Knowledge representation languages — a programmer's interface to satisfiability

Mirosław Truszczyński

Department of Computer Science
University of Kentucky

September 6, 2006

Modeling and search — two sides of the same coin

McCarthy and Hayes on AI, 1969

- [...] intelligence has two parts, which we shall call the epistemological and the heuristic.
  The epistemological part is the representation of the world in such a form that the solution of problems follows from the facts expressed in the representation.
  The heuristic part is the mechanism that on the basis of the information solves the problem and decides what to do.

- Epistemological part $\longrightarrow$ modeling (knowledge representation)
- Heuristic $\longrightarrow$ search

More examples ...

Declarative programming

- Problem specification language $\longrightarrow$ modeling
- Automated reasoning $\longrightarrow$ search (for proofs)

Databases

- Query specification language $\longrightarrow$ modeling
- Query execution $\longrightarrow$ search (for records/answers)

Point missed by SAT (in my view)

SAT focused on search

- Many hard problems reduce to finding models of CNF theories
- Can be solved by SAT solvers — programs that search for models of CNF theories
- Search for models — the main focus SAT solver developers
- But how to build reductions? How to generate inputs to SAT solvers?
- It is fundamental to provide support for these tasks
- KR can help

KR

The goal

- To design and study languages to capture knowledge about environments, their entities and their behaviors

Early proposal (McCarthy)

- Use classical logic — it is “descriptively universal”
- Challenges
  - Qualification problem
  - Frame problem
  - Defaults, rules with exceptions, normative statements, conditionals
  - Definitions and, especially, *inductive* definitions
  - Negative information

Non-monotonic logics

Proposed in response to challenges of KR

- Language of logic with non-classical semantics
- Model preference
  - circumscription
    McCarthy 1977
- Fixpoint conditions defining belief sets
  - default logic
    Reiter 1980
  - autoepistemic logic
    Moore 1984
  - logic programming with stable-model semantics (more managabe fragment of default logic)
    Gelfond-Lifschitz, 1988
  - ID-logic
    Denecker 1998, 2000; Denecker-Ternovska 2004

Logic programming with stable-model semantics

Syntax — that of standard logic programming

- Programs — collections of clauses (in full language of logic)
- $A \leftarrow B_1, \ldots, B_k, \mathbf{not}(C_1), \ldots, \mathbf{not}(C_m)$
  - "if all A_i are computed and none of B_i can be computed, then compute A"

Semantics — stable models

- Certain Herbrand models of a program P
- Alternatively — certain models of $\mathit{ground}(P)$
- A program is viewed as a definition of the collection of its stable models
- departure from traditional logic-programming perspective
- Answer-set programming (ASP)

Graph-coloring example

Description of input: vertices, edges, colors

$vtx(1).\ \ vtx(2).\ \ \ldots$
$edge(1,4).\ \ edge(3,2).\ \ \ldots$
$color(r).\ \ color(g).\ \ color(b).\ \ \ldots$

Problem specification: assignment of colors

$clrd(X, C) \leftarrow vtx(X), color(C), \textbf{not}(othercolor(X, C)).$
$othercolor(X, C) \leftarrow clrd(X, D), D \neq C.$

Problem specification: imposing colorabilty condition

$\leftarrow edge(X, Y), color(C), clrd(X, C), clrd(Y, C).$

Graph-coloring example

Correctness

- A set M of ground atoms is a stable model of the 3-coloring program iff
  - M contains all facts of the program
  - for every vertex v, there is exactly one color c such that $clrd(v,c)$ is in M, and
    for every $d \neq c$, $othercolor(v,c) \in M$
  - for every edge (u,v), if $clrd(u,c) \in M$, then $clrd(v,c) \notin M$
- 1-to-1 correspondence with proper 3-colorings of G
- Given a stable model, the corresponding coloring can be reconstructed easily and quickly

Generalizing ...

