# Knowledge Representation and Reasoning in NMR and Extended LP


Jürgen Dix

University of Koblenz-Landau
Dept. of Computer Science

and

Technical University of Vienna
Institute for Knowledge-Based Systems

# VIII. Applications, Benchmarks and Systems

- Graph Problems
- Planning
- Diagnosis
- Implemented Systems

# 1 Graph Problems

## Idea

- Generate a default theory or a logic program from a given graph and a graph problem in such a way that the extensions of the default theory or the stable models of the program corresponds to the solutions of the original graph problem for the given graph.
- E.g.: *3-colorability* and *Hamiltonian circuit problems*

Given a graph we construct a logic program as follows:

1. a set of atomic facts of the form $vertex(v)$ and $arc(v,u)$ corresponding to the vertices and arcs of the graph is taken and
2. the following rules are included

$$color(V,blue) \leftarrow vertex(V), not\ color(V,green), not\ color(V,red)$$
$$color(V,green) \leftarrow vertex(V), not\ color(V,blue), not\ color(V,red)$$
$$color(V,red) \leftarrow vertex(V), not\ color(V,green), not\ color(V,blue)$$
$$noncoloring \leftarrow arc(V,U), color(V,C), color(U,C)$$
$$p \leftarrow not\ p, noncoloring$$

1. The above program has a stable model if and only if there is a 3-coloring of the graph.

2. The coloring can be read from the stable model directly by taking the facts of the form $color(v,c)$ true in the model.

3. For systems with query-evaluation: benchmarks are obtained by removing the last rule. Then there is a 3-coloring of the graph if and only if the resulting program has a stable model not containing $noncoloring$.

## 1.1 Hamiltonian Circuit Problem

The corresponding logic program is constructed as follows:

- a set of atomic facts of the form $vertex(v)$ and $arc(v,u)$ corresponding to the vertices and arcs is taken,

- the following rules are included

$$hcircuit(V_1,V_2) \leftarrow arc(V_1,V_2), not\ otherroute(V_1,V_2)$$

$$otherroute(V_1,V_2) \leftarrow arc(V_1,V_2), hcircuit(V_1,V_3), not\ (V_2 = V_3)$$

$$otherroute(V_1,V_2) \leftarrow arc(V_1,V_2), hcircuit(V_3,V_2), not\ (V_1 = V_3)$$

$$reached(V_2) \leftarrow hcircuit(V_1,V_2), reached(V_1)$$

$$reached(V_2) \leftarrow hcircuit(V_1,V_2), initialnode(V_1)$$

$$noncircuit \leftarrow vertex(V), not\ reached(V)$$

$$p \leftarrow not\ p, noncircuit$$

- one of the vertices $v$ is taken as the starting vertex ($initialnode(v)$ is added).

# Propositional Satisfiability

Capture propositional satisfiability using the *nonmonotonic primitives* of the formalism rather than directly in terms of classical logical operators that are included in the formalism.

A set is satisfiable *if and only if* a stable model of its translation exists.

For a set of clauses $S$, we build a logic program as follows:

1. we introduce for each atom $a$ appearing in $S$ two atoms $a$ and $\hat{a}$ and include two rules

$$\hat{a} \leftarrow not\ a$$
$$a \leftarrow not\ \hat{a}$$

2. for each clause in $S$, we introduce a new atom $c$ and include one rule for each literal $l$ in the clause as follows: if $l$ is a positive atom $a$, take the rule $c \leftarrow a$ and if $l$ is the negation of an atom $a$, add $c \leftarrow \hat{a}$.

3. Finally we include the rules

$$satisfied \leftarrow c_1, \ldots, c_n$$
$$p \leftarrow not\ p, not\ satisfied$$

where $c_1, \ldots, c_n$ are all the atoms introduced for the clauses in $S$.

The propositional model is obtained from the stable model by assigning true for atoms true in the stable model and assigning false to the rest of the atoms (for atoms $a$ for which $\hat{a}$ is true in the stable model). For query-evaluation a benchmark is obtained by removing the last rule. Then the set of clauses $S$ has a model if and only if the resulting program has a stable model containing the atom $satisfied$.

# 2 Planning

- *Reasoning about action and change* and the *frame problem* have to be addressed and the expressivity of the nonmonotonic formalisms can be utilized to overcome some of the difficulties.

- *Blocks world:* Given initial conditions of blocks on a table, goal conditions and a robot that is able to move blocks.
  *Aim:* Generate a plan for the robot to move blocks from the initial configuration so that the goal conditions are satisfied.

1. Three blocks $a, b, c$ such that $b$ and $c$ are on the table and $a$ is on top of $b$.
2. The goal conditions are that $c$ is on $a$ and $b$ is on $c$.
3. A solution for this planning problem is a sequence of moves where $a$ is moved on the table, $c$ is moved on $a$ and finally $b$ is moved on $c$.
4. *Idea:* Map a planning problem to a logic program such that stable models correspond to valid plans.

