DisLoP: Towards a Disjunctive Logic Programming System*

Chandrabose Aravindan and Jürgen Dix and Ilkka Niemelä

Department of Computer Science, University of Koblenz
Rheinau 1, D-56075 Koblenz, Germany
{arvind,dix,ini}@informatik.uni-koblenz.de
<URL:http://www.uni-koblenz.de/ag-ki/DLP/>

Abstract. This paper gives a brief high-level description of the implementation of a disjunctive logic programming system referred to as DisLoP. This system is a result of research activities of the **Dis**junctive **Lo**gic **P**rogramming-project (funded by **D**eutsche **F**orschungs-**G**emeinschaft), undertaken by the University of Koblenz since July 1995.

1 Introduction

In this report, we describe the implementation of a disjunctive logic programming system referred to as DisLoP. This system is a result of the research project on Disjunctive Logic Programming undertaken by the University of Koblenz since July 1995. The key idea of this project is to develop semantics and procedures for dealing with disjunctions and non-monotonic negations in logic programming. On the procedural side, instead of starting from scratch, the aim is to exploit the techniques that have been developed for yet another project on *Automated Reasoning* (that was started 3 years ago in Koblenz by Ulrich Furbach as a part of the German "Schwerpunkt-Programm" *Deduction*). One important consequence of this is that the PROTEIN theorem prover [BF94b] developed for Deduction project can be used for (positive) Disjunctive Logic Programming with little modifications. Interested readers are welcome to visit the project home page at `<URL:http://www.uni-koblenz.de/ag-ki/DLP/>` to get more information on this and related projects.

Conforming to the aims of the project, the DisLoP system uses the PROTEIN theorem prover as a basis for implementation. PROTEIN can already handle positive disjunctive logic programs and compute answers for them. We first started with extending PROTEIN to perform minimal model reasoning with positive programs, resulting in two techniques. One, a top-down method, uses abductive reasoning to answer queries wrt Generalized Closed World Assumption. The other method is a bottom-up technique to generate minimal models efficiently.

For disjunctive logic programs with non-monotonic negation, DisLoP computes the disjunctive well-founded semantics (D-WFS) of Brass and Dix. This

* Funded by the DFG ("Deutsche Forschungsgemeinschaft") under grant Fu 263/3-1

semantics is computed in two different ways: one employs a transformation calculus to compute a residual program while the other uses minimal model reasoning for positive program as a step to compute D-WFS.

All these five modules of DisLoP — namely answer computing with postive programs, two methods of minimal model reasoning with positive programs, and two methods of computing D-WFS — are explained in this paper. The reader is assumed to be familiar with various research works on (disjunctive) logic programming [Llo87,LMR92] and non-monotonic reasoning.

2 Positive Disjunctive Programs

For positive disjunctive programs, practically all semantics boil down to minimal model reasoning. With the aim of merging theorem proving and logic programming paradigms [ABD$^+$96,Ara96b] two approaches to handling query evaluation for positive disjunctive programs with respect to minimal model semantics have been studied. Subsection 2.2 discusses the first method which is based on abduction and restart model-elimination ([Ara96a]). The second method is explained in Subsection 2.3: it is based on extending the hyper tableau calculus of [BFN96] (see [Nie96b,Nie96a]).

Before presenting these two methods, we consider in Subsection 2.1 the problem of answering positive queries for positive disjunctive programs. Although this is a monotonic problem, it turns out that most classical theorem provers can only compute trivial answers and are unable to find *definite* (i.e. non-disjunctive) answers.

2.1 Computing Answers using Restart ME

In [BFS95], it has been shown that theorem provers using a special calculus, referred to as *restart model elimination calculus*, can be used as answer complete interpreters for positive disjunctive logic programs. Given a disjunctive logic program D and a query $\leftarrow Q$, a disjunction $Q\theta_1 \vee \cdots \vee Q\theta_n$ of instances of Q is a correct answer iff $P \models \forall\,(Q\theta_1 \vee \cdots \vee Q\theta_n)$. On the procedural side, restart model elimination calculus is used to build a tableau and the disjunction of all the instances of the query in the tableau is given as a computed answer. This process is illustrated by a very simple example below. More details can be obtained from [BFS95]. The main result here is that for every correct answer there exists a computed answer that subsumes it and thus restart model elimination is an answer complete calculus for positive programs with positive queries. With respect to definite answers, the completeness result can be stated as follows: if $Q\theta$ is a correct definite answer, then this calculus can compute an answer $Q\sigma$ s.t. $Q\sigma\delta = Q\theta$ (for some substitution δ). To focus only on definite answers or to compute answers with less number of disjuncts, a variant of restart model elimination referred to as ancestry restart model elimination has been introduced and the details are in [BFS95].

