Using Tabling to Extend Prolog with Non-Monotonic Constructs

Terrance Swift

SUNY Stony Brook

The XSB Group

- Active Members
  - Active Faculty
    * Michael Kifer (SUNY Stony Brook)
    * C.R. Ramakrishnan (SUNY Stony Brook)
    * I.V. Ramakrishnan (SUNY Stony Brook)
    * Terrance Swift (SUNY Stony Brook)
    * David S. Warren (SUNY Stony Brook)
  - Current Students
    * Ernie Johnson (SUNY Stony Brook)
    * Rui Marques (Universidade Nova de Lisboa)
    * Luis Pias de Castro (SUNY Stony Brook)
    * Guizhen Yang (SUNY Stony Brook)
- Other Major Contributors
  - Bart Demoen (KU Leuven)
  - Baoqiu Cui (IBM Almaden)
  - Juliana Freire (Bell Labs)
  - Prasad Rao (Bellcore)
  - Kostis Sagonas (Upsalla)

Overview

- Overview of Tabling and WFS
- Preference Logic Grammars
  - *programming with negation at a high-level*
- FLORA
  - *an object-oriented logic over WFS*
- Psychiatric Diagnosis
  - *Using WFS with Explicit Negation (and a little abduction)*
- Abdual
  - *Abduction as satisfiability checking over WFS*
- Environments for non-monotonic constructs.

Overview: problems with (Constraint) Logic Programming

- *Termination and Complexity*
  - *Terminates for Positive Datalog Programs*
  - *Linear for Positive Datalog Programs*
- Default Negation;
- Collection of Solutions

Default Negation: Well-Founded Semantics (WFS)

- 3-valued semantics in which elements can be true, false, or unknown.
- Main difference of WFS with other semantics is the use of *unfounded sets*.

```
s:- not p, not q, not r.
p:- not s,q.
q:- not p,r.
r:- p,not q.
```

- { `p,q,r` } is an unfounded set.

Preferences: Motivation

Expressing Preferences can be useful for grammars [GJM95, JGM98, CS01]:

if cond_1 then (if cond_2 then assign_1 else assign_2)
if cond_1 then (if cond_2 then assign_1) else assign_2

```
stmtseq(stmtseq(S,null)) --> stmt(S).
stmtseq(stmtseq(Ss,S)) -->
        stmtseq(stmtseq(Ss)),
        stmt(S).
stmt(stmt(S)) --> assign(S).
stmt(stmt(S)) --> ifstmt(S).

prefer(ifstmt(if(C,if(C1,T,E)),
        ifstmt(if(C,if(C1,T),E)))).

ifstmt(if(C,T)) --> [if],cond(C),
        [then],stmtseq(C).
ifstmt(if(C,T,E)) --> [if],cond(C),
        [then],stmtseq(C),
        [else],stmtseq(E).
```

Preferences in WFS: Theory

- A *Preference Logic Program* consists of a normal program P along with a set $Pref$ of preference clauses.
- If $P \cup Pref$ is definite, the Preference Logic Program is definite, sometimes called a *Preference Logic Grammar*.
- A *Preference Clause* of the form

  $$prefer(p(t_1, \ldots, t_n), p(u_1, \ldots, u_n)) \text{ :- } A_1, \ldots, A_n$$

  states that $p(u_1, \ldots, u_n)$ can be derivable only if $p(t_1, \ldots, t_n)$ is not.

Preferences in WFS: Theory

- Simple embedding of Preference Logic Programs into WFS
  - ensures that when a given atom succeeds, another is not (transitively) preferred to it
  - ensures that negative calls to atoms take preference overriding into account.
- Cyclic preferences are undefined in WFS embedding
- Definite preference logic programs are equivalent to *monophonic* programs, where all rules for an atom A are of the form:

$$\begin{array}{l} A \text{ :- } A_{1,1}, ..., A_{1,n_1}, not\ B_1, ..., not\ B_p \\ \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ A \text{ :- } A_{m,1}, ..., A_{m,n_q}, not\ B_1, ..., not\ B_p \end{array}$$

Preference Logic Grammars

Preferences and Tabling work together for Grammars

- Tabling allows the use of LR grammars:
  - No Need for factoring
  - In Chomsky Normal form, grammars are cubic in size of input list
- Preferences handle ambiguity
  - Arising naturally in many grammars.
  - Allows libraries of grammars to be created and maintained.
- Generate all reasonable parses and then filter via preferences.

