SMT-based Constraint Answer Set Solver
EZSMT

Yuliya Lierler

Available at: https://works.bepress.com/yuliya_lierler/88/
The EZSMT Solver: Constraint Answer Set Programming meets SMT

Yuliya Lierler
with Da Shen and Benjamin Susman

University of Nebraska at Omaha
What is Constraint Answer Set Programming (CASP)?

Start by: What is Answer Set Programming (ASP)?

Answer Set Programming (ASP)

- Prominent *declarative (constraint) programming paradigm* geared towards solving difficult combinatorial search problems
- Popular in knowledge representation and reasoning
- Syntactically: language of logic programs (rooting in Prolog)
- Semantically: answer set semantics to interpret these programs

In practice:

- An ASP user encodes the specifications of a problem in a way that the answer sets of the program are in one-to-one correspondence with the solutions to the problem
- An answer set solving system is used to enumerate answer sets of a program
A Running Example

Traveling Salesman Problem (TSP)

Given: a graph consisting of nodes as cities and edges as roads. Each road directly connects a pair of cities, and costs a salesman some time to go through. The salesman wants to pass each city exactly once.

Asked to find: a route traversing all the cities under certain maximum cost of total time starting and finishing at a particular initial city.
Sample Instance of TSP

An instance with max cost 4

Encoded in ASP with facts:

city(a). ... city(d).
initial(a).
road(a,b). ... road(b,d).
cost(a,b,1). ... cost(b,d,2).
maxCost(4).

Only instances of route/2 in answer set 1:

route(a,b) route(b,c) route(c,d) route(d,a)

Only instances of route/2 in answer set 2:

route(a,d) route(d,c) route(c,b) route(b,a)

Solution 1

Solution 2
### Encoding of TSP using ASP

#### ASP Encoding in the standard ASP-Core-2 Language

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>road(Y,X):-road(X,Y).</td>
<td>If there is a road from X to Y then it is also a road from Y to X</td>
</tr>
<tr>
<td>cost(Y,X,C):-cost(X,Y,C).</td>
<td></td>
</tr>
<tr>
<td>1{ route(X,Y) : road(X,Y) }1 :: city(X).</td>
<td>For each city, we choose one route <em>leaving from</em> the city.</td>
</tr>
<tr>
<td>1{ route(X,Y) : road(X,Y) }1 :: city(Y).</td>
<td>For each city, we choose one route <em>going to</em> the city.</td>
</tr>
<tr>
<td>reached(X) :- initial(X).</td>
<td>The initial city is reached.</td>
</tr>
<tr>
<td>reached(Y) :- reached(X), route(X,Y).</td>
<td>If city X is reached and the route from city X to city Y is chosen, then city Y is also reached.</td>
</tr>
<tr>
<td>:- city(X), not reached(X).</td>
<td>Having a city that has not been reached leads to a contradiction.</td>
</tr>
<tr>
<td>:- W &lt; #sum{C,X,Y:route(X,Y),cost(X,Y,C)},maxCost(W).</td>
<td>The total time cost of a chosen route greater than maximal cost leads to a contradiction.</td>
</tr>
<tr>
<td>#show route/2.</td>
<td>A directive to only print route predicate within found solutions</td>
</tr>
</tbody>
</table>

% clingo tsp.lp instance.lp 0
Answer: 1
route(a,d) route(c,b) route(d,c) route(b,a)
Answer: 2
route(a,b) route(b,c) route(c,d) route(d,a)

⇐ call to an answer set solving system instructing to enumerate all solutions
Solving

Typical Solver Architecture

logic program → Grounder → grounded program → ASP Solver → answer sets

Grounder

Instantiates variables of the rules with object constants of the program.
For example,

\[
\text{road}(Y,X) :- \text{road}(X,Y). \quad \Rightarrow \quad \text{road}(b,a) :- \text{road}(a,b).
\]
\[
\text{road}(c,b) :- \text{road}(b,c).
\]
\[
\ldots
\]

Output: A propositional program containing propositional atoms only.

