Answer Set Programming

From Wikipedia, the free encyclopedia

Answer set programming (ASP) is a form of declarative programming oriented towards difficult (primarily NP-hard) search problems. It is based on the stable model (answer set) semantics of logic programming. In ASP, search problems are reduced to computing stable models, and answer set solvers -- programs for generating stable models -- are used to perform search. The computational process employed in the design of many answer set solvers is an enhancement of the DPLL algorithm and, in principle, it always terminates (unlike Prolog query evaluation, which may lead to an infinite loop).

In a more general sense, ASP includes all applications of answer sets to knowledge representation[1][2] and the use of Prolog-style query evaluation for solving problems arising in these applications.

Graph coloring

An n-coloring of a graph G is a function color\ from its set of vertices to \{1,\dots,n\} such that color(x)\neq color(y) for every pair of adjacent vertices x,y. We would like to use ASP to find an n-coloring of a given graph (or determine that it does not exist).

This can be accomplished using the following Lparse program:

c(1..n).
1 {color(X,I) : c(I)} 1 :- v(X).
:- color(X,I), color(Y,I), e(X,Y), c(I).
Line 1 defines the numbers 1, \ldots, n to be colors. According to the choice rule in Line 2, a unique color i should be assigned to each vertex x. The constraint in Line 3 prohibits assigning the same color to vertices x and y if there is an edge connecting them.

If we combine this file with a definition of G, such as

\[
v(1..100). \text{ } % \text{ 1,...,100 are vertices} \\
e(1,55). \text{ } % \text{ there is an edge from 1 to 55}
\]

\[
\ldots
\]

and run smodels on it, with the numeric value of n specified on the command line, then the atoms of the form color(\ldots,\ldots) in the output of smodels will represent an n-coloring of G.

The program in this example illustrates the "generate-and-test" organization that is often found in simple ASP programs. The choice rule describes a set of "potential solutions" -- a simple superset of the set of solutions to the given search problem. It is followed by a constraint, which eliminates all potential solutions that are not acceptable. However, the search process employed by smodels and other answer set solvers is not based on trial and error.

[edit] Large clique

A clique in a graph is a set of pairwise adjacent vertices. The following lparse program finds a clique of size \(\geq n\) in a given graph, or determines that it does not exist:

\[
n \{\text{in}(X) : \text{v}(X)\}. \\
\text{:- in}(X), \text{in}(Y), \text{v}(X), \text{v}(Y), X!=Y, \text{not e}(X,Y), \text{not e}(Y,X).
\]

This is another example of the generate-and-test organization. The choice rule in Line 1 "generates" all sets consisting of \(\geq n\) vertices. The constraint in Line 2
"weeds out" the sets that are not cliques.

[edit] Hamiltonian cycle

A Hamiltonian cycle in a directed graph is a cycle that passes through each vertex of the graph exactly once. The following Lparse program can be used to find a Hamiltonian cycle in a given directed graph if it exists; we assume that 0 is one of the vertices.

\[
\{\text{in}(X,Y)\} :\text{-} e(X,Y).
\]

\[
\left\{\begin{array}{c}
\text{:-} 2 \{\text{in}(X,Y) : e(X,Y)\}, v(X).
\text{:-} 2 \{\text{in}(X,Y) : e(X,Y)\}, v(Y).
\end{array}\right.
\]

\[
r(X) : \text{in}(0,X), v(X).
\]

\[
r(Y) : r(X), \text{in}(X,Y), e(X,Y).
\]

\[
\text{:- not } r(X), v(X).
\]

The choice rule in Line 1 "generates" all subsets of the set of edges. The three constraints "weed out" the subsets that are not Hamiltonian cycles. The last of them uses the auxiliary predicate \( r(x) \) ("\( x \) is reachable from 0") to prohibit the vertices that do not satisfy this condition. This predicate is defined recursively in Lines 4 and 5.

This program is an example of the more general "generate, define and test" organization: it includes the definition of an auxiliary predicate that helps us eliminate all "bad" potential solutions.