Answer-Set Programming (ASP)

- Code computational problems as logic programs so that stable models correspond to solutions
  - disallowing function symbols in the language guarantees finiteness of ground programs and their stable models
- Ground the program — bridge from modeling to search
- Search — find stable models of the ground program
  - search problem similar to SAT and with the same complexity
  - place for SAT solvers and SAT techniques
- Output — recover solutions from stable models

Expressive power of ASP

Uniform encoding of a search problem Π

- Problem specification

$$\mathit{clrd}(X,C) \leftarrow \mathit{vtx}(X), \mathit{color}(C), \textbf{not}(\mathit{othercolor}(X,C)).$$
$$\mathit{othercolor}(X,C) \leftarrow \mathit{clrd}(X,D), D \neq C.$$
$$\leftarrow \mathit{edge}(X,Y), \mathit{color}(C), \mathit{clrd}(X,C), \mathit{clrd}(Y,C).$$

- Description of input

$$\mathit{vtx}(1).\ \ \mathit{vtx}(2).\ \ \ldots$$
$$\mathit{edge}(1,4).\ \ \mathit{edge}(3,2).\ \ \ldots$$
$$\mathit{color}(r).\ \ \mathit{color}(g).\ \ \mathit{color}(b).\ \ \ldots$$

Expressive power

- the class of problems that can be represented in this way by finite programs

Expressive power of ASP

Finite programs w/out function symbols capture precisely the class NPMV

- NPMV — the class of all search problems computed by polynomial-time non-deterministic transducers

transducers: non-deterministic Turing Machine-like devices that compute partial multivalued functions from strings to strings, that is, search problems

Great approach ...

Advatages

- Stable-model semantics addresses several of KR challenges
- Gives rise to an effective KR system

  reasoning about action, planning, ... ; Gelfond-Leone 2001, Baral 2002
- Comes with modeling language
- Comes with computational support

  lparse/smodels; Niemelä-Simons-Syrjänen

  dlv; Leone-Eiter-Faber-Pfeifer, ...
- Can it be used as an interface to SAT solvers?

  not directly but essentially yes; more on this later ...

But ...

Stable-model semantics also a problem

- Coding — stable models not a household name
- Computing — methods to compute stable models recieved relatively little attention

Alternatives?

- Stay closer to classical logic

Logic of propositional schemata, PS; East-MT, 2000

Language of predicate logic, essentially

- Sets of constant, variable and predicate symbols (no function symbols)
- Equality symbol "="
- Boolean connectives:
  - $\wedge$ we will write: ,
  - $\vee$ we will write: |
  - $\rightarrow$
- square brackets "[" and "]" for existential quantification
- Terms: constant and variable symbols
- Atoms and ground atoms
- Eq-atom: $p(t)[X, Y, \ldots]$ (stands for: $\exists X \exists Y \ldots p(t)$)

Logic PS

Formulas and theories

- PS-clauses
  - $A_1, \ldots, A_m \rightarrow B_1 \mid \ldots \mid B_n$.
  - A_i — atoms
  - B_i — atoms or eq-atoms
  - implicitly universally quantified
  - implication notation better aligned with typical natural language specs of constraints
- PS-theories
  - finite sets of PS-clauses with at least one constant

Example — graph-coloring problem

Every vertex gets at least one color

- $vtx(X) \rightarrow clrd(X, C)[C].$

For every edge, its vertices are colored differently

- $edge(X, Y), clrd(X, C), clrd(Y, C) \rightarrow .$

Logic PS — semantics

Models of a PS-theory T

- Herbrand models with $HU(T)$ as the domain
- Equivalently, subsets of $HB(T)$
- Or, truth assignments to atoms from $HB(T)$

Logic PS — semantics

$ground(T)$

- $a, b, \ldots$ — all constants in a PS-theory T
- R — a PS-clause in T
- $ground(R)$ — set of propositional clauses obtained by:
  - replacing R by all its ground instances (substitute free variables with constants)
  - eliminating existential quantification — replacing eq-atoms with disjunctions
    $p(t)[X] \quad \longrightarrow \quad p(t_{X/a}) \mid p(t_{X/b}) \mid \ldots$
- $ground(T) = \{ground(R) \colon R \in T\}$

Example

$vtx(X) \rightarrow clrd(X, C)[C].$

- Assume: vertices 1, 2, 3; colors a, b
- Step 1:
  $vtx(2) \rightarrow clrd(2, C)[C]$ (one of the instantiations)
- Step 2:
  $vtx(2) \rightarrow clrd(2, a) \mid clrd(2, b) \mid clrd(2, 1) \mid clrd(2, 2) \mid clrd(2, 3)$