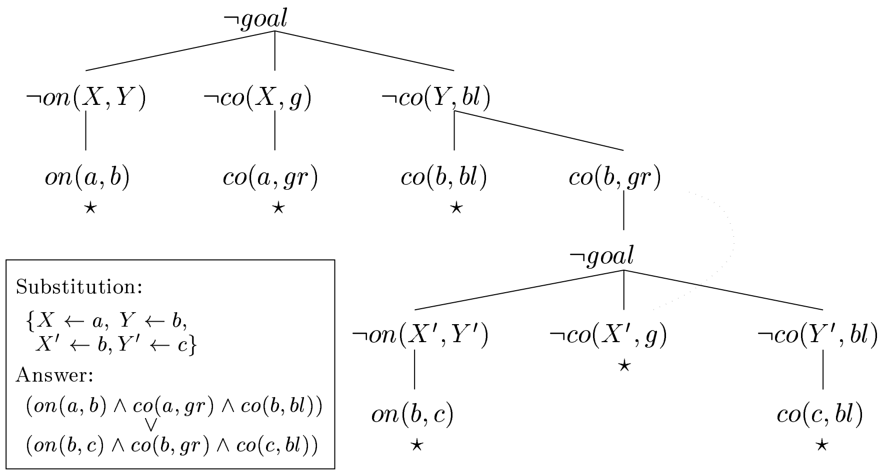

Fig. 1. Answer computing with Restart ME calculus

Example 1. Consider a disjunctive logic program with the following clauses: $on(a,b)$, $on(b,c)$, $co(a,gr)$, $co(c,bl)$, and $co(b,gr) \vee co(b,bl)$. This program describes a block world in which there are three blocks a, b, and c. It is known that a is on b and b is on c. The color of a is green, c is blue, and that of b is either blue or green. Now consider a query $\leftarrow on(X,Y), co(X,gr), co(Y,bl)$. The answer computation process based on restart model elimination is shown in Figure 1. As shown a non-definite answer for the given goal is computed.

2.2 Abduction and Restart ME

In [Ara96a], we have developed an abductive framework for positive disjunctive logic programs that captures the minimal model reasoning wrt both GCWA [Min82] and WGCWA [RLM89,RT88]. Given a program D and a goal G, an abductive explanation Δ for G consists only of negative literals s.t. $D \cup \Delta \models G$ and $D \cup \Delta$ is consistent. We have introduced an inference rule, referred to as *negation by failure to explain*, that allows us to infer negation of an atom A if there is no abductive explanation for A wrt D. It is also shown that negation by failure to explain is equivalent to negation by GCWA.

To generate abductive explanations of a given goal G wrt D, we have modified the *restart model elimination calculus* of [BF94a,BFS95]. The modified calculus is used to generate all necessary (possibly empty) potential candidates Δ s.t. $D \cup \Delta \models G$, and consistency checks are then carried out to verify if $D \cup \Delta$ is consistent or not. If no abductive explanation for G is found, then G is declared to be false under GCWA.

Reasoning wrt WGCWA is relatively easier and requires no consistency check. Given a program D and a goal G, the same modified calculus is used to check if there exists a Δ, referred to as a potential candidate, s.t. $D \cup \Delta \vdash G$. G is declared to be false under WGCWA if there exists no potential candidate (possibly empty) to explain it.

