Tabling for Logic-based Artificial Intelligence

Terrance Swift

EPIA 99: 9th Portuguese Conference on Artificial Intelligence

Foreward

- Tabling is an evaluation method — it can be used to implement various logics and knowledge represenatation formalisms, but does not provide these formalisms directly.
- Thus tabling does not add any new ideas to LP *from the perspective of applications!*
- However, tabling is
  - Complete, and terminating for well-understood classes of programs.
  - Efficient: query oriented, of low complexity, amenable to low-level implementations (XSB, YAP)
  - Integratable with Prolog and with constraints.
- Tabling thus makes logic programming viable for new situations and problems.

Overview

- Introduction to tabling and its applications
- Case study: Using WFSX for psychiatric assesment.
- Case study: Using tabled aggregation for preferences and quantitative reasoning.

Overview

- Tabling can form a means of implementing the Well-Founded Semantics [127], and its extensions such as the Well-Founded Semantics with Explicit Negation *[3, 5]*, or Disjunctive Logic Programs under various semantics.
- Tabling can be used as a preprocessor for stable models and extensions [56, 57], or can compute these models directly by means of abduction *[34, 6]*.
- Tabling can be used as a means to aggregate solutions for annotation logics [74],[75],[89],[115],[41], *[35]* or preference logics [58],[29].

Overview

- A (very partial!) bibliography
  - Formulations of Tabling: [48], [19], [122], [46], [71], [112], [128], [132], [23], [40], [81], [123], [24], [21], *[32]*, [82], *[53]*, [64], [134], [16], [13], [17], [18], [119], [110], [120], [136]
    - Clear progress from Definite LPs, through Stratified and Normal LPs to Abductive LPs, Disjunctive LPs, and Quantitative LPs.
  - Implementation and Systems [8], [129], [79], [49], [67], [98], [121], [109], [39], [9], [22], *[51]*, [133], [99], [135], *[101]*, *[52]*, [42], [43], *[54]*, [93], [107], [108], [31],
    - XSB, YAP, Lola, and others
  - Optimizations [36]

Overview

- Magic Sets and Related Research (e.g. Alexander Method)
  - Formulation [102], [7], [111],[124], [20], [87], [85], [10], [70], [91], [117], [47], [60], [96], [11], [114], [104] [84]
    - References are a little dated.
  - Implementation and Systems [8], [25], [125], [97], [113], [130], [131], [44], [59], [63], [86]
    - LDL, Aditi, Coral and others.
  - Optimizations [95], [88],[105], [106],[103], [72], [116], [62], [61], [118], [15]

Overview

- Practical and Research Applications
- Parsing [1], [78], [30],
  - Tabling eliminates need for transformations from left-recursive grammars to right recursive grammars
  - Tabling allows creation of chart parsers and can simulate Earley Recognition.
  - Preference Logic Grammars create a powerful mechanism for handling ambiguities in Grammars
    - To be discussed below

Overview

- Program Analysis of Imperative, Functional, and Logic Programs: [90], [69], [68], [65], [100], [37], [14], [27], [66]
- Abstraction of programs to be analyzed creates positive loops even when concrete programs do not loop

Example: Groundness Analysis (from [26])

```
append([],Ys,Ys).
append([X|Xs],Ys,[X|Zs]) :- append(Xs,Ys,Zs).
```

$\Downarrow$

```
g_append(g,Y,Y).
g_append(X1,Ys,Z1) :- iff(X1,X,Xs), iff(Z1,X,Zs),
                      g_append(Xs,Ys,Zs).

iff(g,g,g).
iff(n,n,n).
iff(n,n,g).
iff(n,g,n).
```

Overview

- Software Verification [92], [83]
- The modal-mu calculus [76] uses lfp quantification (μx) and gfp quantification (νx) to test properties of transition systems:
  - "Along some computation path, the property P holds infinately often", in the modal-mu calculus:

$$\nu x_1 . \mu x_2 . < \_ > ((P \wedge x_1) \vee x_2)$$

  - $\nu = \neg \mu \neg$, so a lfp tabling engine can compute gfps by means of default negation.

Overview

- Quantitative and Probabilistic Reasoning [2], [119], [41]
  - To be discussed below
- Graphics and Data Visualization [55], [28]
- Diagnosis Systems *[38]*, *[33]*, [45]
  - To be discussed below
- Abduction *[6]*
- Updates *[80]*, *[4]*
- Machine Learning *[77]*
- Other [12], [94]

Case Study 1: Psychiatric Assesement

- Problem: within a medical group, want to assess psychiatric patients efficiently, and in a standard, well-documented manner
- Joint work with Allen Tien and Joseph Gartner

Case Study 1: Psychiatric Assesment

- Difficulties:
  - physicians may have incomplete information
  - many hundreds of possible diagnoses in *Diagnostic and Statistical Manual of Mental Disorders, 4th ed*, published by the Americal Psychiatric Association
    - Does a patient have Altzheimer's Dementia or Major Depression?
    - Does a child have a pervasive development disorder such as Autism, or Mutism?
    - Does a child have Attention Deficit Disorder or an Adjustment Disorder?
  - different physicians may assess patients differently
    - Research psychiatrists are too slow to preclude diagnoses
    - Clinical psychiatrists are too quick to preclude diagnoses, or may not consider various diagnoses.

