Answer Set Programming

João Leite

Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, Portugal
jleite@fct.unl.pt

November 2017

Outline

Outline

Outline

Disjunctive Logic Programs: Syntax

Definition (Disjunctive Rule)

A disjunctive rule, r, is an ordered pair of the form

$$A_1 \ ; \ldots ; A_m \leftarrow A_{m+1}, \ldots, A_n, \mathit{not}\ A_{n+1}, \ldots, \mathit{not}\ A_o,$$

where $o \geq n \geq m \geq 0$, and each A_i $(0 \leq i \leq o)$ is an atom.

Definition (Disjunctive Logic Program)

A disjunctive logic program is a finite set of disjunctive rules.

Notation

$$\begin{array}{rcl} \mathit{head}(r) & = & \{A_1, \ldots, A_m\} \\ \mathit{body}(r) & = & \{A_{m+1}, \ldots, A_n, \mathit{not}\ A_{n+1}, \ldots, \mathit{not}\ A_o\} \\ \mathit{body}^+(r) & = & \{A_{m+1}, \ldots, A_n\} \\ \mathit{body}^-(r) & = & \{A_{n+1}, \ldots, A_o\} \end{array}$$

Disjunctive Logic Programs: Semantics

Definition (Positive Disjunctive Logic Programs)

A program is called positive if $body^-(r) = \emptyset$ for all its rules.

Definition (Closure)

A set X of atoms is closed under a positive program Π iff for any $r \in \Pi$, $head(r) \cap X \neq \emptyset$ whenever $body^+(r) \subseteq X$.

- X corresponds to a model of Π (seen as a formula).

Definition ($\min_{\subseteq}(\Pi)$)

The set of all $\subseteq$-minimal sets of atoms being closed under a positive program Π is denoted by $\min_{\subseteq}(\Pi)$.

- $\min_{\subseteq}(\Pi)$ corresponds to the $\subseteq$-minimal models of Π (seen as a formula).

Disjunctive Logic Programs: Semantics

Definition (Reduct of a Disjunctive Logic Program)

The reduct, Π^X, of a disjunctive program Π relative to a set X of atoms is defined by

$$\Pi^X = \{head(r) \leftarrow body^+(r) \mid r \in \Pi \text{ and } body^-(r) \cap X = \emptyset\}.$$

Definition (Answer Set of a Disjunctive Logic Program)

A set X of atoms is an answer set of a disjunctive program Π if $X \in \min_{\subseteq}(\Pi^X)$.

Positive Disjunctive Logic Programs: Example

Example

$$\Pi = \left\{ \begin{array}{lcl} a & \leftarrow & \\ b\,;c & \leftarrow & a \end{array} \right\}$$

- The sets $\{a, b\}$, $\{a, c\}$, and $\{a, b, c\}$ are closed under Π.
- We have $\min_{\subseteq}(\Pi) = \{\ \{a, b\},\ \{a, c\}\ \}$.

3-colorability of graphs revisited

Problem

Problem instance *A graph* (V, E).

Problem class *Assign each vertex in* V *one of* 3 *colors such that no two vertexes in* V *connected by an edge in* E *have the same color.*

Solution

C(I)	*vertex(1)* $\leftarrow$ *vertex(2)* $\leftarrow$ *vertex(3)* $\leftarrow$ *edge(1,2)* $\leftarrow$ *edge(2,3)* $\leftarrow$ *edge(3,1)* $\leftarrow$
C(P)	*colored(V,r); colored(V,b); colored(V,g)* $\leftarrow$ *vertex(V)* $\leftarrow$ *edge(V,U), colored(V,C), colored(U,C)*
AS's	{ *colored(1,r), colored(2,b), colored(3,g), . . .* }

Disjunctive Logic Programs: Examples

Example

- $\Pi_1 = \{a\,;b\,;c \leftarrow\}$ has answer sets $\{a\}$, $\{b\}$, and $\{c\}$.
- $\Pi_2 = \{a\,;b\,;c \leftarrow\,,\ \leftarrow a\}$ has answer sets $\{b\}$ and $\{c\}$.
- $\Pi_3 = \{a\,;b\,;c \leftarrow\,,\ \leftarrow a\,,\ b \leftarrow c\,,\ c \leftarrow b\}$ has answer set $\{b, c\}$.
- $\Pi_4 = \{a\,;b \leftarrow c\,,\ b \leftarrow \mathit{not}\ a, \mathit{not}\ c\,,\ a\,;c \leftarrow \mathit{not}\ b\}$ has answer sets $\{a\}$ and $\{b\}$.

