WFS or STABLE?

LP or DLP?

Where should it go?

J. Dix

University of Koblenz and TU Vienna

Part I: LP and DEDUCTIVE DB

Part II: SYSTEMS

Part III: APPLICATIONS

Part I: LP and Deductive DB

- LP-Paradigm and DDB
- Abstract Properties
- WFS and STABLE

1 LP-Paradigm and Ded. DB

1.1 PROLOG

1973–today: A. Colmerauer et.al., Paradigm of R. Kowalski:

Algorithm = Logic + Control

Programming Language: Definite Horn-clauses

$A \leftarrow B_1, B_2, \ldots, B_n$

Semantics:	M_P
Calculus:	SLD-Resolution

Turing complete: Function symbols are important!

1.2 LP/DDB with Negation

- Negation as **Finite Failure**: Classical LP-Semantics, based on Clark's completion comp (1978), or generalizations of SLD-Resolution.
- Negation as **Failure**: NMR-Semantics, that detect loops. Workshop on Deductive Databases (1985, Jack Minker).

1.3 DDB with Negation

Codd's relational model (1970) found its way into industry (SQL).

What happened to Ded. DB?

- Bottom-up Procedures. Magic Sets techniques, memoizing.
- DDB influenced real DBMS, but commercial mainstream products (SQL, DB2, SQL3) almost have no recursion.

- Many people consider only stratified semantics important.
- There are non-trivial applications (Bill-of-material, Flight Databases etc) and lots of prototypes.
- Industry refuses non-standard features.
- Broad applications using added capabilities in meaningful ways are needed.

1.4 Some Technical Notions

Definition I.1.1 (Logic Program P, $head(P)$)
A logic program P is a finite set of first-order rules

$$\boxed{A_1 \vee \dots \vee A_k \leftarrow B_1 \wedge \dots \wedge B_m \wedge \neg C_1 \wedge \dots \wedge \neg C_n},$$

where the $A_i/B_i/C_i$ are $\mathcal{L}$-atoms different from $\bot$, $k \geq 1$, $m \geq 0$, $n \geq 0$.

We identify such a rule with a triple and write it as $\boxed{\mathcal{A} \leftarrow \mathcal{B}^+ \wedge \neg \mathcal{B}^-}$.

We write $head(P)$ for the set of all atoms occuring in rule heads in P: these are atoms that are possibly true.

Definition I.1.2 (Sceptical Semantics $\mathcal{S}_{\mid\!\sim}$)
By a *semantic operator* $\mid\!\sim$ *we mean a binary relation between logic programs and disjunctions satisfying*

1. *Right Weakening:* *If* $P \mid\!\sim \psi$ *and* $\psi \subseteq \psi'$, *then* $P \mid\!\sim \psi'$.
2. *Success:* *If* $\mathcal{A} \leftarrow true \in P$ *for* $\mathcal{A}$, *then* $P \mid\!\sim \mathcal{A}$.
3. *Failure:* *If* $A \notin head(P)$ *for a* A, *then* $P \mid\!\sim \neg A$.

Given such $\mid\!\sim$ *and a* P, *by the* *semantics* *of* P *we mean* $\{\psi \mid P \mid\!\sim \psi\}$. *We also assume the dual of Success and failure, called* *Reduction*.

Last definition is only suitable for the sceptical viewpoint.

Semantics as sets of models need a credulous framework.

Definition I.1.3 (Credoulous Semantics SEM)
A semantics SEM is a mapping from the class of all programs into the powerset of the set of all 3-valued structures. SEM assigns to every program P a set of 3-valued models of P:

$$\boxed{\mathrm{SEM}(P) \subseteq \mathrm{MOD}^{\mathcal{L}_P}_{3-\mathrm{val}}(P).}$$

In this framework, Right Weakening and Success are automatically satisfied. We also assume Failure.

1.5 LP vs DDB

- Both in LP and DDB: recursion and Rule form !
- In LP we have Top-down procedure and Relevance: Answering a query should only depend on the call-graph below that query.
- In DDB we have bottom-up procedures.
- DDB is propositional, whereas LP contains function symbols.