Practical Applications

Diagnostic Criteria for Dementia of the Alzheimer's Type

1. The development of multiple cognitive defects manifested by both
   (a) memory impairment (impaired ability to learn new information or to recall previously learned information)
   (b) one (or more) of the following cognitive disturbances
      i. aphasia (language disturbance)
      ii. apraxia (impaired ability to carry out motor activities despite intact motor function)
      iii. agnosia (failure to recognize or identify objects despite intact sensory functions)
      iv. disturbance in executive functioning (i.e., planning, organizing, sequencing, abstracting)
2. The cognitive deficits in criteria A1 and A2 each cause significant impairment in social or occupational functioning and represent a significant decline from a previous level of functioning.
3. The course is characterized by gradual onset and continuing cognitive decline.
4. The cognitive deficits in Criteria A1 and A2 are not due to any of the following
   (a) other central nervous system conditions that cause progressive deficits in memory and cognition (e.g. cerebrovascular disease, Parkinson's disease, Huntington's disease, subdural hematoma, normal-pressure hydrocephalus, brain tumor)
   (b) systemic conditions that are known to cause dementia (g. hypothyroidism, vitamin B_{12} or folic acid deficiency, niacin deficiency, hypercalcemia, neurosyphilis, HIV infection
   (c) substance-induced conditions
5. The deficits do not occur exclusively during the course of a delirium
6. *The disturbance is not better accounted for by another Axis I disorder (e.g., Major Depressive Disorder, Schizophrenia)*

Case Study 1: Psychiatric Assesment

Goal: represent DSM-IV as a logic program and implement in a system for physicians. This will aid the physician in making decisions, and revising these decisions as information becomes available.

- DSM-IV gives rise to many knowledge representation difficulties
  - How to represent conflicting diagnoses

```
'Altzheimer's Dementia' :-
          not_better 'Major Depressive Disorder'
'Major Depressive Disorder':-
          not_better 'Altzheimer's Dementia'
```

- How to represent incomplete information
  - E.g. tests have not been run, a full case history is unavailable, etc.

Case Study 1: Psychiatric Assesment

- How to represent speculative information
  - A criterion for Adjustment Disorder is: Once the stressor (or its consequences) has terminated, the symptoms to not persist for more than an additional 6 months.
- How to represent temporal information
  - The disturbance does not occur exclusively during the course of a delirium.
  - If severe anxiety occurs after the onset of depresssion, it is considered part of the depresssion; if before the onset of a depression, the patient may have multiple diagnoses.

Case Study 1: Psychiatric Assesment

Initial Cut

- Represent conflicting diagnoses as unknown in the well-founded semantics.
  - Alternatively, assign a strength of belief to each diagnosis or assign preference rules to eliminate conflicting diagnoses.
- Represent incomplete vs complete information about criteria through default and explicit negation
  - Alternatively, probability intervals could be assigned to the criteria
- Represent speculative information by means of abduction
- Defer decisions for special temporal representation

Case Study 1: Psychiatric Assesment

- Default negation: conclude a statement to be false because the statement cannot be proved to be true.
- Explicit negation: requires a statement to be proved false,
- Examples:
  – Default: A patient does not have a brain tumor because he presents no symptoms, or because the symptoms that he presents can be better accounted for in some other way.
  – Explicit: A patient does not have a brain tumor because an MRI was run, and the brain was seen to be healthy.
  – Default: A 5-year old child speaks normally, so the physician assumes that there was no significant delay in language acquisition.
  – Explicit: A case history is taken from the child's parents and it can be determined that there was no significant delay in language acquisition.

Case Study 1: Psychiatric Assesment

- Programs with explicit and default negation are called *extended logic programs*.
- The well-founded semantics with explicit negation (WFSX) extends WFS to extended logic programs.
- Answer sets extend stable models to extended logic programs.
- Both obey the *coherency principle*: Infer the default negation $not(A)$ of an atom A from its explicit negation $-A$ (where $A = - - A$).
- Extended logic programs have been shown to solve a variety of problems in default and hypothetical reasoning, reasoning about actions, etc. Their use here is just one case of their applicability.

Case Study 1: Psychiatric Assesment

- Simple Example:

```
diagnosis:- symptom.
-symptom.
```

- WFSX can be linearly embedded into WFS *[32]*
- Explicitly negative symptoms are not revisable, while default negative symptoms are revisable.

Case Study 1: Psychiatric Assesment

- Question: so where does tabling come in?
- Answer: through a meta-interpreter
- WFS reflects through meta-interpreter.

```
:- table demo/1.

demo(true).
demo((X,Y)):- !,demo(X),demo(Y).
demo(not(X)):- !,tnot(demo(Y)).
demo(X):- clause(X,Y),demo(Y).
```

Case Study 1: Psychiatric Assesment

- The WFSX embedding can also be implemented as a meta-interpreter

```
demo(A):- demo_t(A).
demo(A):- demo_tu(A).

:- table demo_t/1, demo_tu/1.
demo_t(true).
demo_tu(true).
demo_t(',',(A,B)):-!, demo_t(A),demo_t(B).
demo_tu(',',(A,B)):-!, demo_tu(A),demo_tu(B).
demo_t(not A):-!, tnot(demo_tu(A)).
demo_tu(not A):-!, tnot(demo_t(A)).
demo_t(A):- clause(A,B), demo_t(B).
demo_tu(A):-clause(A,B), demo_tu(B),
     exchange(A,A_opp),demo_tu(not A_opp).

exchange(-A,A):-!.
exchange(A,-A).
```

Case Study 1: Psychiatric Assesment

- This meta-interpreter can be partially evaluated to a more efficient program, which is a package in XSB.
- XSB has partially implemented WFSX with abduction for a psychiatric assesment system called *Diagnostica*. Web address:

  `http://www.medicinerules.com`

- Process of writing a meta-interpreter for an extension of the well-founded semantics, and then specializing the meta-interpreter is a standard technique for using tabling for non-monotonic programming (used for WFSX, Annotated Logic Programs, Abductive Logic Programs, F-logic [73]).