This approach to minimal model reasoning has been implemented in theorem prover PROTEIN[1] [BF94b]. The following example illustrates the above ideas and how to use the system.

```
% protein_flag(mmr,wgcwa).
protein_flag(mmr,gcwa).

number(0).
number(s(X)) :- number(X).
number(X); bird(X).

?- bird(s(s(s(0)))).
% ?- bird(tweety).
```

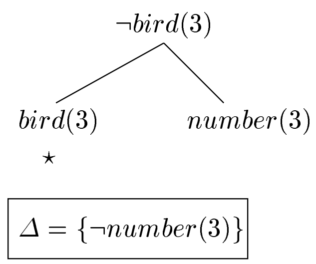

Fig. 2. Minimal model reasoning based on abduction

Example 2. The left half of the Figure 2 lists a PROTEIN input file, a disjunctive logic program and a query. PROTEIN uses Prolog like syntax with the symbol "`;`" in the head of a clause that denotes disjunction. There are two flags "`gcwa`" and "`wgcwa`" for performing minimal model reasoning wrt GCWA and WGCWA respectively. Any line starting with the character "`%`" is discarded as comment.

In this example, given the query `bird(s(s(s(0))))` and the flag `gcwa`, PROTEIN first constructs a potential candidate for explaining this. As shown in the right half of the Figure 2, PROTEIN finds that `s(s(s(0)))` could be a `bird` if it is not a `number`. However, after a consistency check this is discarded, and since no more potential candidates can be generated, PROTEIN declares the given goal to be false. If the same query is run with `wgcwa` flag, PROTEIN declares the goal to be indefinite since it has found a potential explanation. No consistency check is carried out in this case.

2.3 Minimal Model Generation and Reasoning

We have developed a method for handling a very general form of minimal model reasoning (parallel circumscription with both fixed and varying predicates) [Nie96b,Nie96a]. The method is based on a new technique for extending a tableau calculus to minimal model reasoning. As the underlying tableau method we have used a novel hyper tableau calculus [BFN96] which merges features from tableau techniques and hyper resolution. This calculus is extended to minimal model reasoning by employing a new technique which exploits the groundedness property of minimal models and enables a one branch at a time

[1] available in the web at `<URL:http://www.uni-koblenz.de/ag-ki/Protein/>`

approach to constructing tableaux for minimal model reasoning. The calculus is sound for general first-order circumscription and complete when no function symbols are allowed. In contrast to other methods for minimal model reasoning, the new technique leads to low space complexity and, e.g., in the ground case polynomial space complexity is obtained. We are not aware of any other implementation of minimal model reasoning with this property. The method can treat both query-answering and minimal model generation and it can handle examples with hundreds of thousands of minimal models. The method has been implemented in Eclipse Prolog and is available on the web at `<URL:http://www.uni-koblenz.de/ag-ki/DLP/>`.

Example 3. The `mm` module of DisLoP is illustrated by the following two cases `demo1` and `demo2`. `demo1` consists of the following statements and instructs the `mm` module to compute all the minimal models of the given clauses:

```
minimized(rest).
query_mode(all).

a1;b1;c1;d1;e1<-true.               a2 <- a3.
a2;b2;c2;d2;e2<-true.               a3 <- a4.
a3;b3;c3;d3;e3<-true.               a4 <- a5.
a4;b4;c4;d4;e4<-true.               a5 <- a1.
a5;b5;c5;d5;e5<-true.
```

`demo2` consists of all the above statements plus the following additional query statement which instructs the `mm` module to process the given query:

```
?- b1;c1;d1;e1;(a1,a2,a3,a4,a5).
```

Now, a session with `mm` generates all 1365 minimal models for `demo1` and answers affirmatively to the query of `demo2`:

```
ECRC Common Logic Programming System [sepia opium megalog parallel]
Version 3.5.2, Copyright ECRC GmbH, Wed Jan  3 12:54 1996
[eclipse 1]: [mm].
mm version 0.71
mm.pl      compiled traceable 42604 bytes in 0.08 seconds
yes.
[eclipse 2]: mm(demo1).
1. MINIMAL MODEL: [e1, e2, e3, e4, e5]
2. MINIMAL MODEL: [e1, e2, e3, e4, d5]
3. MINIMAL MODEL: [e1, e2, e3, e4, c5]
.....
1363. MINIMAL MODEL: [b1, a2, a3, a4, b5]
1364. MINIMAL MODEL: [b1, a2, a3, a4, a5]
1365. MINIMAL MODEL: [a1, a5, a4, a3, a2]
```

```
mm minimal model reasoner

Results:
1365 model(s) found.
6300 ms. of CPU time used.

yes.
[eclipse 3]: mm(demo2).
mm minimal model reasoner

Results:
Query succeeded.
0 countermodel(s) found.
0 ms. of CPU time used.

yes.
[eclipse 4]:
```

3 General Disjunctive Programs

For general disjunctive programs we have focused on STATIC ([BDP96]) and the D-WFS semantics ([BD95a,BD97b,BD95b]) and on generalizations of the abductive approach to incorporate negation. We have studied their mutual relationships, implementation methods and relationship to minimal model reasoning.