Logic PS — semantics

Propositional characterization of models of PS-theories

- Models of $ground(T)$ = models of T

Logic PS in specyfing problems and their instances

Program-data pairs

- Problem specifications should be independent of instances
- Program-data pair: (P, D)
  - P — program: PS-theory to specify a problem
  - D — data: set of ground atoms to describe a problem instance
- Data predicates — those that appear in D
- Program predicates — all other predicates in P

Graph-coloring problem

Program P

- $vtx(X) \to clrd(X, C)[C].$
- $clrd(X, C), clrd(X, D) \to C = D.$
- $edge(X, Y), clrd(X, C), clrd(Y, C) \to .$
- $clrd(X, C) \to vtx(X).$ (typing)
- $clrd(X, C) \to color(C).$ (typing)

Data (instance) D

- $vtx(1), vtx(2), vtx(3), vtx(4).$
- $edge(1, 2), edge(1, 3), edge(2, 4), edge(4, 3).$
- $color(r), color(b), color(g)$

Semantics of program-data pairs

$CWA(D)$

- Intended meaning of D — a complete specification of a data instance
- For every data-predicate ground atom not listed explicitly in D, assume its negation
  - no $vtx(b)$ in D
  - b is not a vertex
  - $\neg vtx(b)$ holds
- $CWA(D) = D \cup \{\neg p(t) \colon p$ – data predicate, t – ground, $p(t) \notin D\}$

Semantics of program-data pairs

Models of (P, D)

- Meaning of a program-data pair (P, D) — PS-theory $P \cup CWA(D)$
- Models of a program-data pair (P, D) — models of $P \cup CWA(D)$

Graph-coloring example

Correctness of the encoding

- G — a graph
- P — coloring program as described above (with typing)
- D — a set of ground atoms specifying G and the colors
- Colorings of G are in one-to-one correspondence to models of (P, D)

Expressive power of logic PS

Uniform encodings

- Program-data pairs — separation of problem specification from instance description
- Problem specification — a finite PS program
- Expressive power — the class of problems that can be represented by finite PS programs

- Finite PS programs capture precisely the class NPMV

Computing with logic PS

Coding a search problem

- Select the language:
  - a schema to represent problem instances
  - appropriate program predicates
- Specify the problem as a PS-theory (program) P

Solving for an instance D

- Compute models of (P, D), that is, models of $CWA(D) \cup P$
- Ground $CWA(D) \cup P$
- Simplify: use $CWA(D)$ and typing
- Search for a model
- Can use SAT solvers directly!

Computing with logic PS

Tools

- Grounder *psgrnd* — outputs CNF theories (DIMACS)
- Your favorite SAT solver — computes solutions

A more general view, Mitchell-Ternovska, 2005

Model-extension problem

- Given a FO formula φ and a finite structure A_I for vocabulary $\sigma \subseteq \mathit{vocab}(\varphi)$
- Is there a structure A — an extension of A_I to $\mathit{vocab}(\varphi)$ — such that $A \models \varphi$?
- NEXPTIME-complete

Model-extension problem *parametrized*

- Fix φ and σ
- Input: finite structure A_I for the vocabulary σ
- More general version of logic PS (no restriction to conjunctions of clauses)
- Captures class NPMV

What's missing from logic PS?

Inductive definitions

- Expressing some concepts neither straightforward nor concise
- Case in point: inductive definitions (IDs)
  for instance, transitive closure of a graph
- ID-logic addresses the problem!

ID-Logic, Denecker 1998

Integrates inductive definitions with FOL

- Inductive definitions
  - logic programs with the well-founded semantics
- Intuitive semantics
- Semantics grounded in algebra of operators on lattices and their fixpoints
- Directly extends logic PS
- Simple and concise encoding of logic programs with stable model semantics
- Addresses major KR problems

  Denecker-Ternovska on ID-logic and situation calculus, 2004
- Computational support — in progress

Looking back

A common pattern

- Coding (modeling) — place for KR
- Grounding — bridge
- Search (model finding) — place for SAT
- Output (recovering solutions from models)

