Online appendix for the paper

SeaLion: An Eclipse-based IDE for Answer-Set Programming with Advanced Debugging Support

published in Theory and Practice of Logic Programming

PAULA-ANDRA BUSONIU[1], JOHANNES OETSCH[1],
JÖRG PÜHRER[1], PETER SKOČOVSKÝ[2], HANS TOMPITS[1]∗

[1] *Technische Universität Wien,*
Institut für Informationssysteme 184/3,
Favoritenstrasse 9-11, A-1040 Vienna, Austria,
[2] *Universidade Nova de Lisboa,*
CENTRIA and Departamento de Informatica,
2829-516 Caparica, Portugal,
(*e-mail:* {andra.busoniu,aifargonos}@gmail.com)
(*e-mail:* {oetsch,puehrer,tompits}@kr.tuwien.ac.at)

submitted 10 April 2013; revised 23 June 2013; accepted 5 July 2013

Appendix A The Source Code of the Example Program for Stepping

```
% INPUT %
% Having a grid of cells, some of which contain natural numbers,
row(1..4).
col(1..4).
number(c(3,2), 3).
number(c(4,4), 1).

% DEFINE %
cell(c(Y,X)) :- row(Y), col(X).

maxcol(X) :- col(X), not col(X+1).
maxrow(Y) :- row(Y), not row(Y+1).
mincol(X) :- col(X), not col(X-1).
minrow(Y) :- row(Y), not row(Y-1).

adjacent(c(Y,X), c(Y-1,X)) :- cell(c(Y,X)), not minrow(Y).
adjacent(c(Y,X), c(Y,X+1)) :- cell(c(Y,X)), not maxcol(X).
adjacent(c(Y,X), c(Y+1,X)) :- cell(c(Y,X)), not minrow(Y).
adjacent(c(Y,X), c(Y,X-1)) :- cell(c(Y,X)), not mincol(X).

% GENERATE %
% (i) each cell is either black or white,
{white(C) : cell(C)}.
black(C) :- cell(C), not white(C).
```

∗ This work was partially supported by the Austrian Science Fund (FWF) under project P21698.

```
% CHECK %
% (ii) cells with numbers are white,
:- black(C), number(C,_).

% (iii) there are no 2x2 blocks of black cells,
:- black(c(Y,X)), black(c(Y+1,X)), black(c(Y,X+1)), black(c(Y+1,X+1)).

% (iv) all black cells are transitively connected,
black_reach(C1, C2) :- black(C1), black(C2), adjacent(C1, C2).
black_reach(C1, C3) :- black_reach(C1, C2), black(C3), adjacent(C2, C3).
:- black(C1), black(C2), not black_reach(C1, C2).

% (v) each maximal group of white cells that are transitively connected
% must contain exactly one cell with a number;
% this number must be the number of cells in the group,
group(C, C) :- number(C, _).
group(C1, C2) :- group(C3, C2), adjacent(C3, C1), white(C1).

:- number(C1, N), not N{group(C2, C1) : cell(C2)}N.

in_group(C) :- group(C, _).
:- white(C), not in_group(C).

:- white(C1), not 1{group(C1, C2) : number(C2, _)}1.
```

Appendix B Additional Screenshots

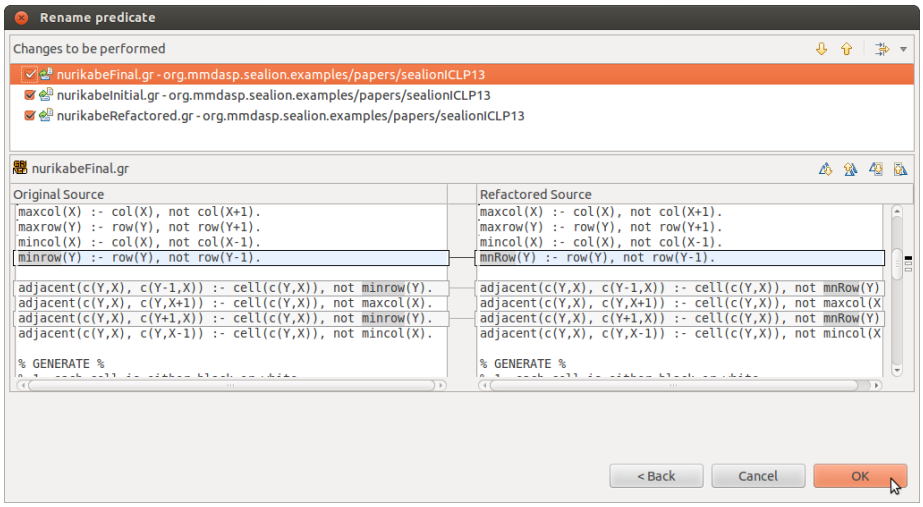

Fig. B 1. Reviewing file changes implied by renaming predicate `minrow/2` to `mnRow/2`.