In Subsection 3.1 we present our transformation approach for ground programs. There is also a corresponding bottom-up computation which works for range-restricted Datalog programs and is closely related to our transformations. Subsection 3.2 describes an implementation of D-WFS using hyper tableaux. To our knowledge, it is the first implementation of disjunctive semantics which runs on polynomial space.

Recently ([DS97], an extended version to appear as [DS98]), Dix and Stolzenburg extended the transformation-calculus to non-ground programs with function-symbols. This is ongoing work and subject to further research.

3.1 D-WFS Based on Transformations

[BD96,BD97a] defines D-WFS as a confluent calculus of program-transformations as well as a bottom-up computation using certain fixpoints. Both methods have been implemented with various settings (which transformations should be given higher priority and so on) and are available as Prolog programs.

All the transformations are defined for ground programs The *Tautology*-transformation allows us to delete rules that contain the same atom in their head and in the body. The *Non-Minimal*-transformation deletes non-minimal rules: if both the head and the body of a rule r are contained in another rule r', then r' can be deleted.

Positive Reduction allows us to simply skip all occurrences of negative literals $\neg A$ if A does not occur in any head-rule. *Negative Reduction* allows us to delete a rule with an occurrence of $\neg A$ in its body, if there is a rule $A \leftarrow$ in the program.

The *Unfolding*-Transformation, finally, is the most costly operation. It allows us to select any positive occurrence of an atom A in the body of a clause, and to replace this clause by resolving it with all *definitions* of A (these are all rules having A in their heads). Obviously, this transformation should only be applied when all others are no more applicable.

The main result then is that our calculus of transformations is confluent and terminating. This implies that any program has a normal form, called the *residuum*. This residuum contains the whole information to answer a query immediately:

$A_1 \vee A_2 \ldots \vee A_n$ is true in P	iff	the residuum contains a rule $\mathcal{A} \leftarrow$, with $\mathcal{A} \subseteq \{A_1, A_2, \ldots, A_n\}$
$\neg A_1 \vee \neg A_2 \ldots \vee \neg A_n$ is true in P	iff	$\exists\ 1 \leq i_0 \leq n$ such that A_{i_0} does not occur in any head of the residuum

In addition, sub-calculi suitable for positive disjunctive programs (GCWA) and normal programs (WFS) (described in [BD96,BD97a]) have been implemented. In particular, the Brass, Freitag and Zukowski method for WFS, which works in quadratic time and computes a slightly different residuum, is available.

In [BD95b,BD97b] we also developed a bottom-up approach to compute the residuum. This approach works also for range-restricted function-free disjunctive programs. In [BD95a] we showed how to compute the stable semantics given the residuum. We are currently comparing these methods with the others described above.

Example 4. Consider the following disjunctive logic program:

```
a,b <- c, ~c, ~d.                c,d <- ~e.
a,c <- b.                        b <- ~c, ~d, ~e.
```

The `dwfs` module of DisLoP computes the residuum of this program as follows:

```
Pos. Red.: c, d <- ~e to c, d <-
Pos. Red.: b <- ~c, ~d, ~e to b <- ~c, ~d
Neg. Red.:  [a, b <- c, ~c, ~d] eliminated by [c, d <- ]
Neg. Red.:  [b <- ~c, ~d] eliminated by [c, d <- ]
GPPE: replacing  a, c <- b by nothing

Residuum:
c, d <-
```

3.2 Implementing D-WFS with Polynomial Space

Although D-WFS is defined using a calculus of program-transformations without any reference to models, a connection between D-WFS and minimal models has been established [BD96,BD97a]. It has been further shown that if the notion of a model of a disjunctive program is strengthened, then D-WFS can be defined iteratively using standard parallel circumscription [BDNP97]. This enables us to develop a novel implementation method for D-WFS which contrary to most other implementations of disjunctive semantics works in *polynomial space* in the ground case[2].