2 Abstract Properties

Original intention:

What are useful properties and
which of them can we have together?

- Definitional extensions should be conservative. ($\rightsquigarrow$ GPPE)
- Tautologies should be removable. ($\rightsquigarrow$ Elim. of Taut.)
- Literals known to be true should be replaced by their truthvalue. ($\rightsquigarrow$ Reduction)

Definition I.2.1 (GPPE)
A semantics SEM satisfies GPPE if the following transformation does not change SEM:
Replace a rule $\mathcal{A} \leftarrow \mathcal{B}^+ \wedge \neg\mathcal{B}^-$ where $\mathcal{B}^+$ contains an atom B by the following rules

$$\mathcal{A} \cup (\mathcal{A}_i - \{B\}) \leftarrow (\mathcal{B}^+ - \{B\}) \cup \mathcal{B}_i^+ \wedge \neg(\mathcal{B}^- \cup \mathcal{B}_i^-)(i = 1, \ldots, n)$$

where $\mathcal{A}_i \leftarrow \mathcal{B}_i^+ \wedge \mathcal{B}_i^-$ $(i = 1, \ldots, n)$ *are all rules with* $B \in \mathcal{A}_i$.

$$\begin{array}{lcl} B & \leftarrow & \neg E \\ A' \vee B & \leftarrow & D, \neg C' \\ A \vee D & \leftarrow & B, \neg C \end{array} \quad \mapsto \quad \begin{array}{lcl} A \vee D \vee A' & \leftarrow & D, \neg C', \neg C \\ A \vee D & \leftarrow & \neg E, \neg C \end{array}$$

Definition I.2.2 (Elimination of Rules)

A semantics SEM satisfies a) Elimination of Tautologies, resp. b) Elimination of Contradictions if the following transformations are SEM-equivalent.

a) *Remove* $\mathcal{A} \leftarrow \mathcal{B}^+ \wedge \neg\mathcal{B}^-$ *if* $\mathcal{A} \cap \mathcal{B}^+ \neq \emptyset$.

b) *Remove* $\mathcal{A} \leftarrow \mathcal{B}^+ \wedge \neg\mathcal{B}^-$ *if* $\mathcal{B}^- \cap \mathcal{B}^+ \neq \emptyset$.

Tautology:	$A \vee B \vee C \leftarrow D, A, \neg E$
Contradiction:	$A \vee B \leftarrow C, D, \neg C$

Definition I.2.3 (Reduction, Nonminimal Rules)

A semantics $\mid\!\sim$ *satisfies* **a)** *Positive, resp.* **b)** *Negative Reduction, resp.* **c)** *Elimination of nonminimal Rules, if the following transformations are* $\mid\!\sim$*-equivalent:*

a)

$$\begin{array}{lcl} A \vee B & \leftarrow & \neg F \\ E & \leftarrow & A, \neg F \end{array} \quad \mapsto \quad \begin{array}{lcl} A \vee B & \leftarrow & \\ E & \leftarrow & A \end{array}$$

b)

$$\begin{array}{lcl} A \vee B & \leftarrow & \\ F \vee E & \leftarrow & C, \neg A, \neg B, \neg F \end{array} \quad \mapsto \quad \begin{array}{lcl} A \vee B & \leftarrow & \end{array}$$

c)

$$\begin{array}{lcl} A \vee B \vee C & \leftarrow & D, \neg E, \neg F \\ A \vee B & \leftarrow & D, \neg F \end{array} \quad \mapsto \quad \begin{array}{lcl} A \vee B & \leftarrow & D, \neg F \end{array}$$

3 WFS and STABLE

Theorem I.3.1 (D-WFS)
There exists a weakest semantics D-WFS of a program P satisfying GPPE, Elimination of Tautologies and nonminimal Rules, positive and negative Reduction.

Theorem I.3.2 (Confluence)
The calculus of our transformation rules is confluent. Every program P can be transformed to a unique irreducible program which coincides with the residual program $\hat{\mathbf{P}}$.

No strict subset of our transformation rules has these properties.