ASP Solver

Searches for answer sets of a propositional program in spirit of a SAT solver that
searches for satisfying assignments of a given propositional formula.
## One of the Challenges in ASP

### Grounding

May result in too large of a propositional program to begin search with.

### Grounding over Reals

Impossible.

### Constraint Answer Set Programming (CASP)

- Allows to bypass grounding partially by introducing *constraint variables* over large domains and encapsulating these into *constraint atoms*.
- *Constraint atoms* are practically *theory atoms* of SMT (Lierler and Susman 2016).
- Search procedures of CASP combine techniques of "SAT solving" and Constraint Processing.
### Encoding of Traveling Sales Person (TSP) using CASP

<table>
<thead>
<tr>
<th>CASP in language EZCSP supported by EZSMT</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>road(Y,X):-road(X,Y).</td>
<td>If there is a road from X to Y then it is also a road from Y to X</td>
</tr>
<tr>
<td>1{ route(X,Y) : road(X,Y) }1 :- city(X).</td>
<td>For each city, we choose one route <em>leaving from</em> the city.</td>
</tr>
<tr>
<td>1{ route(X,Y) : road(X,Y) }1 :- city(Y).</td>
<td>For each city, we choose one route <em>going to</em> the city.</td>
</tr>
<tr>
<td>reached(X) :- initial(X).</td>
<td>The initial city is reached.</td>
</tr>
<tr>
<td>reached(Y) :- reached(X), route(X,Y).</td>
<td>If city X is reached and the route from city X to city Y is chosen, then city Y is also reached.</td>
</tr>
<tr>
<td>:- city(X), not reached(X).</td>
<td>Having a city that has not been reached leads to a contradiction.</td>
</tr>
<tr>
<td>:- W &lt; #sum{C,X,Y:route(X,Y),cost(X,Y,C)},maxCost(W).</td>
<td>The total time cost of a chosen route greater than maximal cost leads to a contradiction.</td>
</tr>
</tbody>
</table>

#### Declaration of constraint vars
- cspvar(c(X,Y)) :- road(X,Y).
- required(c(X,Y) == 0) :- road(X,Y), not route(X,Y).
- required(c(X,Y) == C) :- cost(X,Y,C), route(X,Y).
- required(sum([c/2], <=, W)) :- maxCost(W).

With few modifications, we could use the resulting CASP encoding to solve TSP even if costs of the roads were reals.
What about SMT Solving

**Given**

- *constraint atoms* are practically *theory atoms* of SMT
- Search procedures of CASP combine techniques of "SAT solving" and Constraint Processing
- Search procedures of SMT *extend* these of SAT solving

**Natural question**

Why not to use SMT solvers in place of specialized procedures?
### Niemela 2008

Logic program = Clark’s completion + level ranking formulas

↑ Propositional formula

↑ difference logic (SMT) formula

### Lierler and Susman 2017

Constraint logic program = Clark’s completion with theory atoms

↑ SMT formula

+ level ranking formulas

↑ difference logic (SMT) formula
SMT-based CASP Solver EZSMT

Grounding and Computing *Completion*

Computing Four Variants of *Level Rankings*

EZWMT Transformer

SMT Solver

Solutions

If Non-tight

Completion And Level Ranking Formulas

SMT Formulas

Computing Multiple Solutions

If Tight

preprocessed CASP Program
Relatives of EZSMT

- CASP solvers CLINGCON and CLINSO[LP], University of Potsdam
- CASP solver EZCSP, Drexel University and University of Nebraska Omaha
- ASP+DifferenceLogic solver DINGO, Aalto University
- ASP+SMT solver ASPMT2SMT, Arizona State University
## Experiments