The idea is to implement D-WFS as an iterative reduction on disjunctive logic programs. The reduction employs parallel circumscription and classical reasoning. It starts with the original program and leads to a reduced program with the property that every query can be answered from this program with one call to a theorem prover for parallel circumscription. This can be seen as a compilation or a partial evaluation of the program leading to a smaller program from which all queries can be answered. The polynomial space complexity is obtained by combining the reduction approach with the new technique for minimal model reasoning described in Section 2.3. The method is also applicable to the flat STATIC for which we have shown that it coincides with D-WFS [BDNP97].

Example 5. Consider the same program as in Example 4. The `dwfs_mm` module of DisLoP first computes the following partial evaluation of the program:

```
c ; d <- true
a ; c <- b
b <- not c , not d
a ; b <- c , not c , not d
```

Now a query can be answered through a theorem prover call (with parallel circumscription) with this compiled program. For example, the query `?- c ; d.` succeeds with this program.

4 Applications

We live in a constantly changing and uncertain world and this is reflected in the knowledge/information/data that we have at hand. More often the knowledge is indefinite and in a constant state of flux. Disjunctive logic program is a more natural way of representing and dealing with such knowledge/information/data. One real-world example is the set of rules used by a bank to calculate banking fees. Because of the indefiniteness of the rules, it is not obvious if a banking fee can be calculated for any given situation and if it could be calculated will it be unique etc. We have successfully used constraint disjunctive logic programs to deal with such questions [ST96].

[2] For an implementation, see `<URL:http://www.uni-koblenz.de/ag-ki/DLP/>`

The disjunctive techniques that we have developed during the course of this project (such as minimal model reasoning, abduction, semantics of negation etc.) are useful in various applications such as diagnosis and database updates. For example, [BFFN97b,BFFN97a] explores how hyper tableau calculus [BFN96] for disjunctive programs along with minimal model reasoning technique of [Nie96b,Nie96a] can be exploited for diagnosis. In a diagnosis setup, we have a logical description of the system which clearly describes the behaviour of the system when all components are functioning normally. For a given input vector an observation of the output is made. If this observation contradicts the expected behaviour, the purpose of diagnosis process is to identify those components that could be faulty. To carry out this task, the key idea of [BFFN97b,BFFN97a] is to transform the given system description, input vector, and the observation into a disjunctive logic program in such a way that minimal model reasoning (minimal wrt the abnormality of the components) with the transformed program provides diagnosis for the original system. Note the use of disjunctive techniques even when the system description does not have any disjunctions!

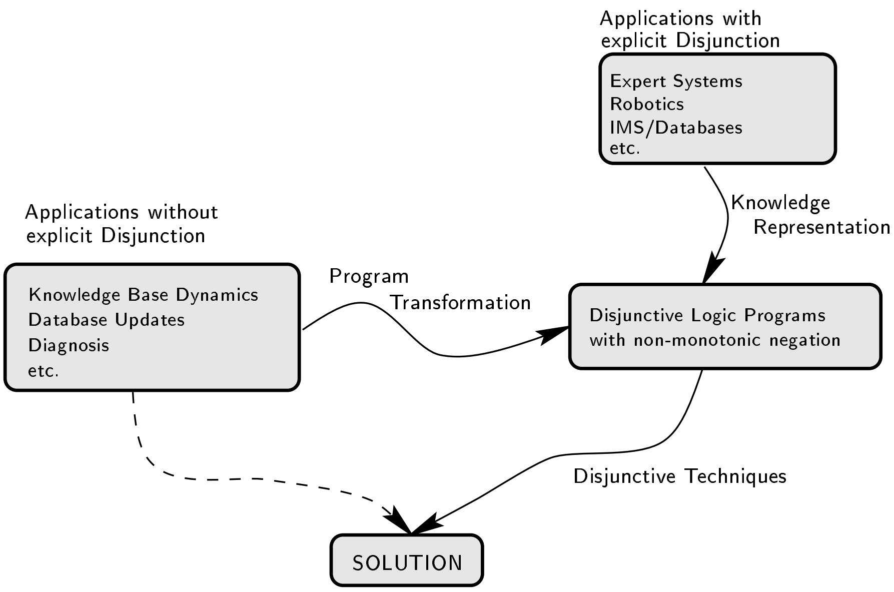

Fig. 3. Applications of Disjunctive Logic Programming

The diagnosis task is not very different from carrying out view updates in deductive and relational databases [AD95]. Especially, it is very closely related to the problem of deleting a view atom. So, disjunctive techniques can also be used for updating databases and we are currently working on a report providing the details in this regard [AB97]. Also interesting and important is to analyze if the solutions that we compute are rational at all [AD95]. This aspect is also being studied in the working report.