Case Study 2: Tabled Aggregation

- We have seen that tabling helps compute queries to `path/2`

```
:- table path/2.
path(X,Y):- edge(X,Y).
path(X,Y):- path(X,Z),edge(Z,Y).

edge(a,b).
edge(b,c).
edge(c,d).
```

- But how do we compute the *shortest* path?

Case Study 2: Tabled Aggregation

Consider the extension of the path/2 predicate that counts the cost of a path between two vertices in a graph.

```
:- table path/3.
path(X,Y,C) :-   path(X,Z,C1),
                 edge(Z,Y,C2),
                 C is C1 + C2.
path(X,Y,C) :-   edge(X,Y,C).
```

Naive solution: compute all paths, take the shortest.

Case Study 2: Tabled Aggregation

For the following edb:

```
edge(1,2,1).            edge(2,2,1).
edge(2,3,1).            edge(2,4,1).
edge(3,5,2).            edge(4,5,1).
```

We get the following trace:

```
| ?- path(1,5,X).

X = 4;

X = 3;

X = 5;

X = 6;

X = 7;

++Warning: Removing incomplete tables...

yes
| ?-
```

Case Study 2: Tabled Aggregation

- Better solution, change the ANSWER CLAUSE RESOLUTION step to include a computable *aggregation function*, i.e. a function from bags of answer nodes to sets of answer nodes.
  - E.g. let the set of answer nodes for a forest $\mathcal{F}$ and subgoal S be $A_{S,\mathcal{F}}$, the aggregated set of answers is $aggregate(A_{S,\mathcal{F}})$.
- In $\mathrm{SLG}_{variance}$, $aggregate$ defines the set of answers, up to variance

$$aggregate(p(a,Y) \text{ :- } |, p(a,a) \text{ :- } |, p(a,X) \text{ :- } |\}) \\ = \{p(a,Y) \text{ :- } |, p(X,a) \text{ :- } |\}$$

- In $\mathrm{SLG}_{subsumption}$, $aggregate$ defines the most general set of answers

$$aggregate(\{p(a,Y) \text{ :- } |, p(a,a) \text{ :- } |\}) \\ = \{p(a,X) \text{ :- } |\}$$

Case Study 2: Tabled Aggregation

- ANSWER CLAUSE RESOLUTION: Let $\mathcal{E}$ be an SLG_X evaluation with answer aggregation function $aggregate$. Let $\mathcal{F}_n$ contain a non-root node N whose selected literal S is positive, and let $Ans \in aggregate(A_{\mathcal{F}_n,S})$ and N_{child} be the SLG resolvent of N and Ans on S. Assume that in $\mathcal{F}_n$, N does not have a child N_{child}. Then add N_{child} as a child of N.

Case Study 2: Tabled Aggregation

- Generalizing, the aggregate function may define *preferred* answers. Let `prefer/2`, define a given partial order. Then define $aggregate(A_{S,\mathcal{F}})$ to consist of those answer templates in $A_{S,\mathcal{F}}$ that are maximal in `prefer/2`.
- This is done automatically with the XSB predicate `filterPO(#Call,+PO_functor)`.

Case Study 2: Tabled Aggregation

```
shorter_path(X,Y,C) :-            filterPO1(sp(X,Y),prefer_sp).

sp(X,Y,C) :-                      shorter_path(X,Z,C1),
                                  edge(Z,Y,C2),C is C1 + C2.
sp(X,Y,C) :-                      edge(X,Y,C).

prefer_sp(sp(F,T,X),sp(F,T,Y)):- X < Y.
```

Tabled Aggregation in XSB

Now XSB terminates:

```
?- shorter_path(1,5,C).

C = 4;

C = 3;
```

But how can we avoid returning non-optimal solutions?

Case Study 2: Tabled Aggregation

`filterPO1/2` is only useful for demonstration.

```
filterPO(Call,PO):-
          filterPO1(Call,PO),fail.
filterPO(Call,PO):-
          filterPO1(Call,PO).
```

```
:- import filterPO/2 from aggregs.

shortest_path(X,Y,C) :- filterPO(sp(X,Y),prefer_sp).

sp(X,Y,C) :- shortest_path(X,Z,C1),
       edge(Z,Y,C2),C is C1 + C2.
sp(X,Y,C) :- edge(X,Y,C).
```

Now the trace is

```
?- shorter_path(1,5,C).

C = 3;

no
```

Case Study 2: Tabled Aggregation

- `filterPO/2` can be used to implement a simple *Preference Logic*, that augments Definite Programs with Preference Clauses.
- Its grammatical form, Preference Logic Grammars, can be quite useful for specifying and resolving ambiguous grammars.
- Example: the "dangling-else" problem [58]:

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

can be disambiguated via

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

Case Study 2: Tabled Aggregation

- A case study [30] of moving from DCGs to PLGs as implemented with tabled aggregation.
  - Test case was a commercial name-address "standardizer" that extracts standardized name and address data from textual fields.
  - The standardizer had about 90,000 lines of code, including 6,000 lines of parsing code.
  - Moving to PLGs reduced size of parsing code to 1,500 lines.
- Preference Logic has also been used in data mining applications to infer association rules. Set up preferences between rules based on length of conditions, on support and justification, on hierarchy of elements in conditions, etc.

