjunctive) facts from S^+. Let $A_1, \ldots, A_n \leftarrow$ be such a fact, then construct a hyper tableau with an initial step and a hyper extension step using this fact. The cut $A_1, \ldots, A_n \leftarrow$ of this tableau has the desired property.

Induction step $i \rightarrow i+1$: If $C \in \Gamma^S\uparrow i+1$, we know by definition of Γ^S, that C is the smallest factor of $C_1 \vee \ldots \vee C_n \vee C'$, such that $C' \leftarrow B_1, \ldots, B_n \in S^+$ and $\{B_1 \vee C_1, \ldots, B_n \vee C_n\} \subseteq \Gamma^S\uparrow i$. Induction hypothesis gives us, that there exists hyper tableaux $T_1, \ldots, T_n$, such that $B_j \vee C_j$ is the smallest factor of a cut C'_j of T_j.

All we have to do is to link these tableaux together to one hyper tableau: Select a branch b_j from tableau T_j which contains the literal B_j. Take the leaf N_j of this branch and use it as the new root of the tableau T_{j+1}. The result is again a hyper tableau. If this linking is done for j from 1 to $n-1$ we get a hyper tableau which fulfills by construction the stated property. Q.E.D.

As an example, assume the following set of clauses $S^+ = \{b \vee c, a \leftarrow b, a \leftarrow c\}$. There is a hyper tableau which consists of the two branches with $\{b, a\}$ and $\{c, a\}$ as sets of labels of its nodes. A set of cuts of this tree is $\{a \vee a, a \vee b, b \vee c, a \vee c\}$. Note that there is no sequence assumed in which the literals from different branches have to occur within a cut.

The iteration of the Γ^S-operator gives $\Gamma^S\uparrow 0 = \{b \vee c\}$, $\Gamma^S\uparrow 1 = \{b \vee c, a \vee c, a \vee b\}$ and $\Gamma^S\uparrow 2 = \{b \vee c, a \vee c, a \vee b, a\} = \Gamma^S\uparrow \omega$.

Having these close relation of hyper tableaux and the fixpoint iteration over states we can use this result to prove completeness of ground hyper tableaux. Note that a proof for full first order clauses is given in [Baumgartner *et al.*, 1996], which includes a fairness consideration. Here we only want to establish the close relationship of the approaches.

Theorem 3.3 (Completeness of Hyper Tableaux)
For every unsatisfiable ground clause set S in goal normal form there is a closed hyper tableau for S.

Proof. Since S is a minimal unsatisfiable set of clauses, we know by compactness that S is finite and that S^+ is consistent. The only clause which is not contained in S^+ is the goal clause $\neg G$, and hence we know that $S^+ \models G$. From theorem 3.1 we learn that $G \in exp(lfp(\Gamma^S))$. Since G is an atom it must be contained in $lfp(\Gamma^S)$ alone. From continuity of the Γ^S-operator we conclude that there is an i such that $G \in \Gamma^S\uparrow i$.

Lemma 3.2 gives us the existence of a hyper tableau T with a cut C', such that G is the smallest factor of C'. Hence C' has the form $G \vee \ldots \vee G$; in other words every branch of the tableau contains the literal G. This tableau can be closed by using the goal clause $\neg G$.
Q.E.D.

4 Model Trees

The other approach we want to relate hyper tableaux to, is that of bottom up evaluation of disjunctive deductive databases. In [Fernandez and Minker, 91] and [Furbach, 1992] a bottom up consequence operator Γ^M for disjunctive deductive databases is given which acts on sets of interpretations, thus yielding models for the given set of clauses. In [Seipel *et al.*, 1995] this approach is related to the consequence operator on states which we discussed above. Fernandez and Minker also introduce model trees as a calculus to compute this operator, this is done in detail in [Lobo *et al.*, 1992]. We will demonstrate that this is related closely to hyper tableaux calculus.