Some properties

Property

A disjunctive logic program may have zero, one, or multiple stable models

Property

If X is a stable model of a disjunctive logic program Π, then X is a model of Π (seen as a formula)

Property

If X and Y are stable models of a disjunctive logic program Π, then $X \not\subset Y$

Property

If $A \in X$ for some stable model X of a disjunctive logic program Π, then there is a rule $r \in \Pi$ such that $body^+(r) \subseteq X$, $body^-(r) \cap X = \emptyset$, and $head(r) \cap X = \{A\}$

Disjunctive Logic Programs: Example with variables

Example

$$\Pi = \left\{ \begin{array}{lll} a(1,2) & \leftarrow & \\ b(X)\,;c(Y) & \leftarrow & a(X,Y), not\ c(Y) \end{array} \right\}$$

$$ground(\Pi) = \left\{ \begin{array}{lll} a(1,2) & \leftarrow & \\ b(1)\,;c(1) & \leftarrow & a(1,1), not\ c(1) \\ b(1)\,;c(2) & \leftarrow & a(1,2), not\ c(2) \\ b(2)\,;c(1) & \leftarrow & a(2,1), not\ c(1) \\ b(2)\,;c(2) & \leftarrow & a(2,2), not\ c(2) \end{array} \right\}$$

For every answer set X of Π, we have

- $a(1,2) \in X$ and
- $\{a(1,1), a(2,1), a(2,2)\} \cap X = \emptyset$.

Disjunctive Logic Programs: Example with variables

Example

$$ground(\Pi)^X = \left\{ \begin{array}{lcl} a(1,2) & \leftarrow & \\ b(1)\,;c(1) & \leftarrow & a(1,1) \\ b(1)\,;c(2) & \leftarrow & a(1,2) \\ b(2)\,;c(1) & \leftarrow & a(2,1) \\ b(2)\,;c(2) & \leftarrow & a(2,2) \end{array} \right\}$$

- Consider $X = \{a(1,2),\ b(1)\}$.
- We get $\min_{\subseteq}(ground(\Pi)^X) = \{\ \{a(1,2),\ b(1)\},\ \{a(1,2),\ c(2)\}\ \}$.
- X is an answer set of Π because $X \in \min_{\subseteq}(ground(\Pi)^X)$.

Disjunctive Logic Programs: Example with variables

Example

$$ground(\Pi)^X = \left\{ \begin{array}{lcl} a(1,2) & \leftarrow & \\ b(1)\,;c(1) & \leftarrow & a(1,1) \\ & & \\ b(2)\,;c(1) & \leftarrow & a(2,1) \\ & & \end{array} \right\}$$

- Consider $X = \{a(1,2),\ c(2)\}$.
- We get $\min_{\subseteq}(ground(\Pi)^X) = \{\ \{a(1,2)\}\ \}$.
- X is no answer set of Π because $X \notin \min_{\subseteq}(ground(\Pi)^X)$.

Outline

Nested Logic Programs: Syntax

Definition (Formulas)

Formulas are formed from propositional atoms, $\top$ and $\bot$, using negation-as-failure (*not*), conjunction (,), and disjunction (;).

Definition (Nested Rules)

A nested rule, r, is an ordered pair of the form

$$F \leftarrow G$$

where F and G are formulas.

Definition (Nested Logic Program)

A nested program is a finite set of rules.

Nested Logic Programs: Semantics

Notation

$head(r) = F$ and $body(r) = G$.