- GPPE is of high complexity even for non-disjunctive programs. The normalform $\hat{\mathbf{P}}$ of a program P can be exponential in the size of P!
- We can replace it by the following linear transformations
  - Success,
  - Failure and a
  - loop-detection rule.

This gives us a new normalform $\overline{\mathbf{P}}$.

Theorem I.3.3 ($comp_3$, WFS, STABLE)

- *$comp_3$ is the weakest semantics satisfying Success, Failure and Reduction. This calculus is strongly terminating and confluent.*
- *WFS is the weakest semantics satisfying in addition Loop-Detection and Elimination of Tautologies. This calculus is strongly terminating and confluent.*
- *STABLE is the weakest semantics satisfying in addition Elimination of Contradiction.*

(Brass/Dix, **JLP**, 32(3), 1997, **JLP**, to appear, 1999, **JAR**, 20(1), 1998.)

- The calculus of our transformations for WFS is provably better that van Gelder's alternating fixpoint procedure.
- Our algorithm is linear for the class of all programs, where in each strongly connected component the following holds: either all the head-atoms defined in that component depend *only positively from each other* or *no predicate defined in that component depends positively on itself* (also called *positive-acyclic*).

(See Brass/Dix/Freitag/Zukowski, submitted to **JLP**, TR 15/98 Uni Koblenz.)

Theorem I.3.4 (Properties of some Semantics)

Semantics	*Taut.*	*Contr.*	*GPPE*	*Cons.*	*Indep.*	*Relev.*	*Isom.*
$comp$	—	•	•	—	—	—	•
$comp_3$	—	—	•	•	•	•	•
M_P^{supp}	•	•	•	•	•	•	•
WFS	•	—	•	•	•	•	•
GCWA	•	•	•	•	•	•	•
DSTABLE	•	•	•	—	—	—	•
STATIC	•	—	•	•	•	•	•
D-WFS	•	—	•	•	•	•	•

4 How to lift the Transformation Approach to Nonground Programs

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

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

$GPPE_B(\Phi)$ 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 $\neg A$—.

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!

1. We extended our calculus of transformations to constraint LP's with disjunctions.
2. The calculus is weakly terminating for arbitrary programs.
3. The calculus is terminating for DATALOG$^{\vee,\neg}$ programs.
4. Any constraint theory can be exploited in the context of NMR.

(Dix/Stolzenburg, **JLP, Special Issue on Constraints**, 35(1,2,3), 1998.)

Part II: Systems

- XSB
- s-models
- (DisLoG, DisLoP, dlv . . .)

- Shakertown meetings (1994, 1998) on the future of LP.
  (Dix, **AICOMM**, 8(1) 1995, **AICOMM**, 11(3) 1998.)

- Recent Initiative of John Lloyd for a new big conference Computational Logic replacing JICLP and hosting smaller workshops (LOPSTR, LPKR, LPNMR, ML).

5 XSB (Warren et. al.)

- Complete first-order PROLOG system. If tabled predicates are used, they are handled according to WFS.
- Based on complicated SLG-Resolution. Implemented by tabling in low level data-structures (WAM-level).

- XSB is terminating on many programs where usual PROLOG systems fall into an infinite loop.
- Termination guarantueed only for non-floundering DATALOG. A finite prop. program is computed: exactly our $\overline{\mathbf{P}}$.
- Thus our transformations give a very clear description of SLG-resolution on the program-level. This is possible due to separation of goal-directedness and overall evaluation method.
- Delaying positive subgoals is done in Loop-Detection. Delaying negative subgoals is governed by all other transformations.

6 s-models (Niemelä/Simons)

- Works on ground propositional (or allowed Datalog) programs.
- Works for hundreds of thousands of rules of non-stratified programs.
- Scalability is an issue!

Planar graphs, 3-coloring			
Vertices	Rules	SLG	smodels
12	87	82.53	0.03
14	101	390.56	0.04
16	115	1421.19	0.05
100	703		0.21
300	2103		0.71
500	3503		1.12

How does smodels work?