The consequence operator Γ^M over sets of Herbrand interpretations is given by

$$\begin{aligned} \Gamma^M &: 2^{2^{HB}} \to 2^{2^{HB}} & \Gamma^M(\mathcal{I}) &= \min(\Gamma^{INT}(\mathcal{I})) \\ \Gamma^{INT} &: 2^{HB} \to 2^{HB} & \Gamma^{INT}(\mathcal{I}) &= \bigcup_{I \in \mathcal{I}} MOD(\Gamma^S(I)) \end{aligned}$$

where *MOD* gives all models of a state and min filters out the minimal models. The latter operator looks harmless; it will turn out later in this paper, that this is a rather important step. Its definition is given by:

$$\min(\mathcal{I}) = \{I \in \mathcal{I} \mid \neg\exists J \in \mathcal{I} : J \subset I\}$$

In [Baumgartner *et al.*, 1996] we gave a proof that the branches of a hyper tableaux correspond to partial models of the program and in particular that in fair derivations branches correspond to models.

In the following we additionally depict the relation between one step with the Γ^M operator and hyper extension.

Definition 4.1
Let T be a hyper tableau and b an open branch. A complete extension of T at b is a tree T', which can be obtained from T by applying every possible extension step with expanding clauses C from P, such that

- only branches b' are selected, which contains b as a prefix,
- only literals from b are used for an extension step,
- if a clause C is the expanding clause for a branch b' it has to be used for expansion for all other branches b', which contain b as a prefix. After this the clause C cannot be used for further extension steps.

□

The last condition could be strengthened by using a well known strong regularity condition ("no literal occurs more than once in a branch"). For our purposes the above somewhat weaker formulation is sufficient.

The following lemma establishes the connection of partial branches, i.e. models from a hyper tableaux to the iterations using the Γ^Moperator.

Lemma 4.2
Let T be a hyper tableau consisting of one single branch b and let T' be a complete extension of T at b. Then $\{[\![b']\!] \mid b' \in T'\} \subset MOD(\Gamma^S(lit(b)))$

5 SLO-Resolution

In [Rajasekar, 1989] SLO-Resolution is introduced as a generalization of SLD-Resolution. This interesting approach offers a goal-directed approach for the interpretation of positive disjunctive programs. In a subsequent paper, Rajesakar and Yusuf offer a modification of the WAM for an implementation. However there is one missing point: there is only a ground completeness result; e.g. SLO-resolution can not answer the query $\leftarrow p(x)$ with respect to the program $p(a), p(b) \leftarrow$. In this section we will demonstrate, that hyper tableau can be easily used to simulate SLO-resolution, by simply inverting the signs of all literals. We do not claim that this transformation is original, it has been used e.g. in [Yahja, 1996] to turn a bottom-up prover into a goal-directed top-down one; moreover we want to point out that this simple technique can be used to simulate and to extend SLO-resolution.

The following definitions are taken from [Rajasekar, 1989]

A *goal* for a disjunctive program is of the form $\leftarrow (C_1, \ldots, C_n)$, where$n \geq 0$ and the C_i are positive clauses.

Definition 5.1
Let P be a positive disjunctive logic program and let G be a goal. An SLO-derivation from P with goal G consists of a (finite or infinite) sequence of goals $G_0 = G, G_1, ..$, such that for all $i \geq 0$, G_{i+1} is obtained from $G_i = \leftarrow (C_1, \ldots, C_m, \ldots, C_k)$ as follows:

1. C_m is a clause in G_i. C_m is called the selected clause.
2. $C \leftarrow B_1, \ldots, B_q$ is a program clause in P.
3. C subsumes C_m with most general unifier θ.

4. G_{i+1} is the goal $\leftarrow (C_1, \ldots, C_{m-1}, B_1 \vee C_m, \ldots, B_q \vee C_m, C_{m+1}, \ldots, C_k)\theta$

As usual derivations of the empty clause from G using P are called refutations; one also says that the goal G succeeds for P.

□

The following ground completeness theorem is proven by induction over the fixpoint operator Γ^S.