Case Study 2: Tabled Aggregation

Can we go from preferences to annotations?

- Define an *annotation domain*, $\mathcal{T}$ as a semi-lattice, i.e. a partial order with a join operation.
- Define a single (infix) predicate `:/2`, e.g. $Atom : Annotation$, in which the second argument consists of elements of $\mathcal{T}$.
- add axiom

  $(A : V_1 \wedge A : V_2) \Rightarrow (A : V_3 \wedge \; V_3 = join(V_1, V_2))$

- Define the preference rule

  `prefer(A1:Ann1,A2:Ann2):- Ann1` $>_{\mathcal{T}}$ `Ann2.`

Case Study 2: Tabled Aggregation

- $\mathcal{T}$ may denote an interval $[\infty, 0]$
- The join operation is simply max on $\mathcal{T}$ (i.e. a min relation)

```
:- import filterPO/2 from aggregs.

shortest_path(X,Y):C :- filterPO(sp(X,Y):C,sp_prefer).

sp(X,Y):C :- shortest_path(X,Z):C1,
       edge(Z,Y):C2,C is C1 + C2.
sp(X,Y):C :- edge(X,Y):C.


A:V3 :- A:V1,A:V2,join(V1,V2,V3).

prefer(A1:Ann1,A2,Ann2):- Ann1 < Ann2.
```

Case Study 2: Tabled Aggregation

Examples:

- Multi-valued logics
  - $\mathcal{T}$ may denote the *Belnap* lattice: $\{\perp, \mathbf{f}, \mathbf{t}, \top\}$
    - Query `?- p:X` to program

      `p:t.        p:f`

      gives `p:`$\top$
  - $\mathcal{T}$ may denote other lattices containing default truth values, or bilattices.
  - (Definite) Bilattice Logic Programs [50] can be embedded into (Definite) Annotated Logic Progrms.

Case Study 2: Tabled Aggregation

- Possible-world semantics
  - For $Atom : Annotation$, $Annotation$ denotes sets of worlds in which $Atom$ is true
  - `prefer/2` is the subset relation
  - the join operation is set union
  - Useful for certain temporal logics, amalgamated databases, agents, etc.
    - Consider the database
      `p:[{a},f]. p:[{b},t].`
      The query `?- p:[{a},X]` gives $X = t$, while `?- p:[{a,b},X]` gives $X = \top$.

Case Study 2: Tabled Aggregation

Examples: Quantitative Reasoning

- $\mathcal{T}$ may denote a "strength of belief" (e.g. [126])
  - `prefer/2` is a numerical comparison
  - the join operation is a max operation
- $\mathcal{T}$ may denote probability intervals (e.g. [89, 41])
  - For $Atom : Annotation$, $Annotation$ denotes a probability interval which $Atom$ is true
  - `prefer/2` is an interval containment relation
  - The join operation is an interval intersection operation
  - E.g. for the program

    `p:[0.7, 0.8].        p:[0.78, 0.92].`

    The query `?- p:X` gives $X = p : [0.78, 0.8]$
- Independant probabilities can be handled in this way, as well as ceratin cases of ignorance and correlation.

Case Study 2: Tabled Aggregation

- The aggregate function may also be changed, so that $aggregate(f_{A,S})$, computes the join of all answer nodes in $f_{A,S}$.
- This gives a more efficient implementation, using the XSB library predicate `filterReduce/3`
- A meta-interpreter for definite annotated programs is now a package of XSB, while stratified annotated programs will be included in the next release.
- Semantics for Well-founded annotated programs has been proposed [115], and work is underway on their tabled implementation.

Conclusions

- Tabling can provide an implementation of Extended Programs, with their rich traditions, or of Annotated Programs with their rich traditions.
- A semantics for a unifying class of programs called *Coherent Well-Founded Annotated Programs (COWAPs)* was recently proposed combining both types of programs *[35]*.

Conclusions

Tabling Provides

- Recursive Query Computation: (Grammars, Program Analysis, Program Verification, Diagnosis)
- Well-Founded Semantics: (Program Verification, Diagnosis)
- Aggregation of Computation Paths: (Grammars, Database Mediators, Data Mining)

Tabling is (mostly) orthogonal to constraints, and work is underway to combine the two.

Conclusions

- XSB and source code is available via

  `http://www.cs.sunysb.edu/~sbprolog`

- Slides are available via

  `http://www.cs.sunysb.edu/~tswift`

Conclusions

- Tabling in XSB may be useful if
  - You want a programming system to construct A.I. Applications
  - You are willing to work with the developers if using negation very heavily or using tabled aggregation heavily
- Several groups depend heavily on XSB
  - XMC: A research group interested in model-checking and concurrency issues.
  - XSB, Inc (Stony Brook, NY): A business focusing on data cleaning and mining with no formal relation to XSB.
  - MD Logic (Towson, Md): A business focussing on information systems for mental health
  - KnowledgeBus (Hanover, Md): A business focussing on the creation of ontologies for A.I. domains.

