Artificial Intelligence

Logic 5: Answer Set Programming

Jendrik Seipp

Linköping University

Feedback

post feedback and ask questions anonymously at

`https://padlet.com/jendrikseipp/tddc17`

Basic Concepts

What is Answer Set Programming?

- Answer Set Programming (ASP) is a form of declarative programming for combinatorial search problems.
- It is based on the stable model (answer set) semantics of logic programming.
- ASP provides a way to describe a problem in terms of constraints and rules and then solve it using a solver.

Example 1: Simple Facts and Rules

Family tree

```
father(paul, mary).
father(paul, lisa).
woman(mary).

sister(X, Y) :- father(P, X), father(P, Y), woman(X), X != Y.

#show sister/2.
```

Example 1: Simple Facts and Rules

Family tree

```
father(paul, mary).
father(paul, lisa).
woman(mary).

sister(X, Y) :- father(P, X), father(P, Y), woman(X), X != Y.

#show sister/2.
```

Solution:

sister(mary,lisa)

Syntax and Semantics

Syntax of ASP

- Atoms: basic propositions, e.g., `a`, `b`, `c`
- Literals: atoms or their negations, e.g., `a`, `-b`
- Default negation: unproven propositions, e.g., `not c`
- Rules: have the form `head :- body.` where the body is a conjunction of literals, e.g., `a :- b, c, not d.`
- Constraints: rules without a head remove solution candidates, e.g., `:- b, c.` (b and c cannot be true at the same time)
- Facts: ground rules with an empty body, e.g., `a.`
- Choice rules: allow to express choices, e.g., `1 {a, b, c} 2.` (make at least one and at most two of a, b, c true)

Semantics of ASP

- ASP uses the concept of stable models to determine which sets of literals are accepted.
- An answer set is a set of literals that satisfies:
  - all program rules when each literal is interpreted as true or false
  - the program's constraints
- Answer sets represent solutions to the encoded problem.

Relation to First-Order Logic (FOL)

- ASP builds on concepts from FOL, such as variables, literals and rules.
- ASP rules can be seen as a special form of implications in FOL.
- Unlike FOL, ASP focuses on finding models (answer sets) that satisfy all constraints.

FOL vs. ASP: Focus on Models

First-Order Logic (FOL)

- Model Finding: Focus on proving logical entailment or satisfiability.
- Constraints: Expressed using quantified implications/negations.
- Example:
  - Axiom 1: $\forall x(P(x) \rightarrow Q(x))$
  - Axiom 2: $\forall x(Q(x) \rightarrow R(x))$
  - Goal: Prove $P(a) \rightarrow R(a)$

Answer Set Programming (ASP)

- Model Finding: Find stable models that satisfy all constraints.
- Constraints: Integral to the problem specification.
- Example:
  - `q(X) :- p(X).`
  - `r(X) :- q(X).`
  - `p(a).`
- Solution: {p(a), q(a), r(a)}

Modeling with ASP

ASP Capabilities

- ASP can model various combinatorial problems, including:
  - Constraint satisfaction problems (e.g., Sudoku)
  - Graph problems (e.g., Coloring, Hamiltonian Path)
  - Planning and scheduling (e.g., job-shop scheduling)
  - Knowledge representation (e.g., formalizing commonsense knowledge)
- Its declarative nature allows for concise problem descriptions and use of powerful solvers.

ASP Solver

- Clingo: `https://potassco.org/`
- Installation on Ubuntu: `sudo apt install clingo`

Example 1: Simple Facts and Rules

Family tree

```
father(paul, mary).
father(paul, lisa).
woman(mary).

sister(X, Y) :- father(P, X), father(P, Y), woman(X), X != Y.

#show sister/2.
```

Example 1: Simple Facts and Rules

Family tree

```
father(paul, mary).
father(paul, lisa).
woman(mary).

sister(X, Y) :- father(P, X), father(P, Y), woman(X), X != Y.

#show sister/2.
```

Solution:

sister(mary,lisa)