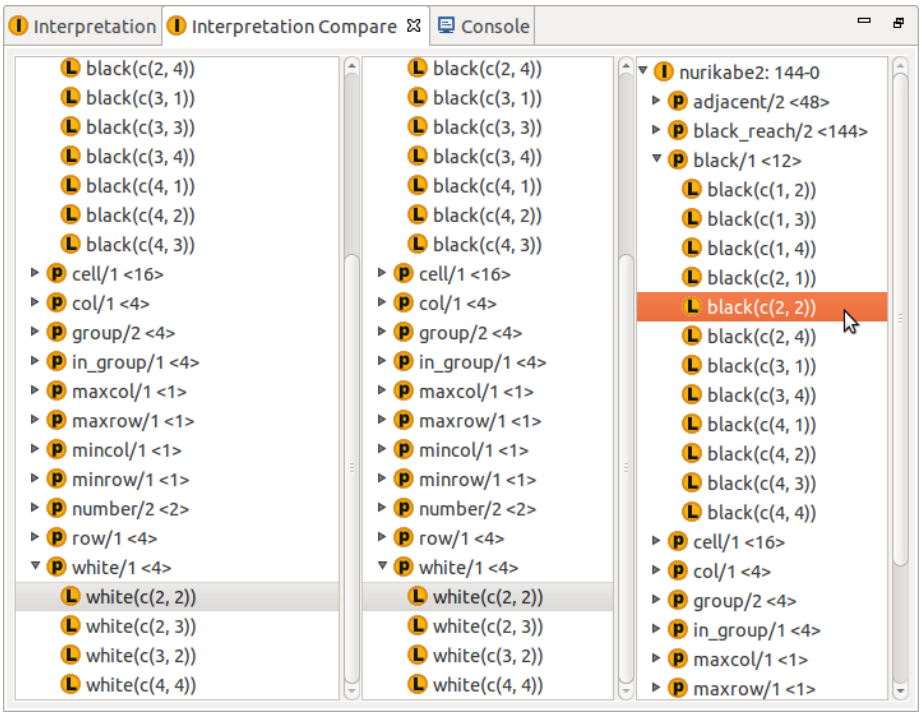

Fig. B 2. `SeaLion`'s interpretation compare view.

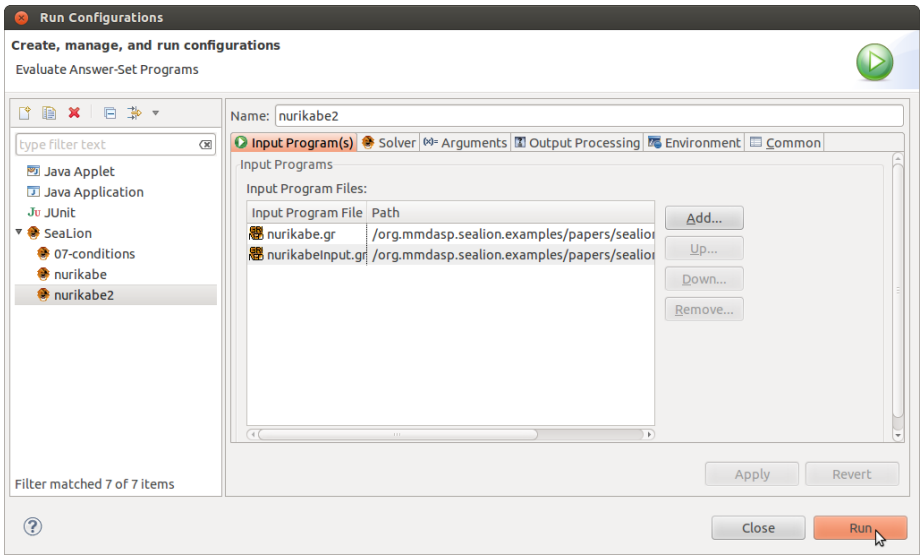

Fig. B 3. Selecting two source files in Eclipse's launch configuration dialog.

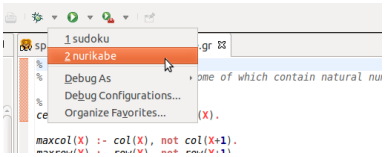

Fig. B 4. Launching in debug mode.

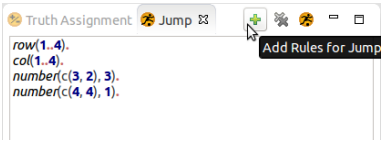

Fig. B 5. Jump view.