References

[1] E. Aarts. *Investigations in Logic Language and Computation.* PhD thesis, University of Utrecht, 1995.

[2] S. Adah and V. S. Subrahmanian. Amalgamating knowledge bases, II: Algorithms, data structures and query processing. *Journal of Logic Programming*, 1996.

[3] 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.

[4] J. Alferes, J. Leite, L. M. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic logic programming. In *Procs of the Sixth Interntional Workshop on Knowledge Representation and Reasoning*, pages 98–109, 1998.

[5] J. Alferes and L. M. Pereira. *Reasoning with Logic Programming*, volume 1111 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, Berlin Germany, 1996.

[6] J. Alferes, L. M. Pereira, and T. Swift. Well-founded abduction via tabling dual programs. In *International Conference on Logic Programming*, 1999. To Appear.

[7] F. Banchilhon, D. Maier, Y. Sagiv, and J. Ullman. Magic sets and other strange ways to implement logic programs. In *PODS*. ACM, 1986.

[8] F. Banchilhon and R. Ramakrishnan. An amateur's introduction to recursive query processing strategies. In *Proc. of SIGMOD 1986 Conference*, pages 16–52. ACM, 1986.

[9] J. Barklund. Tabulation of functions in logic programs. Technical report, Uppsala University, 1995.

[10] C. Beeri and R. Ramakrishnan. On the power of magic. *J. Logic Programming*, 10(3):255–299, 1991.

[11] C. Beeri, R. Ramakrishnan, D. Srivastava, and S. Sudarshan. Extending the well-founded and valid semantics for aggregateion. In *ILPS*, 1993.

[12] R. Bol. Loop checking in partial deduction. *Journal of Logic Programming*, 16:25–46, 1993.

[13] R. Bol and L. Degerstedt. Tabulated resolution for well-founded semantics. *Journal of Logic Programming*, 38(1):31–55, February 1998.

[14] D. Boulanger. Fine-grained goal-directed declarative analysis of logic programs. *Proceedings of the International Workshop on Verification, Model Checking and Abstract Interpretation*, 1997. Available through http://www.dsi.unive.it/˜bossi/VMCAI.html.

[15] S. Brass. SLDMagic — an improved magic set technique. In B. Novikov and J. Schmidt, editors, *Advances in Databases and Information Systems*, 1996.

[16] S. Brass and J. Dix. Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. *Journal of Logic Programming*, 32(3):207–228, 1997.

[17] S. Brass and J. Dix. Characterizations of the Disjunctive Well-founded Semantics: Confluent Calculi and Iterated GCWA. *Journal of Automated Reasoning*, 20(1):143–165, 1998.

[18] S. Brass and J. Dix. Semantics of (Disjunctive) Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, 38(3):167–213, 1999.

[19] D.R. Brough and A. Walker. Some practical properties of logic programming interpreters. In H. Aiso, editor, *International Conference on FIfth Generation Computer Systems*, pages 149–158, 1984.

[20] F. Bry. Query evaluation in recursive databases: Bottom-up and top-down reconciled. In *Deductive and Obkect-Oriented Databases*, pages 25–44, 1990.

[21] W. Chen. Query evaluation of alternating fixpoint logic. *ACM Transactions on Database Systems*, 1996.

[22] W. Chen, T. Swift, and D. S. Warren. Efficient top-down computation of queries under the well-founded semantics. *J. Logic Programming*, 24(3):161–199, September 1995.

[23] W. Chen and D. S. Warren. Computation of stable models and its integration with logical query evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 1995.

[24] W. Chen and D. S. Warren. Tabled Evaluation with Delaying for General Logic Programs. *JACM*, 43(1):20–74, January 1996.

[25] D. Chimenti, R. Gamboa, R. Krishnamurthy, S. Naqvi, S. Tsur, and C. Zaniolo. The LDL system prototype. *IEEE Transactions on Knowledge and Data Engineering*, 2:76–89, 1990.

[26] M. Codish and B. Demoen. Analysing logic programs using "Prop"-ositional logic programs and a magic wand. In *ICLP*, pages 114–130, 1993.

[27] M. Codish, B. Demoen, and K. Sagonas. Semantics-Based Program Analysis for Logic-Based Languages using XSB. *Springer International Journal of Software Tools for Technology Transfer*, 2(1):29–45, November 1998.

[28] M. Consens, A. Mendelzon, and D. Vista. Deductive database support for data visualization. Technical report, U. Toronto, 1994.

[29] C. Crowner, K. Govindarajan, B. Jayaraman, and S. Mantha. Preference logic grammars. *Computer Languages*, 1999. To apear.

[30] B. Cui, T. Swift, and D. S. Warren. A case study in using preference logic grammars for knowledge representation. In *International Workshop on Logic Programming and Non-Monotonic Reasoning*, 1999. To appear.

[31] B. Cui and D. S. Warren. Mutable terms in a tabled logic programming system. To Appear, 1999.

[32] C. Damásio. *Paraconsistent Extended Logic Programming with Constraints.* PhD thesis, Univ. Nova de Lisboa, 1996.

[33] C. Damásio, W. Nejdl, L. M. Pereira, and M. Schroeder. Model-based diagnosis preferences and strategies representation with logic meta-programming. In *Meta-logics and Logic Programming*, pages 269–311, 1995.