Data Standardization

- Data standardization turns unstructured and "dirty" textual data into structured "clean" data.
  - from legacy databases
  - from the world wide web
  - from audio input
  - from optical character recognition of bit-mapped data or pdf files
- Test case was a commercial name-address "standardizer" [SHH$^+$94] that extracts standardized name and address data from textual fields.
- Prolog version had standardized data for
  - U.S. Customs Service
  - Defense Logistics Agency
  - a large investment bank
  - a large electronics distributors

Data Standardization

Example: the string

```
TO THE ORDR OF ZZZ AUTOPARTS INC
DBA PARTCO 123 WASHING
TON STREET EL SEGUNDO CA
```

Is standardized to

Organization:	`ZZZ AUTOPARTS`
Type:	`basic`

Organization:	`PARTCO`
Street:	`129 WASHINGTON ST`
PO BOX:	
City:	`EL SEGUNDO`
State:	`CA`
Zip:	`90245`
Type:	`dba`

Data Standardization

- Prolog Architecture
  1. Tokenize
  2. Collect primitive tokens into "supertokens"; e.g. 'LONG','ISLAND','CITY' into 'LONG ISLAND CITY'
  3. Apply DCG parsing rules
  4. Postprocess to check consistency, and fix mistakes
- First two steps were easily maintained: supertokenization could be generated.
- Second two steps were difficult to maintain:
  - In order to avoid ambiguous parses, the parser had states in which tokens that were not part of a first or follow set (e.g. 'ZZZ','PARTCO', 'WASHINGTON') were interpreted by default as company names, streets, cities etc.

Data Standardization

- Better Solution: generate all parses and keep the best.
- Preferred rules can call other preferred rules, so that pruning can be done for sub-parses.
- Parsing rules become very general because they do not have to account for possible ambiguity
  - An *organization sequence* is an organization
  - An *organization sequence* is an organization followed by an org relation (e.g. 'DBA') followed by an organization
  - An *organization* is defined by any words possibly followed by an org terminator (e.g. 'INC', 'ASSOC')
  - An *organization* is defined by any words possibly followed by an end of field symbol ('|').

Data Standardization

- The actual preference rules indicate that
  1. A parse with a fuller org title is preferred, e.g.

     org('ZZZ AUTOPARTS','INC')

     to

     org('ZZZ AUTOPARTS INC','')
  2. More specific parses are preferred (e.g. a parse with two companies and a relation to a parse with a single company).
- Similar parsing and preference rules are used for addresses, and for large entities (organizations and addresses).

Data Standardization

Comparison as of Jan 1999

Function	Clauses	Lines
Tokenization	94	412
Bottom-up Parse	26205	26205
Domain Information	59150	59150
Control and Utilities	727	1345
(Prolog) Top-Down	724	2082
(Prolog) Post-processing	604	2838
(PLP) Top-Down	198	686
(PLP) Post-processing	7	106

- Two parsers had similar behavior on sample DLA data
- To be taken with a small grain of salt

Data Standardization

- Preference Logic Standardizer has been used for
  - Defense Logistics Agency
  - A large pharmaceutical company
  - Several large electronics distributors
- Preference Logic is now used in a variety of standardization tasks – for aircraft parts, medical supplies, data mining of electronic catalogs, etc.

Psychiatric Diagnosis

- Wanted to see what constructs were necessary to model an external body of knowledge
  - *Diagnostic and Statistical Manual of Mental Disorders vol. IV* [Dia94]
- Part of a commercial project *Diagnostica* whose goals were:
  - to understand, for different sets of symptoms, which diagnoses were precluded, which were statisfied, and which were close.
  - to be able to compute *differential* diagnoses.
- Needed to convince psychiatrists, not just ourselves.