<table>
<thead>
<tr>
<th>Category</th>
<th>Benchmark</th>
<th>CLINGCON</th>
<th>EZSMT</th>
<th>EZSMT</th>
<th>EZSMT</th>
<th>EZSMT</th>
<th>EZCSP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Z3</td>
<td>CVC4</td>
<td>YICES</td>
<td>CLASP SWI-PROLOG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NT-IL</td>
<td>RoutingMin(100)</td>
<td>4.68</td>
<td>8.76</td>
<td>28.2</td>
<td>5.81</td>
<td>126007(70)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RoutingMax(100)</td>
<td>3144</td>
<td>459</td>
<td>17170</td>
<td>5190</td>
<td>180000(100)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Trav. Sals.(30)</td>
<td>455</td>
<td>7347(4)</td>
<td>3841</td>
<td>1881(1)</td>
<td>14.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Labyrinth(22)</td>
<td>3002(1)</td>
<td>9510(1)</td>
<td>4252(1)</td>
<td>4399(2)</td>
<td>12558(6)</td>
<td></td>
</tr>
<tr>
<td>T-IL</td>
<td>Rev. Fold.(50)</td>
<td>326</td>
<td>6058(2)</td>
<td>52786(24)</td>
<td>27840(14)</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Incr. Sched.(30)</td>
<td>9080(30)</td>
<td>9200(5)</td>
<td>10407(5)</td>
<td>9098(5)</td>
<td>41446(21)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Weigh. Seq.(30)</td>
<td>52.5</td>
<td>29.2</td>
<td>25.4</td>
<td>5.23</td>
<td>54000(30)</td>
<td></td>
</tr>
<tr>
<td>T-INL</td>
<td>Car(8)</td>
<td>/</td>
<td>0.32</td>
<td>/</td>
<td>0.25</td>
<td>10.1</td>
<td></td>
</tr>
<tr>
<td>T-RL</td>
<td>Blending(30)</td>
<td>/</td>
<td>88.4</td>
<td>54.1</td>
<td>47.4</td>
<td>18322(9)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generator(8)</td>
<td>/</td>
<td>0.58</td>
<td>1.15</td>
<td>0.48</td>
<td>5641(3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Robotics(8)</td>
<td>/</td>
<td>0.4</td>
<td>0.61</td>
<td>0.39</td>
<td>2.04</td>
<td></td>
</tr>
<tr>
<td>T-RNL</td>
<td>Bounc. Ball(5)</td>
<td>/</td>
<td>3663(2)</td>
<td>/</td>
<td>0.98</td>
<td>9000(5)</td>
<td></td>
</tr>
<tr>
<td>T-ML</td>
<td>Blending*(30)</td>
<td>/</td>
<td>5573(2)</td>
<td>167</td>
<td>/</td>
<td>/</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

- **CASP programs**: traditionally, solved by ASP solvers and finite domain constraint solvers; in EZSMT, solved by SMT solvers

- Experimental analysis shows that EZSMT is a viable tool for finding solutions to CASP programs

- EZSMT also offers constraint/theory atoms over theories that no other CASP tool can handle

- By making clear the translation of arbitrary CASP programs to SMT, our work will boost the cross-fertilization between the two distinct areas of automated reasoning.
Future work

- Designing non-prototypical CASP language allowing for intuitive programming with its extended portfolio of specialized constraints
  → Moving away from EZSCP language
- Adopting sophisticated computation of
  ▶ multiple solutions
  ▶ optimal solutions
  → Moving away from SMT-LIB towards utilizing solver specific API’s
- Incorporating MINIZINC as back-end
  → Comprehensive experimental analysis across distinct automated reasoning paradigm
Basis for Presentation

Related Papers

- Da Shen and Yuliya Lierler. "SMT-based Constraint Answer Set Solver EZSMT+ for Non-tight Programs" 16th International Conference on Principles of Knowledge Representation and Reasoning (KR-2018)
- Benjamin Susman and Yuliya Lierler. "SMT-based Constraint Answer Set Solver EZSMT (System Description)" Proceedings of the 32nd International Conference on Logic Programming (ICLP-2016)

Home of EZSMT

http://unomaha.edu/nlpkr/software/ezsmt/
Thanks and Questions