Another important application of disjunctive techniques in the database area is to glue together different heterogeneous databases to provide a single unified view to the user. With the ever expanding world wide web technology, millions and millions of data in thousands of different formats are thrown at a user who clearly needs some tools to put together information of interest. We have initiated a separate project to address such issues and beginning to use ideas and concepts developed for the DisLoP project. For example, the system that integrates different heterogeneous databases needs to know what kinds of external database calls are necessary to answer a particular query from a user. Abduction naturally solves such problems and the system looks for abductive explanations for the query in terms of external calls. There are so many other aspects where disjunctive techniques could be employed and we are currently studying them in detail.

Other areas for applications of disjunctive techniques include robotics, planning, hypothetical reasoning etc. We plan to explore such areas in the near future.

5 Acknowledgements

The authors would like to thank all the members of the Artificial Intelligence Research Group at the University of Koblenz, Germany, for fruitful discussions and their meticulous reading of the drafts of this paper. DisLoP system is a part of the research project on disjunctive logic programming funded by the DFG under the grant `Fu 263/3-1`.

References

[AB97] Chandrabose Aravindan and Peter Baumgartner. A rational and efficient algorithm for view deletion in databases. Technical Report RR–10–97, Fachbereich Informatik, Universität Koblenz-Landau, Koblenz, Germany, 1997.

[ABD+96] C. Aravindan, P. Baumgartner, J. Dix, U. Furbach, G. Neugebauer, I. Niemelä, D. Schäfer, and F. Stolzenburg. On merging theorem proving and logic programming paradigms. In M. Maher, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, page 546, Bonn, Germany, September 1996. The MIT Press.

[AD95] Chandrabose Aravindan and Phan Minh Dung. Knowledge base dynamics, abduction, and database updates. *Journal of Applied Non-Classical Logics*, 5(1):51–76, 1995. A related report is available on the web from <http://www.uni-koblenz.de/~arvind/papers/>.

[Ara96a] Chandrabose Aravindan. An abductive framework for negation in disjunctive logic programming. In J. J. Alferes, L. M. Pereira, and E. Orlowska, editors, *Proceedings of Joint European workshop on Logics in AI*, Lecture Notes in Artificial Intelligence 1126, pages 252–267. Springer-Verlag,

1996. A related report is available on the web from <http://www.uni-koblenz.de/~arvind/papers/>.

[Ara96b] Chandrabose Aravindan. DisLoP: A disjunctive logic programming system based on PROTEIN theorem prover. In G. Görz and S. Hölldobler, editors, *Proc. of KI'96*, Lecture Notes in Artificial Intelligence 1137, pages 19–23. Springer-Verlag, 1996.

[BD95a] Stefan Brass and Jürgen Dix. A General Approach to Bottom-Up Computation of Disjunctive Semantics. In J. Dix, L. Pereira, and T. Przymusinski, editors, *Nonmonotonic Extensions of Logic Programming*, LNAI 927, pages 127–155. Springer, Berlin, 1995.

[BD95b] Stefan Brass and Jürgen Dix. Disjunctive Semantics based upon Partial and Bottom-Up Evaluation. In Leon Sterling, editor, *Proceedings of the 12th Int. Conf. on Logic Programming, Tokyo*, pages 199–213. MIT Press, June 1995.

[BD96] Stefan Brass and Jürgen Dix. Characterizing D-WFS: Confluence and Iterated GCWA. In L.M. Pereira J.J. Alferes and E. Orlowska, editors, *Logics in Artificial Intelligence (JELIA '96)*, LNCS 1126, pages 268–283. Springer, 1996.

[BD97a] Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Well-founded Semantics: Confluent Calculi and Iterated GCWA. *Journal of Automated Reasoning*, to appear, 1997. (Extended abstract appeared in: Characterizing D-WFS: Confluence and Iterated GCWA. *Logics in Artificial Intelligence, JELIA '96*, pages 268–283, 1996. Springer, LNCS 1126.).

