Answer Set Programming: an Introduction

Stefania Costantini

Dipartimento di Informatica
Universita' degli Studi di L'Aquila
via Vetoio Loc. Coppito, I-67100 L'Aquila (Italy)
stefcost@di.univaq.it

Example: 3-coloring in Logic Programming

Problem: assigning colors red/blue/green to vertices of a graph, so as no adjacent vertices have the same color.

Logic programming representation of the graph:

$node(0..3).$

$col(red).$
$col(blue).$
$col(green).$

$edge(0,1).$
$edge(1,2).$
$edge(2,0).$
$edge(1,3).$
$edge(2,3).$

... Inference Engine ...

Expected solutions:

$\{color(0,red), color(1,blue), color(2,green), color(3,red)\}$
$\{color(0,red), color(1,green), color(2,blue), color(3,red)\}$
$\{color(0,blue), color(1,red), color(2,green), color(3,blue)\}$
$\{color(0,blue), color(1,green), color(2,red), color(3,blue)\}$
$\{color(0,green), color(1,blue), color(2,red), color(3,green)\}$
$\{color(0,green), color(1,red), color(2,blue), color(3,green)\}$

3-coloring in Prolog

$graph\_coloring(G, L) :- gr\_col(G, [], L).$

$gr\_col([], L, L) :- !.$
$gr\_col([N|R], P, L) :- vertex\_color(N, C, P), gr\_col(R, [color(N, C)|P], L).$

$vertex\_color(N, C, P) :- col(C), ok\_color(N, C, P).$
$neighbourhood(N, B, L) :- (edge(N, N1); edge(N1, N)),$
$member(N1, B), !,$
$neighbourhood(N, [N1|B], L).$
$neighbourhood(\_, L, L).$

$ok\_color(N, C, P) :- neighbourhood(N, [], L), check\_color(L, C, P).$

$check\_color([], \_, \_) :- !.$
$check\_color([N1|B], C, P) :- \backslash + member(color(N1, C), P), !,$
$check\_color(B, C, P).$

$member(E, [E|X]) :- !.$
$member(E, [\_|X]) :- member(E, X).$

$?- consult[3col.pro]$
$?- graph\_coloring([0, 1, 2, 3], L).$
$L = [color(0, red), color(1, blue), color(green), color(3, red)];$
$L = [color(0, red), color(1, green), color(blue), color(3, red)]$
$\ldots$

3-coloring in Answer Set Programming

$color(X, red)|color(X, blue)|color(X, green) : -node(X).$

$: -edge(X, Y), col(C), color(X, C), color(Y, C).$

$hide\, node(X).$
$hide\, edge(X, Y).$
$hide\, col(C).$

Using the SMODELS inference engine we obtain:
$lparse\ <\ 3col.txt \mid smodels\, 0$

$Answer1$
$\{color(0, red), color(1, blue), color(green), color(3, red)\}$
$Answer2$
$\{color(0, red), color(1, green), color(blue), color(3, red)\}$
$\ldots$

Horn Logic Programming

Unification + SLD-Resolution

Least (Herbrand) model semantics

Function symbols and recursive definitions
(infinite Herbrand Universe)
Expressivity: Church-Turing computability

DATALOG: no function symbols
(finite Herbrand Universe, finite program grounding)
Expressivity: proper subset of P

Negation in Logic Programming

Negation operator not (\+)
Negation As (finite) Failure
based on Closed World Assumption

Unique least model no longer guaranteed

$a \leftarrow not\, b.$
$b \leftarrow not\, a.$
Classical models $\{a\}$ and $\{b\}$.

First group of semantic proposals:
keep a single intended model

By narrowing class of programs:
perfect model (stratification) (Apt, Blair & Walker)

By weakening semantics:
well-founded model $\langle T; F \rangle$ (Van Gelder, Ross & Schlipf)

Second group of semantic proposal:
collection of *intended* models:

- supported models (Clark)
- stable models (Gelfond & Lifschitz)

Stable Model Semantics (Answer Sets Semantics)

Nice formal features:
related to well-founded semantics, and default logic

Difficult to reconcile with query-based logic programming (skeptical semantics? unique stable model? Too complex!)

Solution: new logic programming paradigm

- SLP: Stable Logic Programming
  (Marek & Truszczynski)
- ASP: Answer Set Programming (Gelfond & Lifschitz)

for for DATALOG plus negation (no function symbols, or limited use)

Solutions are represented by stable models (answer sets), and not by answer substitutions in response to a query.