Theorem 5.2
Let P be a disjunctive logic program and let C be a ground clause. If C is a logical consequence of P then there is an SLO-refutation from P with goal $\leftarrow C$.

Without loss of generality we assume in the following only goals of the form $\leftarrow C$ where C is a positive disjunction. Note that a negative clause $\leftarrow A_1, \ldots, A_n$ is different from a goal $\leftarrow A_1 \vee \ldots \vee A_n$; the latter is standing for a set of negative units.

Definition 5.3
The dual P^d of a clause $P = A_1, \ldots, A_n \leftarrow B_1, \ldots, B_m$ is obtained by inverting the arrow, i.e. $P^d = B_1, \ldots, B_m \leftarrow A_1, \ldots, A_n$. This could be alternatively formulated, by saying that signs of every literal in $P = A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_m$ are complemented to get $P^d = \neg A_1 \vee \ldots \vee \neg A_n \vee B_1 \vee \ldots \vee B_m$.

Note that the dual of a goal $G = \leftarrow A_1 \vee \ldots \vee A_n$ is the set of clauses $\{A_1 \leftarrow, \quad \ldots, \quad A_n \leftarrow\}$, since G , written in clause form is the set of negative units $\{\leftarrow A_1, \quad \ldots, \quad \leftarrow A_n\}$. This transformation is extended to set of clauses in an obvious way. □

Lemma 5.4
A clause set S is unsatisfiable iff S^d is unsatisfiable.

Proof. It only requires the trivial observation that if I is a model for S then I^d is a model for S^d, where $I^d(L) = \overline{I(L)}$ for every literal L. Since the ^{d}-operator is involutory the other direction follows. Q.E.D.

Example 5.5 The following example demonstrates how a SLO-derivation of P with goal G can be seen as a notation for a hyper tableau derivation of $P \cup \{G\}$:

P:	$A, B \leftarrow C, D$	(1)
	$B, C \leftarrow$	(2)
	$A, D \leftarrow$	(3)
G:	$\leftarrow A \vee B$	(4)

An SLO-refutation starting with G is as follows:

$\leftarrow A \vee B$		(5)
$\leftarrow C \vee A \vee B, \quad D \vee A \vee B$	from 4) and 1)	(6)
$\leftarrow D \vee A \vee B$	from 6) and 2)	(7)
$\leftarrow$	from 7) and 3)	(8)

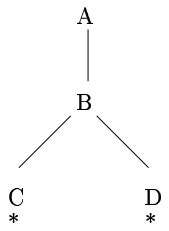

Figure 2: A hyper tableaux refutation of $P \cup \{G\}$. The negative leaf nodes are not displayed.

In order to use the hyper tableau calculus to simulate this derivation we construct the dual program

$$P^d: \quad C, D \leftarrow A, B \tag{9}$$
$$\leftarrow B, C \tag{10}$$
$$\leftarrow A, D \tag{11}$$
$$\tag{12}$$

and from the goal G we get two dual clauses

$$A \leftarrow \tag{13}$$
$$B \leftarrow \tag{14}$$

Applying hyper tableaux to this clause set gives a closed tableau depicted in 2. The starting goal 5 in the SLO-refutation corresponds to the fist two extension steps with the two facts $A \leftarrow$ and $B \leftarrow$ from the dual program P^d, resulting in the tableau with the two nodes A and B. The SLO-step yielding in line 6 the goal $\leftarrow C \vee A \vee B, \quad D \vee A \vee B, \quad$ corresponds to a hyper extension with $C, D \leftarrow A, B$. The two branches from the tableau in figure 2 are coded in line 6 by the two clauses in the goal. The step resulting in goal 7 corresponds to the extension step with $\leftarrow B, C$ and the last step to the extension with $\leftarrow A, D$. □

Lemma 5.6
Given a ground SLO-derivation from P with ground goal $\leftarrow C$ and derived goal $\leftarrow C_1, \ldots, C_m$. Then there is a hyper tableau T for P^d and a substitution σ such that for all $b \in T$ there is a C_i containing $L\sigma$, for any label L from b.