- Compute P-full sets: $\mathrm{F} \subseteq \{not\ A \mid A \in \mathrm{NA}(P)\}$.
  - for P-full F, $\mathrm{Cn}_{\mathrm{Mon}(P,\mathrm{F})}(\emptyset)\ \in$ STABLE(P),
  - for each stable model M,

$$\boxed{\mathrm{F} = \{not\ A \mid A \in \mathrm{NA}(P) - M\}}$$

is the unique corresponding full set.

To compute P-full sets, it suffices to consider those atoms A or $not\ A$ which are not covered.

Very efficient heuristics for choosing the next choice-point:

- Compute first the strongly connected components.
- Use **Mom's heuristics** (Davis-Putnam like). Given FS, select a negative literal from a body of a potentially applicable rule with the least number of negative literals not in FS.

(See Dix/Niemelä/Furbach, **Handbook of AR**, Voronkov/Robinson (eds.), 1999.)

Part III: Applications

- Model Checking
- Planning
- New Constraint Paradigm?
- Agents

Can we find applications, where our methods

- not only can successfully applied,
- but compete with dedicated special purpose systems?

7 Model Checking

Problem: Given a system specification, determine whether a certain property (temporal logic formula) holds.
(Verification, design errors of real life systems.)

Method: There exist lots of systems. They typically specify fixed points of certain types of functionals via structural recursion.

Idea: The minimal Herbrand model in LP is a least fixpoint. Greatest fixpoints can be computed by using negation.

Result:

1. A system XMC (consisting of 200 lines of code in XSB) for model checking finite-state systems (Milner's CCS in a fragment of the modal μ-calculus).
2. XMC is competitive with state-of-the-art model checkers coded in C/C++ (SPIN, Concurrency Factory).

(Warren et. al., *LP and Model Checking*, TR 1998, Swift, *Using Tabling for Non-monotonic Programming*, **AMAI**, to appear.)

Why XSB?:

1. Standard PROLOG approximates the least fixpoint via successful computations. But infinitely many computations do not terminate.
2. SLD is too weak. But WFS is right!
   (Reachability in a transition system as transitive closure.)
3. Take advantage of a full-fledged programming system. Table only those predicates that have to!

8 Planning

Idea: Transform a planning problem in a Datalog$^{\neg}$ program P, such that stable models of P correspond to valid plans. (Subrahmanian/Zaniolo, **ICLP '95**.)

Partial order planning: Non-deterministic choice (can be captured with stable models: Sacca/Zaniolo, **Proc. 9th ACM**, 1990.)

- Dimopoulos/Nebel/Koehler consider particular encodings of realistic planning problems. They use smodels to compute stable models and thereby computing valid plans. Results are often much better than dedicated planning systems (Graphplan, Satplan).

  (Dimopoulos/Nebel/Koehler, **ECP '97**, 1998.)

- How is the encoding done?

1. $\text{fluent}_i(t) \leftarrow \text{oper}_j(t-1)$.
2.

$$\begin{aligned} \text{switchon}_i(t) &\leftarrow \neg\text{blocked}_i(t) \\ \text{blocked}_i(t) &\leftarrow \neg\text{switchon}_i(t) \end{aligned}$$

3. Rules to represent inertia.
4.

$$
\begin{array}{lcl}
\mathrm{oper}_i(t) & \leftarrow & \mathrm{precon}_{i1}(t), \ldots, \mathrm{precon}_{im}(t), \mathrm{switchon}_i(t) \\
\mathrm{inco} & \leftarrow & \mathrm{oper}_i(t), \mathrm{controper}_{i1}(t) \\
 & \vdots & \\
\mathrm{inco} & \leftarrow & \mathrm{oper}_i(t), \mathrm{controper}_{im}(t)
\end{array}
$$

5. Optimizations of encodings are important: minimizing NAF-literals, minimizing number and arities of predicates, . . .

6. Number of time steps must be at least the length of the optimal plan.

7. In certain situations (post-serializability), encodings can be done to optimize finding an initial solution. The final plan is then constructed in a post-processing phase. Problems that are unsolvable by Graphplan are solved by smodels (400-5000 seconds).