Coding

Languages

- Logic programming with stable model semantics (roots in KR)
- Logic PS (close to classical logic)
- ID-logic (a common extension)

  work on specific modeling syntax in progress

Language extensions

Support for “high-level” constraints

- Substantial theoretical work — emerging consensus on the semantics

  Denecker-Bruynooghe-Pelov 2005; Pontelli-Son 2003-05; Faber-Leone-Pfeifer 2004; Marek-Niemelä-MT 2004; Liu-MT 2005

- Currently mostly pseudo-boolean (weight) constraints

  For every vertex y the sum of weights of vertices in U reachable from y is at least k

  $k\{selected\_to\_U(X) = w(X)[X] : edge(Y, X)\}$

  $L\{p(t) = w(t)[X] : cond(s)\}U$

- Need for standardized high-level syntax

Program development

Modeling methodology

- KR focused on representing knowledge needed to create intelligent reasoning agents
  Modeling search problems poses different challenges
- When to use LP, PS or ID logic?
  Does it make a difference as they have the same expressive power (assuming no function symbols)?

Program optimization

- Program equivalence
- Optimization by replacing parts of programs with other equivalent ones

Much theoretical work: Lifschitz-Pearce-Valverde 2001; Turner 2003; Lin 2002; Eiter-Fink-Woltran 2003-05; MT 2006

Program development

Debugging support

- Detecting syntactic errors easy — still needs to be done
- Support for verifying semantic correctness — a major problem mostly untouched

Grounding

A bridge to search

- Logic programming (with extensions): lparse, dlv
- Logic PS with pseudo-boolean atoms and monotone IDs: psgrnd
- Logic ID — grounders under development
- For many problems grounding is a bottleneck
  - astronomical sizes of ground theories
  - time needed to produce them

Long-term opportunities for enhancing search

Interleave grounding and search — tighter integration

- Non-ground program (theory) as data structure for the ground counterpart
- Search w/out grounding

  proposed by Ginsberg and Parkes, 2002
- Search with partial grounding only

Search (model finding) — a place for SAT

Native solvers

- smodels and dlv for logic programming
- aspps and wsat(plpb) for logic PS (with extensions)
- SAT and PB(SAT) can provide ideas and techniques
  - clause learning, restarts, data structures
- Not much of it incorporated so far

Direct use of SAT and PB(SAT)

- Translations of grounder output to DIMACS and OPT
- Straightforward for logic PS (and implemented)
- Less straightforward for logic programming and ID-logic

program completion and loop formulas; Lin-Zhao 2002,
Giunchiglia-Lierler-Maratea 2004
Denecker; Mitchell-Ternovska in progress

Closing words

Main points

- Knowledge representation languages (LP with stable-model semantics, logic PS, logic ID) offer an interface to satisfiability
- KR and SAT together open a way to general-purpose, flexible and fast programming environments for solving search problems
- However, major research challenges still unresolved and must be tackled for this method of solving search problems to gain broader acceptance

Links and References

Links to software

- *psgrnd/aspps* — `www.cs.uky.edu/ai/`
- *lparse/smodels* — `www.tcs.hut.fi/Software/smodels/`
- *dlv* — `www.tuwien.ac.at/proj/dlv/`

References

On the following pages

C. Baral.
Knowledge representation, reasoning and declarative problem solving.
Cambridge University Press, 2003.

T. Dell'Armi, W. Faber, G. Ielpa, N. Leone, S. Perri, and G. Pfeifer.
System description: DLV with aggregates.
In V. Lifschitz and I. Niemelä, editors, *Logic programming and Nonmonotonic Reasoning, Proceedings of the t^{th} International Conference*, volume 2923, pages 326–330. Springer, 2004.

M. Denecker.
The well-founded semantics is the principle of inductive definition.
In J. Dix, L. Fariñas del Cerro, and U. Furbach, editors, *Logics in Artificial Intelligence*, volume 1489, pages 1–16. Springer, 1998.

M. Denecker and E. Ternovska.
Inductive situation calculus.
In D. Dubois, C. A. Welty, and M.-A. Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 9th International Conference (KR2004)*, pages 545–553. AAAI Press, 2004.

Marc Denecker.
Extending classical logic with inductive definitions.
In *Computational Logic - CL 2000*, volume 1861 of *Lecture Notes in Computer Science*, pages 703–717. Springer, 2000.