Psychiatric Viewpoint

- Research Psychistrists wanted to create a system to help diagnose psychistric ilnesses.
  - As a practical tool
    - Research psychiatrists tend to be too slow in precluding disorders.
    - Clinical psychiatrists tend to be too quick in precluding disorders.
  - As a research tool
    - Create a medical database with consistent entry of symptoms and diagnoses.
    - Use database for epidemiological studies

A DSM-IV Rule: Asperger's Disorder

1. Qualitative impairment in social interaction, as manifested by at least two of the following
   (a) marked impairment in the use of multiple nonverbal behaviors such as eye-to-eye gaze, facial expression, body postures, and gestures to regulate social interaction;
   (b) failure to develop peer relationships appropriate to developmental level
   (c) a lack of spontaneous seeking to share enjoyment, interest,m or achievements with other people (e.g. by a lack of showing, bringing, or pointing out objects of interest to other people).
   (d) lack of social or emotional reciprocity
2. Restricted repetitive and stereotyped patterns of behavior, interests, and activities, as manifested by at least one of the following:
   (a) encompassing preoccupations with one or more stereotyped and restricted patterns of interest that is abnormal either in intensity or focus
   (b) apparently inflexible adherence to specific nonfunctional routines or rituals.
   (c) stereotyped and repetitive motor mannerisms (e.g., hand or finger flapping or twisting, or complex whole-body movements)
   (d) persistent preoccupation with parts of objects
3. The disturbance causes clinically significant impairment in social, occupational, or other important areas of functioning
4. There is no significant clinical delay in cognitive development or in the development of age-appropriate self-help skills, adaptive behavior (other than in social interaction) and curiosity about the environment in childhood.
5. *Criteria are not met for another Pervasive Development Disorder or Schizophrenia*

Problems with DSM-IV

[GST+00] considers how preference logic extended with explicit negation and abduction can model DSM-IV

- Cyclic negative dependencies arise for certain symptom states;

  ```
  aspergers :- not autism.
  autism :- not aspergers.
  ```

  This dependency reflects a disagreement in the literature

  - Some researchers say that patients with Asperger's Disorder are autistic with high intelligence
  - Others believe that Asperger's Disorder is separate.

Problems with DSM-IV: Incomplete Knowledge

- Delays in language acquisition.
  - History not available but no evidence for delay
    - default conclusion
  - History available and it is *known* that there is no delay
    - explicit conclusion
- Modelled via Well-founded Semantics with Explicit Negation (WFSX) [ADP95]

Problems with DSM-IV: Hypothetical Reasoning

- Once the stressor (or its consequences) has terminated, the symptoms do not persist for more than an additional 6 months.
- If a patient is diagnosed with adjustment disorder, the physician is making a hypothesis about the future.

Determining whether Criteria for a Diagnosis are Met

- Well-founded semantics to model loops through exclusion rules.
  - Allows representation to stay close to DSM-IV
  - Low complexity solution
- Undefinedness of WFSX can be overridden by 3-valued preferences with a transformational semantics into WFSX.
- Satisfying a diagnosis means determining whether a query is true in a (3-valued) abductive framework over WFSX.

Using 3-valued Preferences for Diagnosis

As before preferences:

- Are *dynamic*: preferences can be derived as part of program execution
- Allow preferences to be defined on preferences themselves

Now preferences:

- are extended clauses (i.e. both default and explicit negation can occur in the body).
- can be incorporated into the abductive framework discussed below.