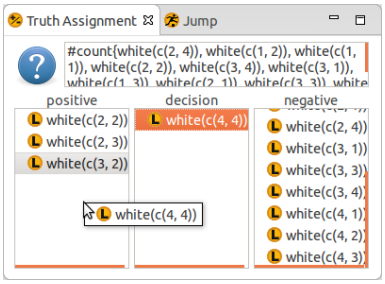

Fig. B 6. Truth Assignment view.

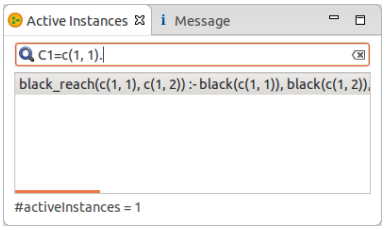

Fig. B 7. Filtering active instances

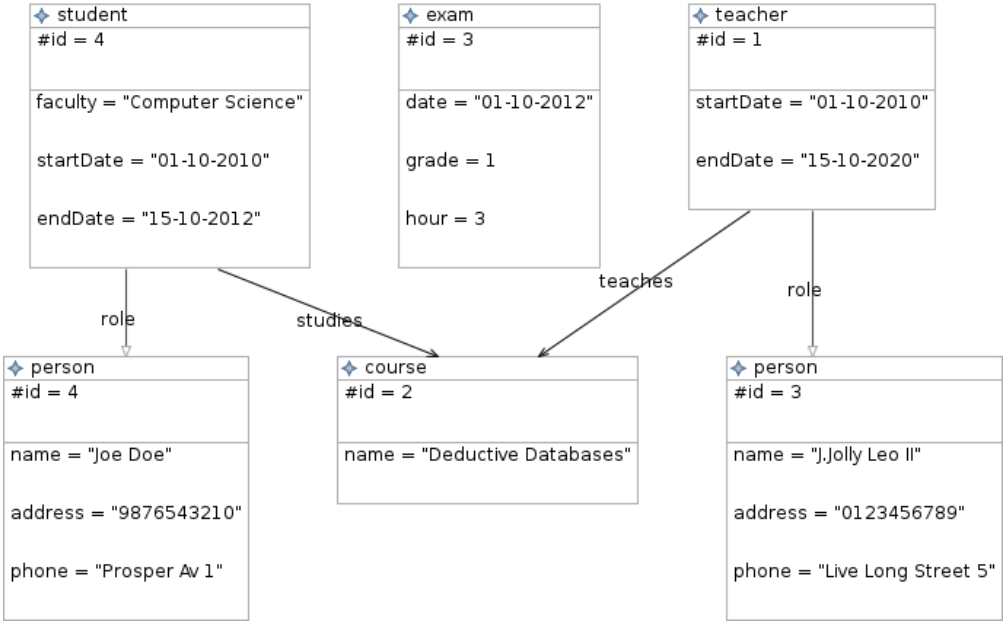

Fig. B 8. A UML object diagram based on the model of Fig. 4.

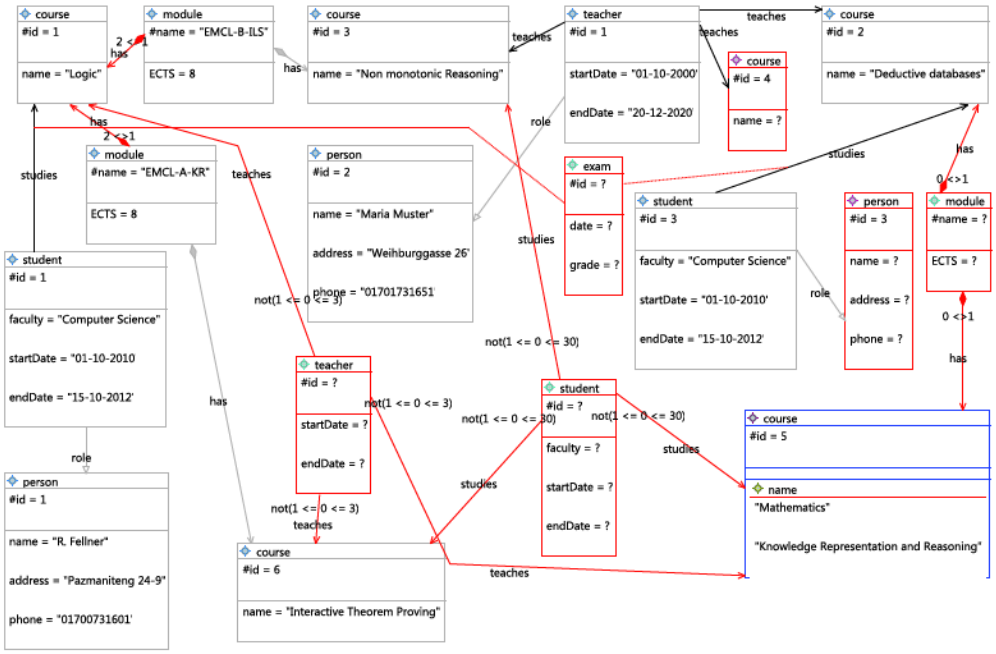

Fig. B 9. Another UML object diagram based on the model of Fig. 4. Concepts and relations displayed in red indicate different violations of the domain constraints in the visualised interpretation.