Based on the previous lemma we are currently investigating how SLO-resolution can be improved by applying the concepts of hyper tableaux. By this it is possible to make SLO-resolution complete with respect to logical consequences and to get rid of some of the rigidly treated variables.

6 Hyper Tableaux and Restart Model Elimination

In this section we will provide a procedural, top-down semantics for hyper tableaux. We do so by relating hyper tableaux to the restart model elimination (RME) calculus of [Baumgartner and Furbach, 1994].

Unlike hyper tableaux, RME is a *top-town* calculus, i.e. derivations start with a (negative) goal clause and ends at the (positive) facts. Our main result below shows how any closed

hyper tableau can be transformed into a RME refutation. This transformation will essentially "reverse" a hyper tableau from the leaves to the root, where a splitting in hyper tableau corresponds to a "restart step" in RME.

This result is in close relationship to the standard result in [Lloyd, 1987] saying that any finite iteration of the T-operator over *definite* programs can be simulated top-town in a SLD-refutation. In fact, we generalize this result to the non-Horn case.

6.1 Restart Model Elimination

We will briefly review the RME calculus as presented in [Baumgartner and Furbach, 1994]. However, for ease of presentation we will use a slightly different notation in the style of Definitions 1.1 and 2.2.

Definition 6.1 (Restart Model Elimination)
Let S be a finite set of clauses and f be a selection function. *Restart Model Elimination (RME) tableaux* for S are inductively defined as follows:
Initialization step: A clausal tableau obtained by extending the root node of the empty tree by a negative clause $\leftarrow B_1, \ldots, B_n \in S$ is a hyper tableau for S. All branches are marked as "open".
Linked extension step: If

1. T is an open RME tableau for S, $f(T) = b$ (i.e. b is selected in T by f) with negative open leaf node $\neg A$, and
2. $C = A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$ is a new variant of a clause from S ($m \geq 1$, $n \geq 0$), called *extending clause* in this context, and
3. σ is a most general unifier for A and some A_i (where $1 \leq i \leq m$),

then the literal tree $T'\sigma$ is a hyper tableau for S, where T' is obtained from T by extending b by C, and then marking the new branches

$$(b, A_1), \ldots, (b, A_{i-1}), (b, A_{i+1}), \ldots, (b, A_{i-1}), \ldots, (b, \neg B_1), \ldots, (b, \neg B_n)$$

as "open", and marking the new branch (b, A_i) as "closed".
Reduction step: If

1. T is an open RME tableau for S, $f(T) = b$ (i.e. b is selected in T by f) with negative open leaf node $\neg A$, and
2. $A' \in b$ is a positive literal in b, and
3. σ is a most general unifier for A and some $\neg A'$,

then the literal tree $T'\sigma$ is a hyper tableau for S, where T' is obtained from T by marking b as "closed".
Restart step: If

1. T is an open RME tableau for S, $f(T) = b$ (i.e. b is selected in T by f) with positive open leaf node A, and
2. $C = \leftarrow B_1, \ldots, B_n$ is a new variant of a negative clause from S ($n \geq 0$),

then the literal tree T' is a hyper tableau for S, where T' is obtained from T by extending b by C.

The notion of *derivation* and *refutation* is taken from Definition 2.6. □

Consider the clause set in Example 5.5 again. Figure 3 contains a RME refutation.

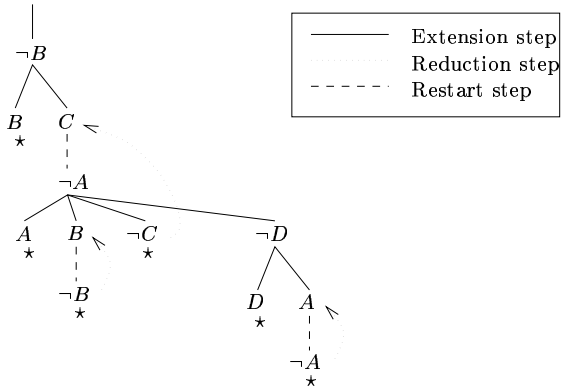

Figure 3: A RME refutation of the clause set of Example 5.5. Notice that the goal $\leftarrow A \vee B$ stands for the two clauses $A \leftarrow$ and B.