D. East and M. Truszczyński.
Datalog with constraints.
In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*, pages 163–168. AAAI Press, 2000.

D. East and M. Truszczyński.
Predicate-calculus based logics for modeling and solving search problems.
ACM Transactions on Computational Logic, 7:38–83, 2006.

T. Eiter and M. Fink.
Uniform equivalence of logic programs under the stable model semantics.
In *Proceedings of the 2003 International Conference on Logic Programming*, volume 2916 of *Lecture Notes in Computer Science*, pages 224–238, Berlin, 2003. Springer.

T. Eiter, M. Fink, and S. Woltran.
Semantical characterizations and complexity of equivalences in answer set programming.
ACM Transactions on Computational Logic, 2006.
To appear.

Wolfgang Faber, Nicola Leone, and Gerald Pfeifer.
Recursive aggregates in disjunctive logic programs: Semantics and complexity.
In *Proceedings of the 9th European Conference on Artificial Intelligence (JELIA 2004)*, volume 3229 of *LNAI*, pages 200 – 212. Springer, 2004.

M. Gelfond and N. Leone.
Logic programming and knowledge representation – the A-prolog perspective.
Artificial Intelligence, 138:3–38, 2002.

M. Gelfond and V. Lifschitz.
The stable semantics for logic programs.
In *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

M.L. Ginsberg and A.J. Parkes.
Satisfiability algorithms and finite quantification.
In *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning, (KR-2000)*, pages 690–701. Morgan Kaufmann, 2000.

E. Giunchiglia, Y. Lierler, and M. Maratea.
SAT-based answer-set programming.
In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-2004)*, pages 61–66. AAAI Press, 2004.

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

F. Lin.
Reducing strong equivalence of logic programs to entailment in classical propositional logic.
In *Principles of Knowledge Representation and Reasoning, Proceedings of the 8th International Conference (KR2002)*. Morgan Kaufmann Publishers, 2002.

F. Lin and Y. Zhao.
ASSAT: Computing answer sets of a logic program by SAT solvers.
In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-2002)*, pages 112–117. AAAI Press, 2002.

L. Liu and M. Truszczyński.
Properties of programs with monotone and convex constraints.
In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pages 701–706. AAAI Press, 2005.

V.W. Marek, I. Niemelä, and M. Truszczyński.
Characterizing stable models of logic programs with cardinality constraints.
In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning*, volume 2923 of *Lecture Notes in Artificial Intelligence*, pages 154–166. Springer, 2004.

V.W. Marek and M. Truszczyński.
Logic programs with abstract constraint atoms.
In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)*, pages 86–91. AAAI Press, 2004.

J. McCarthy.
Epistemological problems of Artificial Intelligence.
In *Proceedings of the 5th Interational Joint Conference on Artificial Intelligence*, pages 1038–1044, 1977.

J. McCarthy and P. Hayes.
Some philosophical problems from the standpoint of artificial intelligence.
In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

R.C. Moore.
Possible-world semantics for autoepistemic logic.
In *Proceedings of the Workshop on Non-Monotonic Reasoning*, pages 344–354, 1984.
Reprinted in: M. Ginsberg, ed., *Readings on Nonmonotonic Reasoning*, pages 137–142, Morgan Kaufmann, 1990.

N. Pelov, M. Denecker, and M. Bruynooghe.
Well-founded and stable semantics of logic programs with aggregates.
Theory and Practice of Logic Programming, 2006.
Accepted (available at `http://www.cs.kuleuven.ac.be/~dtai/projects/ALP/TPLP/`).

R. Reiter.
A logic for default reasoning.
Artificial Intelligence, 13(1-2):81–132, 1980.

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

T. Son and E. Pontelli.
A constructive semantic characterization of aggregates in anser set programming.
Theory and Practice of Logic Programming, 2006.
Accepted (available at `http://arxiv.org/abs/cs.AI/0601051`).

T. Son, E. Pontelli, and P.H. Tu.
Answer sets for logic programs with arbitrary abstract constraint atoms.
In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 129–134. AAAI Press, 2006.

M. Truszczyński.
Strong and uniform equivalence of nonmonotonic theories — an algebraic approach.
In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning, KR 2006*, pages 389–399. AAAI Press, 2006.

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