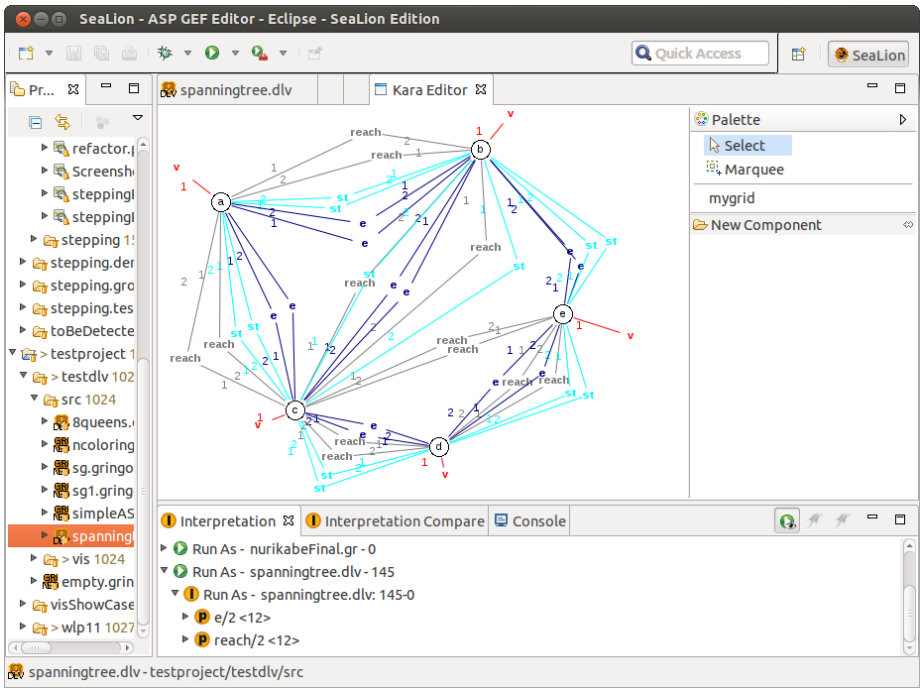

Fig. B 10. A generic visualisation of a spanning tree interpretation (the layout of the graph has been manually optimised in the editor).

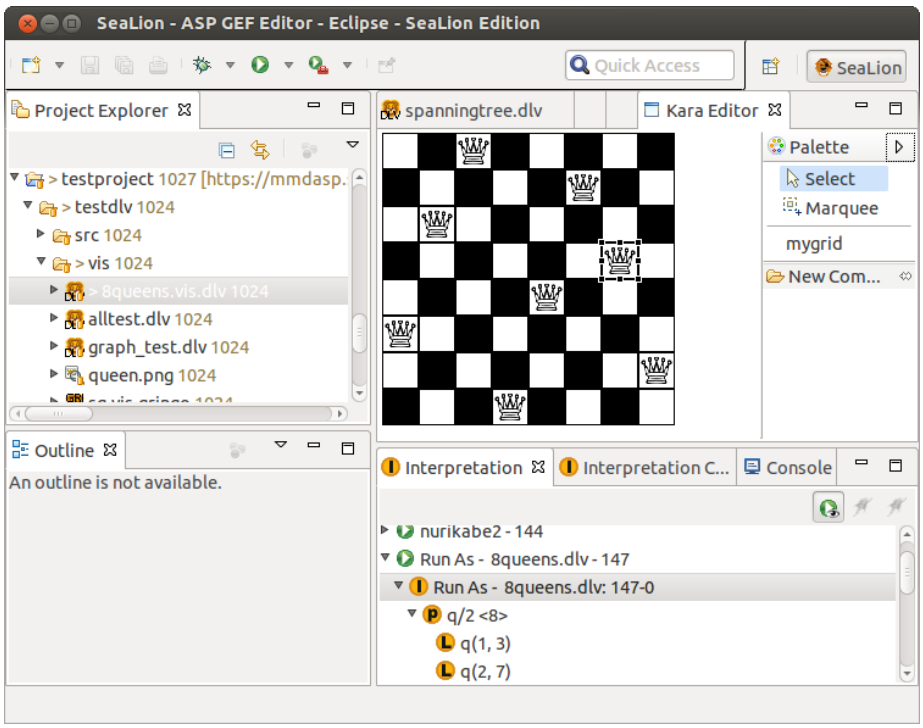

Fig. B 11. A customised visualisation of an instance of the 8-queens problem.