Definition (Satisfaction relation)

The satisfaction relation $X \models F$ between a set of atoms and a formula F is defined recursively as follows:

- $X \models F$ if $F \in X$ for an atom F,
- $X \models \top$,
- $X \not\models \bot$,
- $X \models (F, G)$ if $X \models F$ and $X \models G$,
- $X \models (F; G)$ if $X \models F$ or $X \models G$,
- $X \models not\ F$ if $X \not\models F$.

A set X of atoms satisfies a nested program Π, written $X \models \Pi$, iff for any $r \in \Pi$, $X \models head(r)$ whenever $X \models body(r)$.

Nested Logic Programs: Semantics

Definition ($\min_{\subseteq}(\Pi)$)

The set of all $\subseteq$-minimal sets of atoms satisfying program Π is denoted by $\min_{\subseteq}(\Pi)$.

Definition (Reduct of a Formula)

The reduct, F^X, of a formula F relative to a set X of atoms is defined recursively as follows:

- $F^X = F$ if F is an atom or $\top$ or $\bot$,
- $(F, G)^X = (F^X, G^X)$,
- $(F; G)^X = (F^X; G^X)$,
- $(not\ F)^X = \begin{cases} \bot & \text{if } X \models F \\ \top & \text{otherwise} \end{cases}$

Nested Logic Programs: Semantics

Definition (Reduct of a Nested Logic Program)

The reduct, Π^X, of a nested program Π relative to a set X of atoms is defined by

$$\Pi^X = \{head(r)^X \leftarrow body(r)^X \mid r \in \Pi\}.$$

Definition (Answer Set of a Nested Logic Program)

A set X of atoms is an answer set of a nested program Π iff $X \in \min_{\subseteq}(\Pi^X)$.

Nested Logic Programs: Examples

Example

- $\Pi_1 = \{(p\ ;\ not\ p) \leftarrow \top\}$
  - For $X = \emptyset$, we get
    - $\Pi_1^{\emptyset} = \{(p\ ; \top) \leftarrow \top\}$
    - $\min_{\subseteq}(\Pi_1^{\emptyset}) = \{\emptyset\}$. ✔
  - For $X = \{p\}$, we get
    - $\Pi_1^{\{p\}} = \{(p\ ; \bot) \leftarrow \top\}$
    - $\min_{\subseteq}(\Pi_1^{\{p\}}) = \{\{p\}\}$. ✔
- $\Pi_2 = \{p \leftarrow not\ not\ p\}$
  - For $X = \emptyset$, we get $\Pi_2^{\emptyset} = \{p \leftarrow \bot\}$ and $\min_{\subseteq}(\Pi_2^{\emptyset}) = \{\emptyset\}$. ✔
  - For $X = \{p\}$, we get $\Pi_2^{\{p\}} = \{p \leftarrow \top\}$ and $\min_{\subseteq}(\Pi_2^{\{p\}}) = \{\{p\}\}$. ✔

- In general (Intuitionistic Logics HT (Heyting, 1930) and G3 (Gödel, 1932))
  - $F \leftarrow G,\ not\ not\ H$ is equivalent to $F\ ;\ not\ H \leftarrow G$
  - $F\ ;\ not\ not\ G \leftarrow H$ is equivalent to $F \leftarrow H, not\ G$
  - $not\ not\ not\ F$ is equivalent to $not\ F$

Hamiltonian Paths: Generator Revisited

Example

Normal logic programs

$$inPath(X, Y) \leftarrow arc(X, Y),\ not\ outPath(X, Y)$$
$$outPath(X, Y) \leftarrow arc(X, Y),\ not\ inPath(X, Y)$$

Disjunctive logic programs

$$inPath(X, Y)\ ;\ outPath(X, Y) \leftarrow arc(X, Y)$$

Nested logic programs

$$inPath(X, Y)\ ;\ not\ inPath(X, Y) \leftarrow arc(X, Y)$$

Outline

Propositional Theories: Syntax

Definition (Formulas)

Formulas are formed from atoms and $\bot$ using conjunction ($\wedge$), disjunction ($\vee$), and implication ($\rightarrow$).