6.2 Mapping Hyper Tableaux to Restart Model Elimination

Theorem 6.2 (Top-Down Semantics for Hyper Tableaux)
Let T_H be a closed hyper tableau containing the tableau clauses S. Let $G = \leftarrow B_1, \ldots, B_n$ be a tableau clause in T_H (which closes a branch). Then there is a RME refutation of S with goal clause G.

Proof. Let S_H be the multiset of tableau clauses occurring in T_H. Let $k(S_H)$ denote the number of occurrences of positive literals in S_H minus the number of non-negative clauses[4] in S_H ($k(S_H)$ is a measure for the "Hornness" of S_H; it is related to the well-known *excess literal parameter*). Now we prove the claim by induction on $k(S_H)$.

Base case: $k(S_H) = 0$. S_H and thus also S must be a set of Horn clauses. Apply Lemma 6.3.

Induction step: $k(S_H) > 0$. As the induction hypothesis assume the result to hold for closed hyper tableau for clause sets S'_H satisfying $k(S'_H) < k(S_H)$. Figure 4 depicts the proof.

Some ancestor node A of the tableau clause $G = \leftarrow B_1, \ldots, B_n$ must have one or more positive brother nodes, because otherwise S_H would be a Horn multiset. Let $C = (A_1, \ldots, A_m, A \leftarrow \mathcal{B}) \in S_H$ be the tableau clause where the node A is contained in. Here, $\mathcal{B}$ is understood as a (possibly empty) sequence of positive literals. Below we will also write expressions like $\neg\mathcal{B}$ and mean the clause $\bigvee_{B \in \mathcal{B}} \neg B$.

We split T_H in $m+1$ closed hyper tableaux: the hyper tableaux T_H^A is obtained from T_H by replacing the tableau clause C by $A \leftarrow \mathcal{B}$ (and thus deleting the subtrees below $A_1, \ldots, A_m$), and the hyper tableaux $T_H^{A_i}$ is obtained from T_H by replacing C by A_i (for $i = 1, \ldots, m$).

Let S_H^A and $S_H^{A_i}$ be the tableau clause multisets corresponding to T_H^A and $T_H^{A_i}$. It holds that $k(S_H^A) < k(S_H)$ and $k(S_H^{A_i}) < k(S_H)$. Notice that T_H^A still contains G. Hence, by the induction hypothesis, there is a RME refutation T_{RME}^A[5] of S^A with goal clause G.

Similarly, by applying the induction hypotheses m times we learn that there are RME refutations $T_{RME}^{A_i}$ of S^{A_i} with some respective goal clauses $\leftarrow \mathcal{G}_i \in S^{A_i}$. Since splitting does not affect the negative clauses it holds that $\leftarrow \mathcal{G}_i \in S$. Hence, $T_{RME}^{A_i}$ is a RME refutation of $S \cup \{A_i \leftarrow\}$. Notice that positive unit clauses like $A_i \leftarrow$ can be used in RME refutations only to *close* branches (as indicated in Figure 4).

[4] A *non-negative clause* is a clause containing at least one positive literal.

[5] To be precise, there is a RME refutation *generating* T_{RME}^A; but we will confuse this.