[34] C. Damásio and L. M. Pereira. Abduction over 3-valued extended logic programs. In *LPNMR*, pages 29–42. Springer-Verlag, 1995.

[35] C. Damásio, L. M. Pereira, and T. Swift. Coherent well-founded annotated programs. In *LPNMR*. Springer-Verlag, 1999. To Appear.

[36] S. Dawson, C. R. Ramakrishnan, I. V. Ramakrishnan, and T. Swift. Optimizing clause resolution: Beyond unification factoring. In *ILPS*, pages 194–208. MIT Press, 1995.

[37] S. Dawson, C. R. Ramakrishnan, and D. S. Warren. Practical program analysis using general purpose logic programming systems — a case study. In *ACM PLDI*, pages 117–126, May 1996.

[38] I. de Almeida Morá and J. Alferes. Modelling diagnosis systems with logic programming. Technical report, Univ. Nova de Lisboa, 1995.

[39] E. Villemont de la Clergerie. Layer sharing: an improved structure-sharing framework. In *Proc. of the 20th. Symposium on Principles of Programming Languages*, pages 345–359, 1993.

[40] E. Villemont de la Clergerie and B. Lang. LPDA: Another look at tabulation in logic programming. In *ICLP*, pages 470–488, 1994.

[41] A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. *Journal of Logic Programming*, 1999. To appear.

[42] B. Demoen and K. Sagonas. CAT: A copying approach to tabling. In *PLILP*, pages 21–35, 1998.

[43] Bart Demoen and Konstantinos Sagonas. Memory Management for Prolog with Tabling. In *Proceedings of ISMM'98: ACM SIGPLAN International Symposium on Memory Management*, pages 97–106, 1998.

[44] M. Derr, S. Morishita, and G. Phipps. Design and implementation of the Glue-Nail database system. In *Proc. of the SIGMOD 1993 Conference*, pages 147–156. ACM, 1993.

[45] Diagnostica. Available from Medicine Rules Inc: http://medicinerules.com.

[46] S. Dietrich. *Extension Tables for Recursive Query Evaluation.* PhD thesis, SUNY at Stony Brook, 1987.

[47] F. Dong and L.V. Lakshmanan. Deductive databases with incomplete information. In *JICSLP*, 1992.

[48] Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[49] C. Fan and S. Dietrich. Extension table built-ins for prolog. *Software — Practice and Experience*, 22:573–597, 1992.

[50] M. Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming*, 11:91–116, 1991.

[51] J. Freire, R. Hu, T. Swift, and D. S. Warren. Parallelizing tabled evaluation. In *7th International PLILP Symposium*, pages 115–132. Springer-Verlag, 1995.

[52] J. Freire, P. Rao, K. Sagonas, T. Swift, and D. S. Warren. XSB: A system for efficiently computing the well-founded semantics. In *LPNMR*, pages 430–440. MIT Press, 1997.

[53] J. Freire, T. Swift, and D. S. Warren. Treating I/O seriously: Resolution reconsidered for disk. In *ICLP*, pages 198–212, 1997.

[54] J. Freire, T. Swift, and D. S. Warren. Beyond depth-first: Improving tabled logic programs through alternative scheduling strategies. *Journal of Functional and Logic Programming*, 1998(3), 1998.

[55] H. Gao. *Declarative Picture Description and Interpretation in Logic*. PhD thesis, Department of Computer Science, SUNY at Stony Brook, 1993.

[56] M. Gelfond and V. Lifshitz. The stable model semantics for logic programming. In *JICSLP*, pages 1070–1080. MIT Press, 1988.

[57] M. Gelfond and V. Lifshitz. Logic programs with classical negation. In *ICLP*, pages 579–597. MIT Press, 1990.

[58] K. Govindarajan, B. Jayaraman, and S. Mantha. Preference logic programming. In *ICLP*, pages 731–746. MIT Press, 1995.

[59] S. Greco, D. Sacca, and C. Zaniolo. Dynamic programming optimization for logic queries with aggregates. In *ILPS*, pages 575–589, 1993.

[60] A. Gupta and I. Mumick. Magic sets transformations in non-recursive systems. In *PODS*, pages 354–367, 1992.

[61] J. Han. Compilation and evaluation of linear mutual recursions. *Information Sciences*, 69:157–183, 1993.

[62] J. Han and L. Liu. Efficient evaluation of multiple linear recursions. *IEEE Transactions on Software Engineering*, 17(12):1241–1252, 1991.

[63] J. Han, L. Liu, and Z. Xie. LogicBase: A deductive database system prototype. Technical report, Simon Fraser University, 1994.

[64] R. Hu. *Distributed Tabled Evaluation*. PhD thesis, SUNY at Stony Brook, 1997.

[65] G. Janssens, M. Bruynooghie, and V. Dumortier. A blueprint for an abstract machine for the abstract interpretation of (constraint) logic programs. In *ILPS*, pages 336–351, 1995.

[66] G. Janssens and K. Sagonas. On the use of tabling for abstract interpretation: An experiment with abstract equation systems. In *Tabulation in Parsing and Deduction*, 1998.

[67] B. Jayaraman and K. Moon. Implementation of subset logic progrms. Technical report, Dept. of Computer Science, SUNY Buffalo, 1994.