Using 3-valued Preferences for Diagnosis

- (Super-)User can state that one diagnosis is preferred to another when loops through exclusion rules are met;
  - `prefer(aspergers,autism)`
- Can make diagnoses undefined when criteria are met for both, even when there are no loops through exclusion rules
  - Some psychiatrists believe that the DSM-IV distinction between Alzheimer's Dementia and Major Depressive Disorder is too coarse: so can create rules
    - `prefer(major_depressive_disorder,alzheimers).`
    - `prefer(alzheimers,major_depressive_disorder).`
- Such preferences can then be overridden by preferring preferences on preferences

FLORA-2

FLORA-2 [GY01] extends XSB via the object logic F-logic [KLW95].

- Allows dynamic creation of subclass, isa, and various attribute relations – including schema information.
- Allows "second-order" rule schemata via HiLog
- Allows aggregation implemented via `findall/3`.
- Allows inherited and non-inherited scalar and multi-valued attributes.

Certain types of inheritance in FLORA-2 can be viewed as non-monotonic constructs.

FLORA-2

- Behavioral inheritance (`=> =>>`):
  - `person[name => string]` roughly means that any subclass of person or object that "isa" person may have a name attribute of type `string`
- propagates from classes to subclasses
- is (anti-)monotonic on subclass relationship:
  - `a::b` implies `a` has all the `=>, =>>` attributes of `b`
  - Used for typing information.

FLORA-2

- Structural inheritance (`-> ->> *-> *->>`):
  - `person[name *-> string]` roughly means that any object of the class person may have a `name` attribute whose *value* is the class atom `string`
- propagates from classes to objects
- is non-monotonic:
  - can be overridden by direct definitions of attributes, or inheritance via subclasses.
  - can conflict with inheritance arising from other subclasses

FLORA-2 Logic of Inheritance:

- Is two-valued iff $(isa \cup subclass)$ relation is acyclic.
- Can infer term equality (or fail) for conflicting structural scalar attributes:
  - Under equality of scalar attributes `mary[spouse -> joe]` and `mary[spouse -> john]` implies `joe = john`.
- Implemented via tabling and well-founded negation

FLORA-2 is currently used by about half-a-dozen companies to prototype various ontology-oriented applications – agents, medical support systems, etc.

FLORA-2: A Simple Fragment

```
passatC[hasConfig(T) -> Q]:-
        T:stage1,
        S is T - 1,
        passatC[hasConfig(S) -> Q],
        tnot(passatC[change(S) -> newConfig(T)]).

passatC[hasConfig(T) -> Q]:-
        T:stage1,
        S is T - 1,
        passatC[change(S) -> Q].

passatC[change(S) -> newConfig(T)]:-
       T is S + 1.

passatC[hasConfig(0) -> newConfig(0)].

stage1[ready_edb -> true].
stage1[ready_changes -> true].

0:stage1.
0:stage2 :- S[ready_edb -> true].
T:stage1 :- S[ready_changes -> true], T is S + 1.
```

FLORA-2: Program Characteristics

- Heavy use of tabling to traverse subclass and isa relations for inherited attributes.
- New subclass, isa, and attribute tuples may be derived in the course of executing a given goal.
- Tabled negation used to identify conflicts, implement overriding.
- Tabled negation may also be used in source program.

A Stratification Hierarchy

Dynamically stratified

```
s:- not p, not q, not r.
p:- not s, q.
q:- not p, r.                r:- p, not q.
```

LR-dynamically stratified

```
s:- not p, not q, not r.
p:- q,not s.
q:- r, not p.                r:- p, not q.
```

Weakly Stratifed

```
s:- not p, not q, not r.
p:- not s, q.                p.
q:- not p, r.                r:- p, not q.
```

LR-weakly stratified

```
s:- not p, not q, not r.
p:- q, not s.                p.
q:- r, not p.                r:- p, not q.
```

Modularly Stratified

```
s:- not p, not q, not r.
p:- not s, q.                m.
q:- not p, r.                r:- p, not q.
```

Locally-Stratified

no cycles through negation in atom dependency graph

Predicate Stratified

no cycles through negation in predicate dependency graph

A Stratification Hierarchy
(Results from [SSW00, DS01]