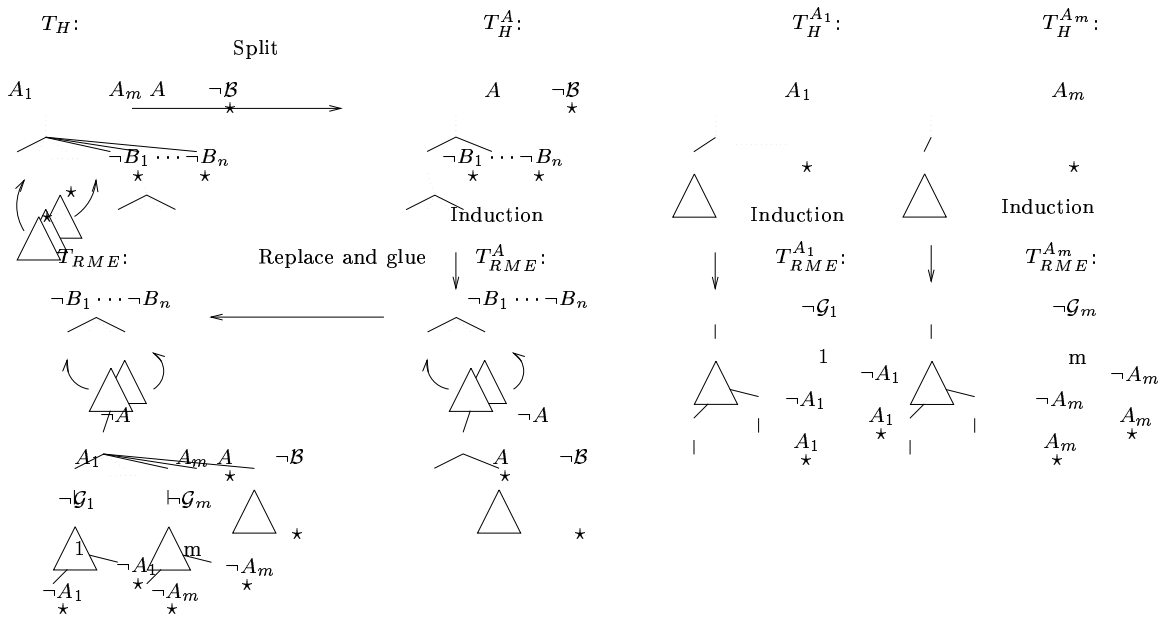

Figure 4: Proof of Theorem 6.2.

Now we can put things together. Consider T^A_{RME} again. It (possible) uses the clause $A \leftarrow \mathcal{B}$. However, this clause is (possibly) not contained in S. In order to turn T^A_{RME} into a RME refutation of S, we first replace every occurrence of the tableau clause $A \leftarrow \mathcal{B}$ in T^A_{RME} by C. This leaves us with open branches ending in (possibly several occurrences of) $A_1, \ldots, A_m$. Now, at each of these branches ending in A_i we can restart with the clause $\leftarrow \mathcal{G}_i$. Then we append below the upcoming tableau clause $\neg\mathcal{G}_i$ the refutation $T^{A_i}_{RME}$ *and* we replace possible extension steps in $T^{A_i}_{RME}$ with $A_i \leftarrow$ by reduction steps to the branch literal A_i where the restart occurred. As a result we get the desired RME refutation T_{RME} of S with goal clause G. Q.E.D.

The next lemma rephrases in our setting the corresponding Result from [Lloyd, 1987], which links the T-operator and SLD-Resolution. It was needed as a special case (induction start) in the proof of the previous Theorem 6.2.

Lemma 6.3
Let T_H be a closed hyper tableau containing the tableau clauses S, which is a Horn clause set. Let $G = \leftarrow B_1, \ldots, B_n$ be a tableau clause in T_H (which closes a branch). Then there is a RME refutation of S with goal clause G.

Proof. Let S_H be the multiset of tableau clauses occurring in T_H. Let $k(S_H)$ denote the number of occurrences of definite clauses in S_H with non-empty body. Now we prove the claim by induction on $k(S_H)$.

Base case: $k(S_H) = 0$. Thus all definite clauses in T_H are positive unit clauses. T_H consists of one single branch p with G fanned below it, and p contains nodes $B_1, \ldots, B_n$. It is easy to find a RME refutation for this case (cf. Figure 5, "Base case").

