EVOLOG

Actions and Modularization in Lazy-Grounding Answer Set Programming

Master Thesis Proposal by Michael Langowski
Advisors: Thomas Eiter, Antonius Weinzierl

A Lofty Vision

```
#import "xml".
#import "io".

action main(in: arg/1, out: empty) {
    infile(X) :- arg(X).
    graph_parse_result(R) : @xml::parse_graph[X] = R :- infile(X).

    graph(G) :- graph_parse_result(graph(G)).
    parsing_error(MSG) :- graph_parse_result(err(MSG)).

    graph_coloring(G, COL) :- graph(G), {all}@3col[G](COL).
    write_result(R) : @io::write_list[OUT, COL] = R :-
        OUT = "col-" + IDX, graph_coloring(G, coloring(IDX, COL)).
}

predicate 3col(in:graph/1, out: coloring/1) {
    node(N) :- [ node(N) in G : graph(G) ].
    edge(N1, N2) :- [ edge(N1, N2) in G : graph(G) ].

    1 {col(N, red); col(N, blue); col(N, green)} 1 :- node(N).
    :- col(N1, C), col(N2, C), edge(N1, N2).

    coloring(COL) :- COL = #list-collect{ node_colored(N, C) : col(N, C) }.
}
```

A Lofty Vision

```
#import "xml".
#import "io".

action main(in: arg/1, out: empty) {
    infile(X) :- arg(X).
    graph_parse_result(R) : @xml::parse_graph[X] = R :- infile(X).

    graph(G) :- graph_parse_result(graph(G)).
    parsing_error(MSG) :- graph_parse_result(err(MSG)).

    graph_coloring(G, COL) :- graph(G), {all}@3col[G](COL).
    write_result(R) : @io::write_list[OUT, COL] = R :-
        OUT = "col-" + IDX, graph_coloring(G, coloring(IDX, COL)).
}

predicate 3col(in:graph/1, out: coloring/1) {
    node(N) :- [ node(N) in G : graph(G) ].
    edge(N1, N2) :- [ edge(N1, N2) in G : graph(G) ].

    1 {col(N, red); col(N, blue); col(N, green)} 1 :- node(N).
    :- col(N1, C), col(N2, C), edge(N1, N2).

    coloring(COL) :- COL = #list-collect{ node_colored(N, C) : col(N, C) }.
}
```

How do we enable this in ASP?

Thesis Goals

- Define a semantics for actions in ASP such that
  - every executed action is visible in answer set
  - actions are executed in correct order while preserving declarative semantics
  - “vanilla” ASP is a subset of the resulting language
- Define a simple modularization and scoping mechanism which
  - offsets impact of (potential) restrictions imposed by action semantics
  - offers a way of writing composite actions
  - increases code readability and reusability

Thesis Goals

Based on the existing lazy-grounding ASP solver Alpha [alp],

- create a prototype solver with action and modularization support
- create at least one sample application demonstrating these capabilities

Thesis Outline

- Introduction
  - ASP in Software Engineering
  - Motivating Examples
- Preliminaries
  - ASP Core-2 Standard
  - Lazy-Grounding
- Evolog Language Specification
- Implementation
- Verification and Evaluation

Thesis Outline

Evolog Language Specification

- Action Semantics
  - Inspired by Monads in Haskell
  - Actions are interpreted function symbols
  - Interpretation function for actions is part of an Evolog model (frame)
  - World state at time of execution is an input parameter
  - Actions are restricted to "stratifiable bottom" of a program
- Module semantics
  - Modules are a special case of external atom
  - Inputs and outputs are terms

State of the art

Actions

- ACTHEX - DLVHEX extension with comprehensively defined action semantics [ahx]
- oClingo - No true semantic support for actions, but powerful external atoms [ocl]

Modules

- “nonmonotonic modular logic programs” - powerful, but very computationally costly module semantics [mlp]
- “Templates” - purely syntactic, code reuse mechanism [tpl]
- clingo multi-shot solving - parameterized grounding through API [cms]

References

- [alp] - Antonius Weinzierl et al.
  The alpha solver for lazy-grounding answer-set programming.
  2019
- [ahx] - Selen Basol, Ozan Erdem, Michael Fink, and Giovambattista Ianni.
  Hex programs with action atoms.
  In *Technical Communications of the 26th International Conference on Logic Programming*.
  SchlossDagstuhl-Leibniz-Zentrum fuer Informatik, 2010.
- [ocl] - Martin Gebser, Torsten Grote, Roland Kaminski, and Torsten Schaub.
  Reactive answer set programming.
  In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 54–66.
  Springer, 2011.
- [mlp] - Thomas Krennwallner.
  Modular nonmonotonic logic programs.
  PhD thesis,
  Technical University of Vienna, 2018.

References

- [tpl] - Giovambattista Ianni, Giuseppe Ielpa, Adriana Pietramala, Maria Carmela Santoro, and Francesco Calimeri.
  Enhancing answer set programming with templates.
  In *NMR*, pages 233–239,
  2004.
- [cms] - Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub.
  Multi-shot asp solving with clingo.
  Theory and Practice of Logic Programming, 19(1):27–82,
  2019.