Inference engine: answer set solvers
SMODELS, Dlv, DeRes, CCalc

Complexity: existence of a stable model NP-complete

Expressivity:

- all decision problems in NP and
- all search problems whose associated decision problems are in NP (claim!)

Answer Set/Stable model semantics

$p :- not\, p.$
Classical model $\{p\}$ NOT STABLE

$a :- not\, b.$
$b :- not\, a.$
Classical models $\{a\}$, $\{b\}$ STABLE

$p :- not\, p, not\, a.$
$a :- not\, b.$
$b :- not\, a.$
Classical models
$\{b, p\}$ NOT STABLE
$\{a\}$ STABLE

$p :- not\, p.$
$p :- not\, a.$
$a :- not\, b.$
$b :- not\, a.$
Classical models
$\{b, p\}$ STABLE
$\{a, p\}$ NOT STABLE

Answer set semantics

Constraints

$:\!-v, w, z.$
rephrased as
$p :\!-not\, p, v, w, z.$

Disjunction

$v|w|z.$
rephrased as
$v :\!-not\, w, not\, z.$
$w :\!-not\, v, not\, z.$
$z :\!-not\, v, not\, w.$

Choice (XOR)

$v + w + z.$
rephrased as
$v|w|z.$
$:\!-w, z.$
$:\!-v, z.$
$:\!-v, w.$

Answer set semantics

Classical Negation $\neg p$

$-p :- q, r.$
rephrased as
$p' :- q, r.$
$:- p, p'$

Example

$mushroom(boletus\_edulis).$
$mushroom(amanita\_phalloides).$

$eat(M) :- mushroom(M), -poisonous(M).$

$not\ poisonous$ would be hazardous!

$discard(M) :- mushroom(M), poisonous(M).$

$poisonous(M) :- not\ -poisonous(M).$
$-poisonous(boletus\_edilis).$

Drawbacks of stable model semantics

Fix-point definition:
interpretation S is a stable model iff $S = \Gamma(S)$

No Relevance :
for atom A, $REL\_RUL(A)$ may have stable models where A is true/false, while the overall program does not.

Example :
$p :- not\, p, not\, a.$

$q :- not\, q, not\, b.$

$a :- b.$
$b :- a.$

$REL\_RUL(a) = \{a :- b.\ b :- a.\}$
with stable models $\{a\}$ and $\{b\}$.

Overall program: no stable models.

P_1 with stable models, P_2 with stable models
$P_1 \cup P_2$ may not have stable models.

Answer Set Programming: deals with Uncertainty

$new(n1).$
$new(n2).$
$source(n1, a1).$
$association(a1).$
$source(n2, government).$

$economical(N) : -new(N), source(N, A),$
$\qquad\qquad\qquad\qquad economical\_association(A).$
$cultural(N) : -new(N), source(N, A), cultural\_association(A).$
$political(N) : -new(N), source(N, government).$

$cultural\_association(A) : -association(A),$
$\qquad\qquad\qquad\qquad not\, economical\_association(A).$
$economical\_association(A) : -association(A),$
$\qquad\qquad\qquad\qquad not\, cultural\_association(A).$

$economical(N) : -new(N), not\, cultural(N).$
$cultural(N) : -new(N), not\, political(N), not\, economical(N).$
$political(N) : -new(N), not\, cultural(N).$

Answer Set Programming: deals with Uncertainty

Prolog:

$? - economical(n1).$
$? - political(n1).$
$? - cultural(n1)$
infinite loop!

$? - economical(n2).$
$? - cultural(n1)$
infinite loop!

$? - political(n2).$ yes

Well-Founded Semantics:
truth value undefined instead of infinite loop

Answer Set Solver (SMODELS):
Answer1
$\{political(n2), economical(n2), cultural(n1), cultural\_association(a1)\}$
Answer2
$\{political(n2), economical(n2), political(n1), political\_association(a1)\}$

Each Answer Set makes a different (consistent) hypothesis
on association a1 (political/cultural).

Answer Set Programming and Intelligent Agents

Planning oriented Agents: cycle *observe-think-act*

- observe: collect new facts;
- think: answer set programming for making a plan;
- act: execute the plan.

Reactive/proactive agents: cycle *event-condition-action*

- event: come asynchronously form the outside, or from the inside;
- condition: internal inference process, possibly involving answer set programming for checking constraints, resolving uncertainty, making a plan;
- action: possibly affect the environment.