Induction step: $k(S_H) > 0$. As the induction hypothesis assume the result to hold for closed hyper tableau for clause sets S'_H satisfying $k(S'_H) < k(S_H)$. Figure 5 ("Induction Step")

depicts the proof.

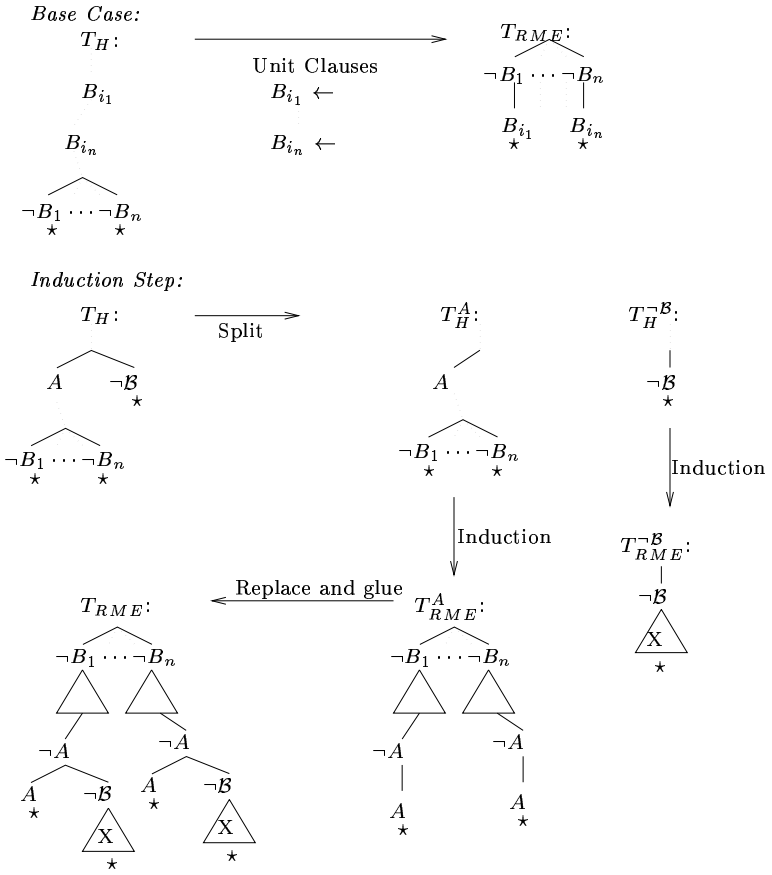

Figure 5: Proof of Lemma 6.3.

Some ancestor node A of the tableau clause $G = \leftarrow B_1, \ldots, B_n$ must have one or more (negative) brother nodes, because otherwise $k(S_H) = 0$. Let $C = (A \leftarrow \mathcal{B}) \in S_H$ be the tableau clause where the node A is contained in. Here, $\mathcal{B}$ is understood as a (possibly empty) sequence of positive literals. Below we will also write expressions like $\neg\mathcal{B}$ and mean the clause $\bigvee_{B \in \mathcal{B}} \neg B$.

We split T_H in 2 closed hyper tableaux: the hyper tableaux T_H^A is obtained from T_H by replacing the tableau clause C by A (and thus deleting the subtrees below $\neg\mathcal{B}$), and the hyper tableaux $T_H^{\neg\mathcal{B}}$ is obtained from T_H by replacing C by $\neg\mathcal{B}$.

Let S_H^A and $S_H^{\neg\mathcal{B}}$ be the tableau clause multisets corresponding to T_H^A and $T_H^{\neg\mathcal{B}}$. It holds that $k(S_H^A) < k(S_H)$ and $k(S_H^{\neg\mathcal{B}}) < k(S_H)$. Notice that T_H^A still contains G. Hence, by the induction hypothesis, there is a RME refutation T_{RME}^A of S^A with goal clause G.

Similarly, by applying the induction hypotheses we learn that there is a RME refutation $T_{RME}^{\neg\mathcal{B}}$ of $S^{\neg\mathcal{B}}$ with goal clauses $\leftarrow \mathcal{B} \in S^{\neg\mathcal{B}}$.