Notation

$$\top = (\bot \rightarrow \bot)$$
$$\sim F = (F \rightarrow \bot) \qquad \text{(or: } \textit{not } F\text{)}$$

Definition (Propositional Theory)

A propositional theory is a finite set of formulas.

Propositional Theories: Semantics

Definition (Satisfaction relation)

The satisfaction relation $X \models F$ between a set X of atoms and a (set of) formula(s) F is defined as in propositional logic.

Definition (Reduct of a formula)

The reduct, F^X, of a formula F relative to a set X of atoms is defined recursively as follows:

- $F^X = \bot$ if $X \not\models F$
- $F^X = F$ if $F \in X$
- $F^X = (G^X \circ H^X)$ if $X \models F$ and $F = (G \circ H)$ for $\circ \in \{\wedge, \vee, \rightarrow\}$

➥ If $F = \sim G = (G \rightarrow \bot)$,
then $F^X = (\bot \rightarrow \bot) = \top$, if $X \not\models G$, and $F^X = \bot$, otherwise.

Propositional Theories: Semantics

Definition (Reduct of a Propositional Theory)

The reduct, $\mathcal{F}^X$, of a propositional theory $\mathcal{F}$ relative to a set X of atoms is defined as

$$\mathcal{F}^X = \{F^X \mid F \in \mathcal{F}\}.$$

Definition (Satisfaction of a Propositional Theory)

A set X of atoms satisfies a propositional theory $\mathcal{F}$, written $X \models \mathcal{F}$, iff $X \models F$ for each $F \in \mathcal{F}$.

Propositional Theories: Semantics

Definition ($\min_{\subseteq}(\mathcal{F})$)

The set of all $\subseteq$-minimal sets of atoms satisfying a propositional theory $\mathcal{F}$ is denoted by $\min_{\subseteq}(\mathcal{F})$.

Definition (Answer Set of a Propositional Theory)

A set X of atoms is an answer set of a propositional theory $\mathcal{F}$ if $X \in \min_{\subseteq}(\mathcal{F}^X)$.

Proposition

If X is an answer set of $\mathcal{F}$, then $X \models \mathcal{F}$.

- *In general, this does not imply $X \in \min_{\subseteq}(\mathcal{F})$!*

Propositional Theories: Two examples

Example

- $\mathcal{F}_1 = \{p \vee (p \rightarrow (q \wedge r))\}$
  - For $X = \{p, q, r\}$, we get
    $\mathcal{F}_1^{\{p,q,r\}} = \{p \vee (p \rightarrow (q \wedge r))\}$ and $\min_{\subseteq}(\mathcal{F}_1^{\{p,q,r\}}) = \{\emptyset\}$. ✘
  - For $X = \emptyset$, we get
    $\mathcal{F}_1^{\emptyset} = \{\bot \vee (\bot \rightarrow \bot)\}$ and $\min_{\subseteq}(\mathcal{F}_1^{\emptyset}) = \{\emptyset\}$. ✔

- $\mathcal{F}_2 = \{p \vee (\sim p \rightarrow (q \wedge r))\}$
  - For $X = \emptyset$, we get
    $\mathcal{F}_2^{\emptyset} = \{\bot\}$ and $\min_{\subseteq}(\mathcal{F}_2^{\emptyset}) = \emptyset$. ✘
  - For $X = \{p\}$, we get
    $\mathcal{F}_2^{\{p\}} = \{p \vee (\bot \rightarrow \bot)\}$ and $\min_{\subseteq}(\mathcal{F}_2^{\{p\}}) = \{\emptyset\}$. ✘
  - For $X = \{q, r\}$, we get
    $\mathcal{F}_2^{\{q,r\}} = \{\bot \vee (\top \rightarrow (q \wedge r))\}$ and $\min_{\subseteq}(\mathcal{F}_2^{\{q,r\}}) = \{\{q, r\}\}$. ✔

Propositional Theories: Relationship with Logic Programs

Definition (Translation of a nested rule)

The translation, $\tau[(F \leftarrow G)]$, of a (nested) rule $(F \leftarrow G)$ is defined recursively as follows:

- $\tau[(F \leftarrow G)] = (\tau[G] \rightarrow \tau[F])$,
- $\tau[\bot] = \bot$,
- $\tau[\top] = \top$,
- $\tau[F] = F$ if F is an atom,
- $\tau[not\ F] = {\sim}\tau[F]$,
- $\tau[(F, G)] = (\tau[F] \wedge \tau[G])$,
- $\tau[(F; G)] = (\tau[F] \vee \tau[G])$.