Dynamically stratified

quadratic; delay required

LR-dynamically stratified

quadratic; no delay required

Weakly Stratifed

quasi-linear; delay required

LR-weakly stratified

quasi-linear; no delay required

Modularly Stratified

Locally-Stratified

Predicate Stratified

Complexity of Stratification

Quadratic nature of WFS arises from cascading unfounded sets *within* a dynamic stratum.

```
s:- not p0, not q0, not r0.

p0:- not s,q0.
p0:- p1.
q0:- not p0,r0.
r:- p0,not q0.

p1:- not s,q1.
p1:- p2.
q1:- not p1,r1.
r:- p1,not q1.

    :
```

Does such cascading arise in FLORA-2?
Profiles so far do not indicate this.

Using NM Constructs for Solving Combinatorial Problems

Within LP, combinatorial problems in Planning, Diagnosis, Configuration Management and other areas have been approached using:

- Abduction (e.g. [KM97, ND00]);
- Stable Models (e.g. [NBG$^+$01, ELW00, Syr00]);
- Logical Constraints;
- and just plain Prolog.

Advantage of LP is that the problem can be modelled simply and transparently.

Combinatorial Problems: From WFS to Stable Models

Stable Models are:

- Defined for ground programs
- Guess at model M for P
  - Eliminate negative literals consitent with M.
  - Eliminate rules with negative literals inconsistent with M.
  - Find LFP of transformed program
- Computation of one stable model is NP-complete.

```
r:- q,not r.
p:- not q.
q:- not p.
```

Combinatorial Problems: Stable Models

- WFS is sound approximation of Stable Models
- Optimized method:
  1. construct WFS non-ground "model" M_w
  2. construct residual program P_r from M_w
  3. ground and guess for P_r.
- Implemented via XSB's SModels interface.
  - First two steps performed by XSB
  - Last step performed by SModels [NS97]

Combinatorial Problems: Abduction over WFSX

Abductive Framework $\mathcal{A}$ is:

- A program P.
- A set A of abducible object literals and their explicit negations.
- A set I of Integrity Rules.

Abductive solution to a query Q to $\mathcal{A}$ is a WFM of $P \cup A$ in which no rule in I is true.