Now we can put things together. Consider T_{RME}^A again. It possible uses the clause $A \leftarrow$ (as extending clause in extension steps). However, this clause is possibly not contained in S. In order to turn T_{RME}^A into a RME refutation of S, we first replace every occurrence of the tableau clause A in T_{RME}^A by C. This leaves us with zero or more open branches labeled with the literals from $\neg\mathcal{B}$. Now, each of these branches can be extended by $T_{RME}^{\neg\mathcal{B}}$. As a result we

get the desired RME refutation T_{RME} of S with goal clause G. Q.E.D.

We consider the result of this Section — Theorem 6.2 — as an initial investigation in the relationship between Hyper tableaux and RME. It would be interesting to investigate the complexity of this mapping and to improve it. Currently each *single* hyper extension step might result in *many* extension and restart steps. It might be possible to improve the situation by additional RME inference rules like factoring.

References

[Baumgartner and Furbach, 1994] P. Baumgartner and U. Furbach. Model Elimination without Contrapositives and its Application to PTTP. *Journal of Automated Reasoning*, 13:339–359, 1994. Short version in: Proceedings of CADE-12, Springer LNAI 814, 1994, pp 87–101.

[Baumgartner *et al.*, 1995] P. Baumgartner, U. Furbach, and F. Stolzenburg. Model Elimination, Logic Programming and Computing Answers. In *14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, volume 1, 1995. (Long version in: Research Report 1/95, University of Koblenz, Germany. To appear in *Artificial Intelligence*).

[Baumgartner *et al.*, 1996] P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *JELIA 96*. European Workshop on Logic in AI, Springer, LNCS, 1996. (Long version in: *Fachberichte Informatik*, 8–96, Universität Koblenz-Landau).

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

[Fernandez and Minker, 91] J.A. Fernandez and J. Minker. Bottom-up evaluation of hierarchical disjunctive deductive databases. In Koichi Furukawa, editor, *Proc. 8th International Conference on Logic Programming*, pages 660–675, 91.

[Fitting, 1990] M. Fitting. *First Order Logic and Automated Theorem Proving*. Texts and Monographs in Computer Science. Springer, 1990.

[Furbach, 1992] U. Furbach. Computing answers for disjunctive logic programs. In Pearce and Wagner, editors, *Logics in AI, JELIA'92*. Springer, LNAI 633, 1992.

[Letz *et al.*, 1994] R. Letz, K. Mayr, and C. Goller. Controlled Integrations of the Cut Rule into Connection Tableau Calculi. *Journal of Automated Reasoning*, 13, 1994.

[Lloyd, 1987] J. Lloyd. *Foundations of Logic Programming*. Symbolic Computation. Springer, second, extended edition, 1987.

[Lobo *et al.*, 1992] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, 1992.

[Loveland, 1987] D.W. Loveland. Near-Horn Prolog. In J.-L. Lassez, editor, *Proc. of the 4th Int. Conf. on Logic Programming*, pages 456–469. The MIT Press, 1987.

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

[Rajasekar, 1989] Arcot Rajasekar. *Semantics for Disjunctive Logic Programs*. PhD thesis, University of Maryland, 1989.

[Robinson, 1965] J. A. Robinson. Automated deduction with hyper-resolution. *Internat. J. Comput. Math.*, 1:227–234, 1965.

[Seipel *et al.*, 1995] D. Seipel, J. Minker, and C. Ruiz. Model generation and state generation for disjunctive logic programs. Technical report, Univ. of Tübingen, 1995.

[Siekmann, 1989] Jörg H. Siekmann. Unification Theory. *Journal of Symbolic Computation*, 7(1):207–274, January 1989.

[Yahja, 1996] Adnan Yahja. Query Answering in Disjunctive Deductive Databases. Dagstuhl-Seminar on *Disjunctive logic programming and databases: Non-monotonic aspects*, 1996.