Definition (Translation of a nested logic program)

The translation of a logic program Π is $\tau[\Pi] = \{\tau[r] \mid r \in \Pi\}$.

Propositional Theories: Relationship with Logic Programs

Theorem (Embedding of nested logic programs)

Given a logic program Π *and a set* X *of atoms,* X *is an answer set of* Π *iff* X *is an answer set of* $\tau[\Pi]$.

Example

- The normal logic program $\Pi = \{p \leftarrow not\ q,\ q \leftarrow not\ p\}$ corresponds to $\tau[\Pi] = \{\sim q \rightarrow p,\ \sim p \rightarrow q\}$.
  - Answer sets: $\{p\}$ and $\{q\}$
- The disjunctive logic program $\Pi = \{p\ ;\ q \leftarrow\}$ corresponds to $\tau[\Pi] = \{\top \rightarrow p \vee q\}$.
  - Answer sets: $\{p\}$ and $\{q\}$
- The nested logic program $\Pi = \{p \leftarrow not\ not\ p\}$ corresponds to $\tau[\Pi] = \{\sim\sim p \rightarrow p\}$.
  - ➥ Answer sets: $\emptyset$ and $\{p\}$

Outline

Computational Complexity

Computational Complexity

Let A be an atom and X be a set of atoms.

- For a positive normal logic program Π:
  - Deciding whether X is the answer set of Π is **P**-complete.
  - Deciding whether A is in the answer set of Π is **P**-complete.
- For a normal logic program Π:
  - Deciding whether X is an answer set of Π is **P**-complete.
  - Deciding whether A is in an answer set of Π is **NP**-complete.

Computational Complexity

Computational Complexity

- For a positive disjunctive logic program Π:
  - Deciding whether X is an answer set of Π is **co-NP**-complete.
  - Deciding whether A is in an answer set of Π is $\mathbf{NP^{NP}}$-complete.
- For a disjunctive logic program Π:
  - Deciding whether X is an answer set of Π is **co-NP**-complete.
  - Deciding whether A is in an answer set of Π is $\mathbf{NP^{NP}}$-complete.
- For a nested logic program Π:
  - Deciding whether X is an answer set of Π is **co-NP**-complete.
  - Deciding whether A is in an answer set of Π is $\mathbf{NP^{NP}}$-complete.
- For a propositional theory $\mathcal{F}$:
  - Deciding whether X is an answer set of $\mathcal{F}$ is **co-NP**-complete.
  - Deciding whether A is in an answer set of $\mathcal{F}$ is $\mathbf{NP^{NP}}$-complete.

Outline

Bibliography I


C. Baral.
Knowledge Representation, Reasoning and Declarative Problem Solving.
Cambridge University Press, 2003.


S. Brass and J. Dix.
Semantics of (disjunctive) logic programs based on partial evaluation.
Journal of Logic Programming, 40(1):1–46, 1999.


P. Cabalar and P. Ferraris.
Propositional theories are strongly equivalent to logic programs.
Theory and Practice of Logic Programming, 7(6):745–759, 2007.


K. Clark.
Negation as failure.
In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

Bibliography II


E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov.
Complexity and expressive power of logic programming.
In *Proceedings of the Twelfth Annual IEEE Conference on Computational Complexity (CCC'97)*, pages 82–101. IEEE Computer Society Press, 1997.


E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov.
Complexity and expressive power of logic programming.
ACM Computing Surveys, 33(3):374–425, 2001.