P:
```
r:- q, not(r*).
p:- not(q*).
q:- not(p*).
```

A:
```
q*,-q*,p*,-p*,r*,-r*
```

I:
```
⊥ :- p, -p*.
⊥ :- q, -q*.
⊥ :- r, -r*.
```

Combinatorial Problems: Abduction over WFSX

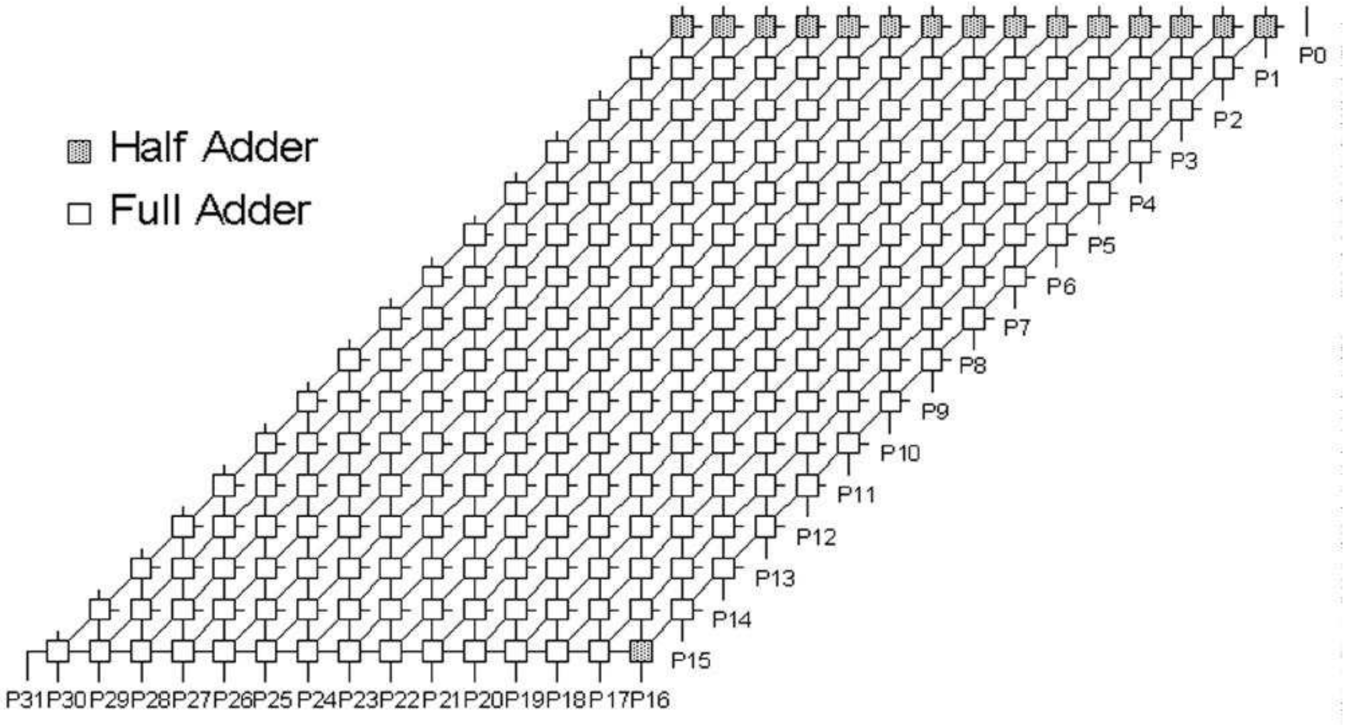
- Can be implemented via a tabling method, Abdual [APS01], similar to that used by XSB.
- Stable models and partial stable models can be implemented via Abdual
- Constructs a *dual* program $Dual_P$ in which an atom p_{dual} are true iff p is false.
  - Dual program must be (mostly) ground
  - Need to use dual program upon a call through negation that may require abduction.
- Similar to Stable Models, Abdual has polynomial complexity for fixed-parameter problems.
- Only meta-interpreters of Abdual currently exist – full implementation requires engine changes to XSB.

Combinatorial Problems

- Advantages of Stable Models
  - Clear semantics; subset of disjunctive logic programs;
  - Good implementations: SModels, DLV [ELM+97].
- Advantages of Abduction
  - Clear semantics.
  - May require less grounding – can be combined with CLP
  - Goal-orientation – useful for diagnosis, planning of separable events.
  - Non-tabled systems:
    - ACLP (`www.cs.ucy.ac.cy/aclp`) combines abduction and constraints over Prolog
    - SLDNFA (`www.cs.kuleuven.ac.be`) allows non-ground abduction over Prolog.
  - May be combined more easily with Prolog-style environments.

Combinatorial Problems

Fault Diagnosis of 16-bit multiplier [AAB+01]



- 240 half + full adders, 2406 gates in all
- Given two numbers to multiply, faults were diagnosed for correct output and for 31 incorrect outputs;

Combinatorial Problems

- Single Faults:
  - Constraint Propagation (cf. [Sim89]) (SICStus): 1 sec
  - Backtracking (GNU Prolog Native): 133 sec
  - Tabled Abduction (XSB): 187 sec
  - Stable Models (XSB + SModels): 2624 secs
- For Double Faults, Tabled Abduction was fastest.
- Shows that NMR can be useful for combinatorial optimization, but is not automatically so.
- Most of the approaches appear competitive or better than TMS-style systems.
- More detailed study of circuit diagnosis appears useful

Programming Environments

	Compilation	Interface	Debugger
XSB	to bytecode	Yes (2)	Yes (2)
FLORA-2	to XSB	Yes	Yes
LMC	to XSB	Yes	Yes
WFSX	to XSB	No	No
Preferences	to XSB	No	No
Annotations	Meta-interpreter	No	No
Abdual	Meta-interpteter	No	No

Significant amount of work needed to make these constructs easily available to programmers.