## Initial Situation:

$$on(a,b,t_0)$$

$$on(b,table,t_0)$$

$$on(c,table,t_0)$$

## Goal:

$$goal(T) \leftarrow on(b,c,T), on(c,a,T), on(a,table,T)$$

$$goal(T_2) \leftarrow nextstate(T_2,T_1), goal(T_1)$$

$$goal \leftarrow goal(T)$$

$$p \leftarrow not\ p, not\ goal$$

## Preconditions and effects:

$$
\begin{aligned}
moveop(X,Y,T) \leftarrow\ & moveable(X), block(Y), not\ (X = Y), on\_something(X,T),\\
& not\ covered(X,T), not\ covered(Y,T),\\
& not\ blocked\_move(X,Y,T)
\end{aligned}
$$

$$on(X,Y,T_2) \leftarrow nextstate(T_2,T_1), moveop(X,Y,T_1)$$

$$on\_something(X,T) \leftarrow on(X,Z,T)$$

$$covered(X,T) \leftarrow moveable(X), on(Z,X,T)$$

$$moveable(X) \leftarrow block(X), not\ (X = table)$$

## Frame Axiom:

$$on(X,Y,T_2) \leftarrow nextstate(T_2,T_1), on(X,Y,T_1), not\ moving(X,T_1)$$

$$moving(X,T) \leftarrow moveop(X,Y,T)$$

## Blocking Conditions I:

$$blocked\_move(X,Y,T) \leftarrow moveable(X), block(Y), goal(T)$$

$$blocked\_move(X,Y,T) \leftarrow on\_something(X,T), block(Y), not\ moveop(X,Y,T)$$

## Blocking Conditions II:

$$blocked\_move(X,Y,T) \leftarrow moveable(X), block(Y), moveop(X,Z,T), not\ (Y = Z)$$

$$blocked\_move(X,Y,T) \leftarrow moveable(X), moving(Y,T)$$

$$\begin{aligned} blocked\_move(X,Y,T) \leftarrow\ & moveable(X), block(Y), not\ (Y = table), \\ & moveop(Z,Y,T), not\ (X = Z) \end{aligned}$$

1. The above program has a stable model if and only if there is a sequence of moves from the initial configuration to a situation satisfying the goal conditions that can be executed concurrently in at most $n$ steps.

2. *Query-evaluation:* Remove the rule with $p$ in the head. Then there is plan if and only if the resulting program has a stable model containing $goal$.

# 3 Diagnosis

1. Examples stemming from hardware design usually contain a large number of clauses, which have to be handled efficiently in order to solve realistic problems.

2. Diagnosis task involves minimal model reasoning by definition, and hence techniques from nonmonotonic reasoning have to be applied.

## 3.1 Simulation Model

1. Simulation model of the technical device under consideration is constructed and is used to predict its normal behavior. The diagnosis is derived by comparing the prediction with the actual behaviour.

2. The system description (*SD*) is formalized by a set of first–order formulas.

3. The diagnostic problem is described by system description *SD*, a set *COMP* of components and a set *OBS* of observations (logical facts).

4. With each component we associate a behavioral mode: $Mode(c, Ok)$ means that component $c$ is behaving correctly, while $Mode(c, Ab)$ (abbreviated by $Ab(c)$) denotes that $c$ is faulty.

A *Diagnosis* of $(SD, COMP, OBS)$ is a set $\Delta \subseteq COMP$, such that

$$SD \cup OBS \cup \{Mode(c, Ab) | c \in \Delta\} \cup \{\neg Mode(c, Ab) | c \in COMP - \Delta\}$$

is consistent. $\Delta$ is called a *Minimal Diagnosis*, *if and only if* it is the minimal set (wrt. $\subseteq$) with this property.

# 4 Systems

A web-page on LP/NMR systems has been created and is maintained continuously:

http://www.uni-koblenz.de/ag-ki/LP/

**Non-Disjunctive:**

- Model-generation TP (Inoue),
- Explicit Negation (Pereira et. al.),
- Fast Computation of Stable Models, handles 100000 models (Niemelä, Simons).

**Non-Disjunctive:**

- Implementation of WFS: XSB-System, based on SLG-Resolution (Warren et. al.).
  - Handles first-order programs and in addition it Constitutes a complete PROLOG System.
  - XSB implements Success, Failure, Reduction, Loop-Detection using Tabling in low level data-structures (WAM-level).
  - Termination is not guarantueed in general (undecidable) but in restricted cases (non-floundering Datalog). A finite propositional residual program is computed: it is the same as introduced in Chapter III.

**Disjunctive:**

- Model-generation TP (Inoue),
- DisLog-System (Seipel): Minker's semantics,
- `dlv`-System at TU Vienna (Leone/Gottlob): started last summer. Based on complexity analysis.

**Disjunctive:**

- DisLoP (Dix/Furbach): Programming System for D-WFS, DSTABLE, STATIC.
  - Taking advantage of a Tableau-Theorem Prover for classical predicate logic: Polynomial Space implementation.
  - Query-answering for pos.-disj. programs: definite answers are preferred.
  - Abductive Methods are also incorporated.
  - Implementation of first-order D-WFS on its way: Dix/Stolzenburg, Special Issue of JLP (1998): Combining CLP and NMR. Also work of Sakama/Seki, JLP(1997) 32(3)*229-245.