T. Eiter and G. Gottlob.
On the computational cost of disjunctive logic programming: Propositional case.
Annals of Mathematics and Artificial Intelligence, 15(3-4):289–323, 1995.

Bibliography III


T. Eiter, M. Fink, H. Tompits, and S. Woltran.
Simplifying logic programs under uniform and strong equivalence.
In V. Lifschitz and I. Niemelä, editors, *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'04)*, volume 2923 of *Lecture Notes in Artificial Intelligence*, pages 87–99. Springer-Verlag, 2004.


Wolfgang Faber, Nicola Leone, and Gerald Pfeifer.
Recursive aggregates in disjunctive logic programs: Semantics and complexity.
In José Júlio Alferes and João Alexandre Leite, editors, *JELIA*, volume 3229 of *Lecture Notes in Computer Science*, pages 200–212. Springer, 2004.


P. Ferraris and V. Lifschitz.
Mathematical foundations of answer set programming.
In S. Artëmov, H. Barringer, A. d'Avila Garcez, L. Lamb, and J. Woods, editors, *We Will Show Them! Essays in Honour of Dov Gabbay, Volume One*, pages 615–664. College Publications, 2005.

Bibliography IV


Paolo Ferraris and Vladimir Lifschitz.
Mathematical foundations of answer set programming.
In Sergei N. Artëmov, Howard Barringer, Artur S. d'Avila Garcez, Luís C. Lamb, and John Woods, editors, *We Will Show Them! (1)*, pages 615–664. College Publications, 2005.


Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz.
A new perspective on stable models.
In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 372–379, 2007.


Paolo Ferraris.
Answer sets for propositional theories.
In Chitta Baral, Gianluigi Greco, Nicola Leone, and Giorgio Terracina, editors, *LPNMR*, volume 3662 of *Lecture Notes in Computer Science*, pages 119–131. Springer, 2005.

Bibliography V


H Gaifman and E. Shapiro.
Fully abstract compositional semantics for logic programs.
In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Programming Languages (POPL'89)*, pages 134–142, 1989.


M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele.
A user's guide to `gringo`, `clasp`, `clingo`, and `iclingo`.
Available at `http://potassco.sourceforge.net`.


Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf.
The well-founded semantics for general logic programs.
J. ACM, 38(3):620–650, 1991.


M. Gelfond and V. Lifschitz.
The stable model semantics for logic programming.
In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, pages 1070–1080. The MIT Press, 1988.

Bibliography VI


M. Gelfond and V. Lifschitz.
Logic programs with classical negation.
In *Proceedings of the International Conference on Logic Programming*, pages 579–597, 1990.


M. Gelfond and V. Lifschitz.
Classical negation in logic programs and disjunctive databases.
New Generation Computing, 9:365–385, 1991.


H. Kautz and B. Selman.
Planning as satisfiability.
In B. Neumann, editor, *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, pages 359–363. John Wiley & sons, 1992.


R. Kowalski.
Logic for data description.
In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 77–103. Plenum Press, 1978.

Bibliography VII


Joohyung Lee, Vladimir Lifschitz, and Ravi Palla.
A reductive semantics for counting and choice in answer set programming.
In Dieter Fox and Carla P. Gomes, editors, *AAAI*, pages 472–479. AAAI Press, 2008.


Joohyung Lee.
A model-theoretic counterpart of loop formulas.
In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 503–508. Professional Book Center, 2005.


N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello.
The DLV system for knowledge representation and reasoning.
ACM Transactions on Computational Logic, 7(3):499–562, 2006.

Bibliography VIII


V. Lifschitz and H. Turner.
Splitting a logic program.
In *Proceedings of the Eleventh International Conference on Logic Programming*, pages 23–37. MIT Press, 1994.


V. Lifschitz, L. Tang, and H. Turner.
Nested expressions in logic programs.
Annals of Mathematics and Artificial Intelligence, 25(3-4):369–389, 1999.


V. Lifschitz, D. Pearce, and A. Valverde.
Strongly equivalent logic programs.
ACM Transactions on Computational Logic, 2(4):526–541, 2001.