9 New Constraint Paradigm?

Quick look on Complexity

1. order with functions:

- Horn clauses under M_P: $\boldsymbol{\Sigma^0_1, \Pi^0_1}$
- Normal programs under STABLE: $\boldsymbol{\Pi^1_1}$.

propositional:

- Horn clauses under M_P: linear in $|P|$
- Normal programs under STABLE: **co-NP**.

Quick look on Expressivity

Variables but no functions:

- Horn clauses under M_P: proper subset of **IND**. Without functions, no higher order objects can be encoded (recursive definitions).
- Normal programs under STABLE: **co-NP**.

DATALOG$^\neg$ never played a role in the database community.

Idea: A program P specifies a finite family of finite sets:

$$\{\{(c_1,\ldots,c_n) : \mathcal{M} \models p(c_1,\ldots,c_n)\} : \mathcal{M} \text{ a stable model of } P\}$$

where p is a distinguished predicate.

This allows to express second-order objects in Datalog$^{\neg}$!

Methodology: Clauses do not represent recursive definitions of individual stable models, but they are viewed as constraints determining the whole set of stable models in its own right.

Develop a methodology for uniformly encoding problems.

Marek/Truszczynski, *Stable models and an alternative logic programming paradigm*, LNAI, to appear 1999, Niemelä, *LP with stable models semantics as a constraint programming paradigm*, NMR-Workshop, 1998.)

10 Multi-Agent-Systems

Kowalski: From LP to Multi-agent systems.

- No logic for change of state, no procedural syntax, input-output.
- Distributed active deductive databases are an example of a multi-agent system.
- Concurrent LP is based on message-parsing architectures. But concurrent interactions of agents are better handled by Linda-like systems.

Kowalski/Sadri, *From LP to Multi-agent systems*, **AMAI**, to appear 1999.)

Subrahmanian et al.: Heterogenous Active Agents.

- Formal Multi Agents Framework. Prototype Implementation and further developed by the army.
- Agents (arbitrary software plus an additional layer on top) are described by Agent Programs: the semantics is based on stable-like status sets.
- Programs allow also: beliefs (about other agents), time (duration of actions), uncertainty, security.

(VS et.al., *Heterogenous Active Agents*, MIT Press, 1999.)

11 Diagnosis and Banking Fees

- In principle, the problem can also be described as a clause set and the solution computed by a theorem prover.
- However, the search space is huge and so this does not work.
- Our solution: transform into positive disjunctive programs and compute minimal models. They constitute the solutions of the problem.
- Diagnosis: We use PROTEIN with a slightly modified calculus to take advantage of domain dependent pruning techniques.
- Banking Fees: positive disj. programs + constraints.

11.1 Consistency-based Diagnosis

Definition I.11.1 (Diagnosis)
A Diagnosis of $(SD, COMP, OBS)$ is a set

$$\boxed{\Delta} \subseteq COMP$$

s. t.

$$\begin{array}{ll} SD \cup OBS \cup & \{Mode(C, Ab) : c \in COMP\} \\ \cup & \{\neg Mode(C, Ab) : c \in COMP - \Delta\} \end{array}$$

is consistent.

SD: systemdescription, $COMP$: set of components, OBS: observed behaviour. Every component c comes with a mode describing the correct behaviour: $Mode(c, OK)$.

Determining Diagnosis means to compute models of $D \cup OBS$.

Minimal diagnosis correspond to minimal models.

Our Hypertableaux-Prover deals with realistic descriptions (appr. 2000 clauses) and is comparable with the special purpose system (DRUM).

(Baumgartner/Furbach/Fröhlich/Nejdl, **IJCAI 97**.)

11.2 Banking-Fees

Problem-Description

1. A credit institution sells stocks and shares to its costumers.
2. The fees depend on the value, the costumer-type, the country etc.
3. The set of rules is given in natural language.

Is the rule set deterministic and total?

Specification

1. The rules are transformed into a logic program.
2. The program contains disjunctions and negations.

Results

Example	Rules	Parameter	Time	Depth
small	15	8	0.1s	3
large	170	20	7.8s	4

(Stolzenburg/Thomas, **Automated Deduction, Vol. III**, ALS 10, 1998.)