[68] T. Kanamori. Abstract interpretation based on alexander templates. *Journal of Logic Programming*, 15:31–54, 1993.

[69] T. Kanamori and T. Kawamura. Abstract interpretation based on oldt resolution. *Journal of Logic Programming*, 15:1–30, 1993.

[70] D. Kemp and P. Stuckey. Semantics of logic programs with aggregates. In *ILPS*, pages 387–404, 1991.

[71] D. Kemp and R. Topor. Completeness of a top-down query evaluation procedure for stratified databases. In *JICSLP*, pages 178–194, 1988.

[72] David B. Kemp, Kotagiri Ramamohanarao, and Zoltan Somogyi. Right-, left- and multi-linear rule transformations that maintain context information. In *Proceedings of the 16th Conference on Very Large Data Bases*, pages 380–391, 1990.

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

[74] M. Kifer and E.L. Lozinskii. A logic for reasoning with inconsistency. *J. Automated Reasoning*, 9(2):179–215, 1992.

[75] M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. Logic Programming*, 12(4):335–368, 1992.

[76] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27(333-354), 1983.

[77] E. Lamma, F. Riguzzi, and L. M. Pereira. Strategies in learning with extended logic programs. Technical report, Universita di Bologna, 1998.

[78] R. Larson, D. S. Warren, J. Freire, and K. Sagonas. *Syntactica, Symantica.* MIT Press, Cambridge Ma., 1995.

[79] Alexandre Lefebvre. Recursive aggregates in the eks-v1 system. Technical Report KB34, ECRC, 1991.

[80] J. Leite and L. M. Pereira. Iterated logic programming updates. In *ICLP*, pages 265–278. MIT Press, 1998.

[81] V. Lifschitz, N. McCain, T.C. Przymusinski, and R.F. Stark. Loop checking in the well-founded semantics. In V.W. Marek, A. Nerode, and M. Truszczynski, editors, *LPNMR*, pages 127–142, 1995. LNAI 928.

[82] J. Liu, L. Adams, and W. Chen. Constructive negation under the well-founded semantics. Technical report, SMU, 1996.

[83] X. Liu, C. R. Ramakrishnan, and S. Smolka. Fully local and efficient evaluation of alternating fixed points. In *TACAS 98: Tools and Algorithms for Construction and Analysis of Systems*, pages 5–19. Springer-Verlag, 1998.

[84] S. Morishita. An Extension of Van Gelder's Alternating Fixpoint to Magic Programs. *Journal of Computer and System Sciences*, 52:506–521, June 1996.

[85] I. Mumick, S. Finklestein, R. Ramakrishnan, and H. Pirahesh. Magic conditions. In *PODS*, 1990.

[86] I. Mumick and H. Pirahesh. Implementation of Magic-sets in a Relational Database System. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, pages 103–114, 1994.

[87] I. Mumick, H. Pirahesh, and R. Ramakrishnan. The magic of duplicates and aggregates. In *VLDB*, pages 264–277. VLDB End., 1990.

[88] J. Naughton, R. Ramakrishnan, Y. Sagiv, and J. Ullman. Argument reduction through factoring. In *VLDB*, pages 173–182. VLDB End., 1989.

[89] R. Ng and V.S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1993.

[90] Ulf Nilsson. Abstract interpretation: A kind of magic. In *PLILP91*, pages 299–310, 1991.

[91] G. Phipps, M. Derr, and K. Ross. Glue-Nail: A deductive database system. pages 308–317, 1991.

[92] Y. S. Ramakrishna, C. R. Ramakrishnan, I. V. Ramakrishnan, S. Smolka, T. Swift, and D. S. Warren. Efficient model checking using tabled resolution. In *Proceedings on the Conference on Automated Verification*, pages 143–154, 1997.

[93] I. V. Ramakrishnan, P. Rao, K. Sagonas, T. Swift, and D. S. Warren. Efficient access mechanisms for tabled logic programs. *Journal of Logic Programming*, 38(1):31–55, January 1999.

[94] R. Ramakrishnan, editor. *Applications of Logic Databases*. Kluwer Academic Publishers, 1995.

[95] R. Ramakrishnan, C. Beeri, and R. Krishnamurthi. Optimizing existential datalog queries. In *Proc. of the ACM Symposium on Principles of Database Systems*, pages 89–102. ACM, 1988.

[96] R. Ramakrishnan, D. Srivastava, and S. Sudarshan. Controlling the search in bottom-up evaluation. In *JICSLP*, 1992.

[97] Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan, and Praveen Seshadri. Implementation of the CORAL Deductive Database System. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, pages 167–176, 1993.

[98] R. Ramesh and W. Chen. A portable method of integrating SLG resolution into Prolog systems. In *Proc. of the Symposium on Logic Programming*, 1994.

[99] P. Rao, C.R. Ramakrishnan, and I.V. Ramakrishnan. A thread in time saves tabling time. In *JICSLP*, 1996.

[100] T. Reps. Shape analysis as a generalized path problem. In *Symposium on Partial Evaluation and Semantics-based Program Manipulation*, pages 1–11. ACM Press, 1995.

[101] R. Rocha and V. Santos Costa. Tabled-or parallelism. In *First International Conference on Tabled Logic Programming*. 1997. Approximate Title.

[102] J. Rohmer, R. Lescoeur, and J. Kerisit. The Alexander method: a technique for the processing of recursive atoms in deductive databases. *New Generation Computing*, 4:522–528, 1986.