Example 1: Simple Facts and Rules

Family tree extended

```
father(paul, mary).
father(paul, lisa).
woman(mary).
sister(anna, paul).

sister(X, Y) :- father(P, X), father(P, Y), woman(X), X != Y.
aunt(X, Y) :- sister(X, P), father(P, Y).

#show aunt/2.
```

Example 1: Simple Facts and Rules

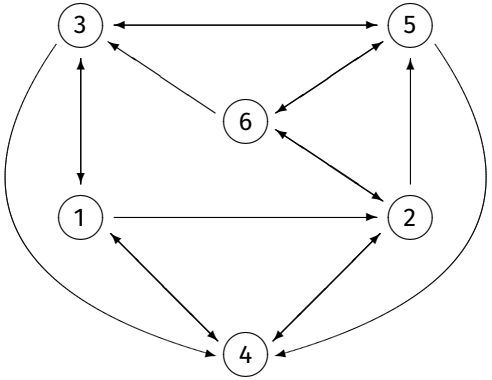
Family tree extended

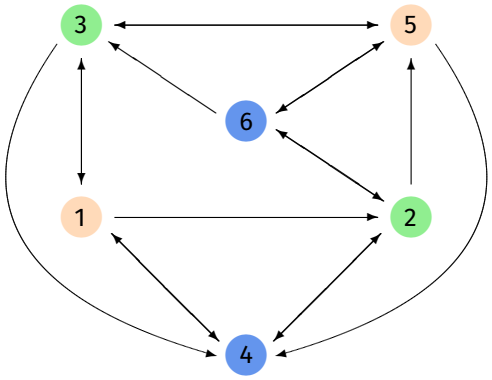
```
father(paul, mary).
father(paul, lisa).
woman(mary).
sister(anna, paul).

sister(X, Y) :- father(P, X), father(P, Y), woman(X), X != Y.
aunt(X, Y) :- sister(X, P), father(P, Y).

#show aunt/2.
```

Solution:

aunt(anna,mary) aunt(anna,lisa)

Example 2: Graph Coloring





Graph coloring (from `https://github.com/potassco/guide/`)

```
node(1..6).

edge(1, (2;3;4)).  edge(2, (4;5;6)).  edge(3, (1;4;5)).
edge(4, (1;2)).    edge(5, (3;4;6)).  edge(6, (2;3;5)).

#const n = 3.
{ color(X, 1..n) } = 1 :- node(X).
:- edge(X, Y), color(X, C), color(Y, C).
```

Example 3: Sudoku Solver

Sudoku part 1

```
% Define the ranges for columns, rows, and numbers.
x(1..9).  % columns
y(1..9).  % rows
n(1..9).  % numbers (1 to 9)

% Ensure each cell has exactly one number from 1 to 9.
{sudoku(X, Y, N) : n(N)} = 1 :- x(X), y(Y).

% Cells (X, Y) and (A, B) belong to the same 3x3 subgrid.
subgrid(X, Y, A, B) :-
    x(X), x(A), y(Y), y(B),
    (X - 1) / 3 == (A - 1) / 3,
    (Y - 1) / 3 == (B - 1) / 3.
```

Example 4: Sudoku Solver

Sudoku part 2

```
% Constraint: No number can appear twice in the same row.
:- sudoku(X, Y, N), sudoku(A, Y, N), X != A.

% Constraint: No number can appear twice in the same column.
:- sudoku(X, Y, N), sudoku(X, B, N), Y != B.

% Constraint: No number can appear twice in the same 3x3 subgrid.
:- sudoku(X, Y, N), sudoku(A, B, N), subgrid(X, Y, A, B), X != A, Y != B.

% Display the solution.
#show sudoku/3.
```

Solution:

sudoku(1,1,2) sudoku(2,2,5) ...

Summary

Summary

- Answer Set Programming is a powerful paradigm for solving combinatorial problems.
- Relies on declarative specifications and stable model semantics.
- Strongly connected to concepts from first-order logic, extending them with practical solving techniques.