Summary

- Non-monotonic constructs can be make programs "more declarative".
- Requires applications, environments, a good engine, and education.

References

[AAB+01] J. J. Alferes, F. Azevedo, P. Barahona, C. Damásio, and T. Swift. Logic programming techniques for solving circuit diagnosis. Technical report, Universidade Nova de Lisboa, 2001.

[ADP95] J. Alferes, C. Damásio, and L. M. Pereira. A logic programming system for non-monotonic reasoning. *Journal of Automated Reasoning*, 14(1):93–147, 1995.

[APS01] J. Alferes, L. M. Pereira, and T. Swift. Well-founded abduction and generalized stable models via tabled dual programs. Full version of paper presented in ICLP 1999, pp. 426-440, 2001.

[CS01] B. Cui and T. Swift. Preference logic grammars: Fixed-point semantics and application to data standardization. *Artificial Intelligence*, 2001. To Appear.

[Dia94] *Diagnostic and Statistical Manual of Mental Disorders*. American Psychiatric Association, Washington,DC, 4th edition, 1994. Prepared by the Task Force on DSM-IV and other committees and work groups of the American Psychiatric Association.

[DS01] J. Dix and T. Swift. Linear subclasses of the well-founded semantics, 2001. In preparation.

[ELM+97] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A deductive system for non-monotonic reasoning. In *International Conference on Logic Programming and Non-Monotonic Reasoning*, 1997.

[ELW00] E. Erdem, V. Lifschitz, and M. Wong. Wire routing and satisfiability planning. In *Proceedings of Computational Logic, 2000*, 2000.

[GJM95] K. Govindarajan, B. Jayaraman, and S. Mantha. Preference logic programming. In *International Conference on Logic Programming*, pages 731–746, 1995.

[GST+00] J. Gartner, T. Swift, A. Tien, L. M. Pereira, and C. Damásio. Psychiatric diagnosis from the viewpoint of computational logic. In Lloyd et. al, editor, *International Conference on Computational Logic*, pages 1362–1376. Springer-Verlag, 2000. LNAI 1861.

[GY01] Michael Kifer Guizhen Yang. *FLORA-2: User's Manual*, 2001. Avaliable via xsb.sourceforge.net.

[JGM98] B. Jayaraman, K. Govindarajan, and S. Mantha. Preference logic grammars. *Computer Languages*, 1998. 179-196.

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, July 1995.

[KM97] A. Kakas and C. Mourias. ACLP: Flexible solutions to complex problems. In *International Conference on Logic Programming and Non-Monotonic Reasoning*, 1997.

[NBG+01] A. Nogueira, M Balduccini, M. Gelfond, R. Watson, and M. Barry. An A-prolog decision support system for the space shuttle. In *Practical Applications of Declarative Languages*, 2001.

[ND00] B. Van Nuffelen and M. Denecker. Problem solving in id-logic with aggregates: Some experiments. In *NMR 2000*, 2000. Available via www.cs.kuleuven.ac.be.

[NS97] Ilkka Niemelä and Patrik Simons. SModels — An implementation of the stable model and well-founded semantics for normal LP. In *International Conference on Logic Programming and Non-Monotonic Reasoning*, pages 420–429. Springer-Verlag, 1997.

[SHH+94] T. Swift, C. Henderson, R. Holberger, J. Murphey, and E. Neham. CCTIS: an expert transaction processing system. In *Sixth Conference on Industrial Applications of Artificial Intelligence*, pages 131–140, 1994.

[Sim89] H. Simonis. Test generation using the constraint logic programming language chip. In *International Conference on Logic Programming*, pages 101–112, 1989.

[SSW00] K. Sagonas, T. Swift, and D. S. Warren. The limits of fixed-order computation. *Theoretical Computer Science*, 254(1-2):465–499, 2000.

[Syr00] T. Syrjanen. Including diagnostic information in configuration models. In *Proceedings of Computational Logic, 2000*, 2000.