[103] K. Ross. Modular acyclicity and tail recursion in logic programs. In *Proc. of 10th PODS*, 1991.

[104] K.A. Ross. Modular stratification and magic sets for datalog programs with negation. In *JACM*, pages 1216–1266, 1994.

[105] D. Sacca and C. Zaniolo. The generalized counting method for recursive logic queries. *Theoretical Computer Science*, 62:187–220, 1989.

[106] Y. Sagiv. Is there anything better than magic? In *Proc. of the 1990 North American Conference*, pages 235–254, 1990.

[107] K. Sagonas and T. Swift. An abstract machine for tabled execution of fixed-order stratified logic programs. *ACM TOPLAS*, 20(3):586 – 635, May 1998.

[108] K. Sagonas, T. Swift, and D. S. Warren. An abstract machine for computing the well-founded semantics. *Journal of Logic Programming*. To appear. Preliminary version appeared in Joint International Conference and Symposium on Logic Programming, 1996, pp. 274-289, MIT Press.

[109] K. Sagonas, T. Swift, and D. S. Warren. XSB as an efficient deductive database engine. In *ACM SIGMOD Conference*, 1994.

[110] K. Sagonas, T. Swift, and D. S. Warren. The limits of fixed-order computation. *Theoretical Computer Science*, 1999. To apperar. Preliminary version apeared in International Workshop on Logic and Databases, LNCS.

[111] H. Seki. On the power of Alexandrer templates. In *Proc. of 8th PODS*, pages 150–159. ACM, 1989.

[112] H. Seki and H. Itoh. A query evaluation method for stratified programs under the extended CWA. In *JICSLP*, pages 195–211, 1988.

[113] O. Shmueli, S. Tsur, and C. Zaniolo. Compilation of set terms in the logic data language (LDL). *Journal of Logic Programming*, 12:89–119, 1992.

[114] P. Stuckey and S. Sudarshan. Well-founded ordered search. In *13th conference on Foundations of Software Technology and Theoretical Computer Science*, pages 161–172, 1993.

[115] V. S. Subrahmanian. Amalgamating knowledge bases. *ACM Transactions on Database Systems*, 19(2):291–331, 1994.

[116] S. Sudarshan. *Optimizing Bottom-up Query Evaluation for Deductive Databases*. PhD thesis, University of Wisconsin, 1992.

[117] S. Sudarshan and R. Ramakrishnan. Aggregation and relevance in deductive databases. In *VLDB*, pages 501–511. VLDB End., 1991.

[118] S. Sudarshan and R. Ramakrishnan. Optimizations of bottom-up evaluation with non-ground terms. In *Proc. of the Symposium on Logic Programming*, 1993.

[119] T. Swift. Tabling for non-monotonic programming. *Annals of Mathematics and Artificial Intelligence*. To appear.

[120] T. Swift. A new formulation of tabled resolution with delay. In *EPIA 99*, 1999.

[121] T. Swift and D. S. Warren. Analysis of sequential SLG evaluation. In *Proceedings of the Symposium on Logic Programming*, pages 219–238, 1994.

[122] H. Tamaki and T. Sato. OLDT resolution with tabulation. In *ICLP*, pages 84–98. Springer-Verlag, 1986.

[123] D. Toman. Top-down beats bottom-up for constraint extensions of datalog. In *ICLP*, pages 98–115, 1995.

[124] J. Ullman. Bottom-up beats top-down for datalog. In *Proc. of 8th PODS*, pages 140–149. ACM, 1989.

[125] J. Vaghani, K. Ramamohanorao, D. Kemp, Z. Somogyi, and P. Stuckey. Design overview of the Aditi deductive database system. In *Seventh International Conference on Data Engineering*, pages 240–247, 1991.

[126] M. van Emden. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming*, 4:37–53, 1986.

[127] A. van Gelder, K.A. Ross, and J.S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. *JACM*, 38(3):620–650, 1991.

[128] L. Vieille. Recursive query processing: The power of logic. *Theoretical Computer Science*, 69:1–53, 1989.

[129] L. Vieille, P. Bayer, V. Kuchenhoff, and A. Lefebvre. EKS-V1, a short overview. In *AAAI-90 Workshop on Knowledge Base Management Systems*, 1990.

[130] A. Walker. The Syllog expert database system. Technical report, IBM, 1992.

[131] A. Walker. Backchain iteration: Towards a practical inference method that is simple enough to be proved terminating, sound, and complete. *J. Automated Reasoning*, 11(1):1–23, 1993. Originally formulated in New York University TR 34, 1981.

[132] D. S. Warren. Memoing for logic programs with applications to abstract interpretatino and partial deduction. *Communications of the ACM*, 1992.

[133] J. Wunderwald. A portable implementation of memoing evaluation. In *LOPSTR 95*, 1995.

[134] U. Zukowski, S. Brass, and B. Freitag. Improving the alternating fixpoint: The transformation approach. In *LPNMR*, pages 40–59. Springer-Verlag, 1997.

[135] U. Zukowski and B. Freitag. Adding flexibility to query evaluation for modularly stratified databases. In *JICSLP*, pages 304–319, 1996.

[136] U. Zukowski and B. Freitag. Well-founded semantics by transformation: The non-ground case. In *ICLP*, 1999. To appear.