## What is the problem of extending D-WFS?

**Example 4.1 (Duplicate Occurrences and GPPE)**
*In the program $\Phi$ we apply GPPE and obtain $GPPE_B(\Phi)$.*

$$\Phi: \quad \begin{array}{rcl} A & \leftarrow & B \\ B \vee B & \leftarrow & \end{array} \qquad GPPE_B(\Phi): \quad \begin{array}{rcl} A \vee B & \leftarrow & \\ B \vee B & \leftarrow & \end{array}$$

*$GPPE_B(\Phi)$ is irreducible, but equivalent to the program $A \vee B \leftarrow$, $B \leftarrow$, which reduces to $B \leftarrow$ only. Therefore we can derive $B$ but not $A$—we can even derive $\neg A$—, if we apply our normalform-characterisation given in Theorem 2.4.*

*But the original program is certainly equivalent to $A \leftarrow B$, $B \leftarrow$ , which reduces to $A \leftarrow$ , $B \leftarrow$ , and therefore not only $B$ but also $A$ is derivable.*

Solve this problem by using constraints!

# References


Berman, K. A., J. S. Schlipf, and J. V. Franco (1995, June). Computing the Well-Founded Semantics Faster. In A. Nerode, W. Marek, and M. Truszczyński (Eds.), *Logic Programming and Non-Monotonic Reasoning, Proceedings of the Third International Conference*, LNCS 928, Berlin, pp. 113–126. Springer.

Brass, S. and J. Dix (1995, September). D-WFS: A Confluent calculus and an Equivalent Characterization. Technical Report TR 12/95, University of Koblenz, Department of Computer Science, Rheinau 1.

Brass, S. and J. Dix (1996). Characterizing D-WFS: Confluence and Iterated GCWA. In L. P. J.J. Alferes and E. Orlowska (Eds.), *Logics in Artificial Intelligence (JELIA '96)*, LNCS 1126, pp. 268–283. Springer. (Extended version will appear in the *Journal of Automated Reasoning* in 1998.).

Dix, J. and P. Schmitt (1993). Nichtmonotones Schließen: Wieviel Nichtmonotonie ist nötig? *Kognitionswissenschaft 3*, 53–69.

Dix, J. and F. Stolzenburg (1998). A Framework to incorporate Nonmonotonic Reasoning into Constraint Logic Programming. *Journal of Logic Program-*

*ming 36(1,2,3)*, 5—37. Special Issue on *Constraint Logic Programming*, Guest Editors: Kim Marriott and Peter Stuckey.

Eiter, T. and G. Gottlob (1993). Propositional Circumscription and Extended Closed World Reasoning are $\Pi_2^P$-complete. *Theoretical Computer Science 144*(2), 231–245, Addendum: vol. 118, p. 315, 1993.

Eiter, T. and G. Gottlob (1995). On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence 15*(3/4), 289–323.

Gottlob, G. (1992). Complexity results for nonmonotonic logics. *Journal of Logic and Computation 2*(3), 397–425.

Lehmann, D. and M. Magidor (1992). What does a conditional knowledge base entail? *Artificial Intelligence 55*, 1–60.

Lifschitz, V. (1989, January). Benchmark Problems for Formal NMR, Version 2.00. In Reinfrank, de Kleer, Ginsberg, and Sandewall (Eds.), *Non-Monotonic Reasoning*, LNAI 346, Berlin, pp. 202–219. Springer.

Lloyd, J. W. (1987). *Foundations of Logic Programming*. Berlin: Springer. 2nd edition.

McCarthy, J. (1986). Applications of Circumscription to formalizing Common-Sense Reasoning. *Artificial Intelligence 28*, 89–116.

Niemelä, I. and P. Simons (1996, September). Efficient Implementation of the Well-founded and Stable Model Semantics. In M. Maher (Ed.), *Proceedings of the Joint International Conference and Symposium on Logic Programming*, Bonn, Germany, pp. 289–303. The MIT Press.

Reiter, R. (1980). A Logic for Default-Reasoning. *Artificial Intelligence 13*, 81–132.

Schlipf, J. (1986, June). How uncomputable is general circumscription? In *Proceedings of the Symposium on Logic in Computer Science*, Cambridge, USA, pp. 92–95. IEEE Computer Society Press.

Stillman, J. (1992, July). The complexity of propositional default logics. In *Proceedings of the 10th National Conference on Artificial Intelligence*, San Jose, California, USA, pp. 794–800. The MIT Press.

Tarjan, R. (1972). Depth first search and linear graph algorithms. *SIAM Journal on Computing 1*, 146–160.

Van Gelder, A. (1989, March). The Alternating Fixpoint of Logic Programs with Negation. In *Proc. of the Eight ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Philadelphia, Pennsylvania*, pp. 1–10. ACM Press.

Van Gelder, A., K. A. Ross, and J. S. Schlipf (1988). Unfounded Sets and well-founded Semantics for general logic Programs. In *Proceedings 7th Symposion on Principles of Database Systems*, pp. 221–230.