Vladimir Lifschitz, David Pearce, and Agustín Valverde.
Strongly equivalent logic programs.
ACM Trans. Comput. Log., 2(4):526–541, 2001.

Bibliography IX


V. Lifschitz.
Answer set programming and plan generation.
Artificial Intelligence, 138(1-2):39–54, 2002.


Vladimir Lifschitz.
Twelve definitions of a stable model.
In Maria Garcia de la Banda and Enrico Pontelli, editors, *ICLP*, volume 5366 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2008.


Fangzhen Lin and Yuting Zhao.
Assat: computing answer sets of a logic program by sat solvers.
Artif. Intell., 157(1-2):115–137, 2004.


Fangzhen Lin and Yi Zhou.
From answer set logic programming to circumscription via logic of gk.
In Manuela M. Veloso, editor, *IJCAI*, pages 441–446, 2007.

Bibliography X


J. Lloyd.
Foundations of Logic Programming.
Symbolic Computation. Springer-Verlag, 2nd edition, 1987.


J. McCarthy and P. J. Hayes.
Some philosophical problems from the standpoint of artificial intelligence.
pages 26–45, 1987.


John McCarthy.
Circumscription - a form of non-monotonic reasoning.
Artif. Intell., 13(1-2):27–39, 1980.


Drew V. McDermott and Jon Doyle.
Non-monotonic logic i.
Artif. Intell., 13(1-2):41–72, 1980.


Drew V. McDermott.
Nonmonotonic logic ii: Nonmonotonic modal theories.
J. ACM, 29(1):33–57, 1982.

Bibliography XI


M. Nogueira, M. Balduccini, M. Gelfond, R. Watson, and M. Barry.
An A-prolog decision support system for the space shuttle.
In I. Ramakrishnan, editor, *Proceedings of the Third International Symposium on Practical Aspects of Declarative Languages (PADL'01)*, volume 1990 of *Lecture Notes in Computer Science*, pages 169–183. Springer-Verlag, 2001.


E. Oikarinen and T. Janhunen.
Modular equivalence for normal logic programs.
In G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, editors, *Proceedings of the Seventeenth European Conference on Artificial Intelligence (ECAI'06)*, pages 412–416. IOS Press, 2006.


M. Osorio, J. Navarro, and J. Arrazola.
Equivalence in answer set programming.
In A. Pettorossi, editor, *Proceedings of the Eleventh International Workshop on Logic Based Program Synthesis and Transformation (LOPSTR'01)*, volume 2372 of *Lecture Notes in Computer Science*, pages 57–75. Springer-Verlag, 2001.

Bibliography XII


David Pearce, Hans Tompits, and Stefan Woltran.
Encodings for equilibrium logic and logic programs with nested expressions.
In Pavel Brazdil and Alípio Jorge, editors, *EPIA*, volume 2258 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2001.


David Pearce.
A new logical characterisation of stable models and answer sets.
In Jürgen Dix, Luís Moniz Pereira, and Teodor C. Przymusinski, editors, *NMELP*, volume 1216 of *Lecture Notes in Computer Science*, pages 57–70. Springer, 1996.


David Pearce.
Equilibrium logic.
Ann. Math. Artif. Intell., 47(1-2):3–41, 2006.


Raymond Reiter.
A logic for default reasoning.
Artif. Intell., 13(1-2):81–132, 1980.

Bibliography XIII


Domenico Saccà and Carlo Zaniolo.
Stable models and non-determinism in logic programs with negation.
In *PODS*, pages 205–217. ACM Press, 1990.


P. Simons, I. Niemelä, and T. Soininen.
Extending and implementing the stable model semantics.
Artificial Intelligence, 138(1-2):181–234, 2002.


T. Syrjänen.
Lparse 1.0 user's manual.
http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz.


H. Turner.
Strong equivalence made easy: nested expressions and weight constraints.
Theory and Practice of Logic Programming, 3(4-5):609–622, 2003.

Bibliography XIV


Maarten H. van Emden and Robert A. Kowalski.
The semantics of predicate logic as a programming language.
J. ACM, 23(4):733–742, 1976.