[BD97b] Stefan Brass and Jürgen Dix. Semantics of Disjunctive Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, accepted for publication, 1997.

[BDNP97] Stefan Brass, Jürgen Dix, Ilkka Niemelä, and Teodor. C. Przymusinski. Comparison and Efficient Computation of the Static and the Disjunctive WFS. In Gerd Brewka, Emil Weydert, and Cees Witteveen, editors, *Proceedings of the third Dutch-German Workshop on Nonmonotonic Reasoning and its Applications*, pages 37–42. February 1997.

[BDP96] Stefan Brass, Jürgen Dix, and Teodor. C. Przymusinski. Super Logic Programs. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, pages 529–541. San Francisco, CA, Morgan Kaufmann, 1996.

[BF94a] 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.

[BF94b] P. Baumgartner and U. Furbach. PROTEIN: A *PRO*ver with a *T*heory *E*xtension *I*nterface. In A. Bundy, editor, *Automated Deduction – CADE-12*, volume 814 of *LNAI*, pages 769–773. Springer, 1994. Available in the WWW, URL: `http://www.uni-koblenz.de/ag-ki/Systems/PROTEIN/`.

[BFFN97a] Peter Baumgartner, Peter Fröhlich, Ulrich Furbach, and Wolfgang Nejdl. Semantically Guided Theorem Proving for Diagnosis Applications. In *Proc. of IJCAI '97*. IJCAI, 1997. To appear.

[BFFN97b] Peter Baumgartner, Peter Fröhlich, Ulrich Furbach, and Wolfgang Nejdl. Tableaux for Diagnosis Applications. In *Tableaux '97*, LNAI. Springer, 1997. To appear.

[BFN96] Peter Baumgartner, Ulrich Furbach, and Ilkka Niemelä. Hyper tableaux. In José Júlio Alferes, Luís Moniz Pereira, and Ewa Orlowska, editors, *Proceedings of the European Workshop on Logics in Artificial Intelligence (JELIA'96)*, pages 1–17, Èvora, Portugal, September/October 1996. Springer-Verlag.

[BFS95] 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*).

[DS97] Jürgen Dix and Frieder Stolzenburg. Computation of Non-Ground Disjunctive Well-Founded Semantics with Constraint Logic Programming (preliminary report). In J. Dix, L. Pereira, and T. Przymusinski, editors, *Nonmonotonic Extensions of Logic Programming*, LNAI 1216, pages 202–226. Springer, Berlin, 1997.

[DS98] Jürgen Dix and Frieder Stolzenburg. A Framework to incorporate Nonmonotonic Reasoning into Constraint Logic Programming. *Journal of Logic Programming*, 1998. Special Issue on *Constraint Logic Programming*, Guest Editors: Kim Marriott and Peter Stuckey, to appear.

[Llo87] J. W. Lloyd. *Foundations of Logic Programming.* Springer–Verlag, second extended edition, 1987.

[LMR92] Jorge Lobo, Jack Minker, and Arcot Rajasekar. *Foundations of disjunctive logic programming.* MIT Press, 1992.

[Min82] Jack Minker. On indefinite databases and the closed world assumption. In *Lecture Notes in Computer Science 138*, pages 292–308. Springer-Verlag, 1982.

[Nie96a] I. Niemelä. Implementing circumscription using a tableau method. In *Proceedings of the European Conference on Artificial Intelligence*, pages 80–84, Budapest, Hungary, August 1996. John Wiley.

[Nie96b] I. Niemelä. A tableau calculus for minimal model reasoning. In *Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 278–294, Terrasini, Italy, May 1996. Springer-Verlag.

[RLM89] Arcot Rajasekar, Jorge Lobo, and Jack Minker. Weak generalized closed world assumption. *Journal of Automated Reasoning*, 5:293–307, 1989.

[RT88] K. A. Ross and R. W. Topor. Inferring negative information from disjunctive databases. *Journal of Automated Reasoning*, 4(2):397–424, 1988.

[ST96] Frieder Stolzenburg and Bernd Thomas. Analysing rule sets for the calculation of banking fees by a theorem prover with constraints. In *Proceedings of the 2nd International Conference on Practical Application of Constraint Technology*, pages 269–282, London, 1996. Practical Application Company.