Exploiting Justifications for Lazy Grounding of Answer Set Programs

Bart Bogaerts[†] Antonius Weinzierl[‡]

† KU Leuven, Department of Computer Science
Celestijnenlaan 200A, Leuven, Belgium

‡ Aalto University, Department of Computer Science
FI-00076 AALTO, Finland

July 18, 2018

Bart Bogaerts is a postdoctoral fellow of the Research Foundation – Flanders (FWO).
Antonius Weinzierl has been supported by the Academy of Finland, project 251170.

Introduction

- Answer-Set Programming (ASP) a KR formalism.
- Rule-based, nonmonotonic, expressive (NP-hard).

Example (Encoding Graph Coloring)

$$\{pickedCol(N, C)\} \leftarrow node(N) \wedge color(C).$$

$$colored(N) \leftarrow pickedCol(N, C).$$

$$\leftarrow node(N) \wedge \neg colored(N).$$

$$\leftarrow node(N) \wedge pickedCol(N, C1) \wedge pickedCol(N, C2) \wedge C1 \neq C2.$$

$$\leftarrow edge(N1, N2) \wedge pickedCol(N1, C) \wedge pickedCol(N2, C).$$

- Formal semantics: answer sets.

Introduction

- Answer-Set Programming (ASP) a KR formalism.
- Rule-based, nonmonotonic, expressive (NP-hard).

Example (Encoding Graph Coloring)

$$\{pickedCol(N, C)\} \leftarrow node(N) \wedge color(C).$$

$$colored(N) \leftarrow pickedCol(N, C).$$

$$\leftarrow node(N) \wedge \neg colored(N).$$

$$\leftarrow node(N) \wedge pickedCol(N, C1) \wedge pickedCol(N, C2) \wedge C1 \neq C2.$$

$$\leftarrow edge(N1, N2) \wedge pickedCol(N1, C) \wedge pickedCol(N2, C).$$

- Formal semantics: answer sets.

ASP Evaluation

- Traditional two-step evaluation: ground-and-solve.
- Grounding: replace variables by ground terms.
- Solving: mainly SAT techniques.

Example (Grounding)

$$\{pickedCol(N, C)\} \leftarrow node(N) \wedge color(C).$$

$$color(red).\ color(blue).\ color(green).\ color(yellow).$$

$$node(1).\ node(2).$$

$$\{pickedCol(1, red)\} \leftarrow node(1) \wedge color(red).$$

$$\{pickedCol(1, green)\} \leftarrow node(1) \wedge color(green).$$

$$\{pickedCol(1, blue)\} \leftarrow node(1) \wedge color(blue).$$

$$\{pickedCol(1, yellow)\} \leftarrow node(1) \wedge color(yellow).$$

$$\{pickedCol(2, red)\} \leftarrow node(2) \wedge color(red).$$

$$\vdots$$

$$\{pickedCol(2, yellow)\} \leftarrow node(2) \wedge color(yellow).$$

ASP Evaluation

- Traditional two-step evaluation: ground-and-solve.
- Grounding: replace variables by ground terms.
- Solving: mainly SAT techniques.

Example (Grounding)

$$\{pickedCol(N, C)\} \leftarrow node(N) \land color(C).$$

$$color(red).\ color(blue).\ color(green).\ color(yellow).$$

$$node(1).\ node(2).$$

$$\{pickedCol(1, red)\} \leftarrow node(1) \land color(red).$$

$$\{pickedCol(1, green)\} \leftarrow node(1) \land color(green).$$

$$\{pickedCol(1, blue)\} \leftarrow node(1) \land color(blue).$$

$$\{pickedCol(1, yellow)\} \leftarrow node(1) \land color(yellow).$$

$$\{pickedCol(2, red)\} \leftarrow node(2) \land color(red).$$

$$\vdots$$

$$\{pickedCol(2, yellow)\} \leftarrow node(2) \land color(yellow).$$

ASP Evaluation

- Traditional two-step evaluation: ground-and-solve.
- Grounding: replace variables by ground terms.
- Solving: mainly SAT techniques.

Example (Grounding)

$$\{pickedCol(N, C)\} \leftarrow node(N) \wedge color(C).$$

$color(red).\ color(blue).\ color(green).\ color(yellow).$

$node(1).\ node(2).$

$\{pickedCol(1, red)\} \leftarrow node(1) \wedge color(red).$

$\{pickedCol(1, green)\} \leftarrow node(1) \wedge color(green).$

$\{pickedCol(1, blue)\} \leftarrow node(1) \wedge color(blue).$

$\{pickedCol(1, yellow)\} \leftarrow node(1) \wedge color(yellow).$

$\{pickedCol(2, red)\} \leftarrow node(2) \wedge color(red).$

$\vdots$

$\{pickedCol(2, yellow)\} \leftarrow node(2) \wedge color(yellow).$

ASP Evaluation

- Traditional two-step evaluation: ground-and-solve.
- Grounding: replace variables by ground terms.
- Solving: mainly SAT techniques.

Example (Grounding)

$$\{pickedCol(N, C)\} \leftarrow node(N) \land color(C).$$

$$color(red).\ color(blue).\ color(green).\ color(yellow).$$

$$node(1).\ node(2).$$

$$\{pickedCol(1, red)\} \leftarrow node(1) \land color(red).$$

$$\{pickedCol(1, green)\} \leftarrow node(1) \land color(green).$$

$$\{pickedCol(1, blue)\} \leftarrow node(1) \land color(blue).$$

$$\{pickedCol(1, yellow)\} \leftarrow node(1) \land color(yellow).$$

$$\{pickedCol(2, red)\} \leftarrow node(2) \land color(red).$$

$$\vdots$$

$$\{pickedCol(2, yellow)\} \leftarrow node(2) \land color(yellow).$$

ASP Evaluation

- Traditional two-step evaluation: ground-and-solve.
- Grounding: replace variables by ground terms.
- Solving: mainly SAT techniques.

Example (Grounding)

$$\{pickedCol(N, C)\} \leftarrow node(N) \wedge color(C).$$

$$color(red).\ color(blue).\ color(green).\ color(yellow).$$

$$node(1).\ node(2).$$

$$\{pickedCol(1, red)\} \leftarrow node(1) \wedge color(red).$$

$$\{pickedCol(1, green)\} \leftarrow node(1) \wedge color(green).$$

$$\{pickedCol(1, blue)\} \leftarrow node(1) \wedge color(blue).$$

$$\{pickedCol(1, yellow)\} \leftarrow node(1) \wedge color(yellow).$$

$$\{pickedCol(2, red)\} \leftarrow node(2) \wedge color(red).$$

$$\vdots$$

$$\{pickedCol(2, yellow)\} \leftarrow node(2) \wedge color(yellow).$$

ASP Evaluation

- Traditional two-step evaluation: ground-and-solve.
- Grounding: replace variables by ground terms. (exponential!)
- Solving: mainly SAT techniques.

Example (Grounding)

$$\{pickedCol(N, C)\} \leftarrow node(N) \wedge color(C).$$

$color(red).\ color(blue).\ color(green).\ color(yellow).$

$node(1).\ node(2).$

$\{pickedCol(1, red)\} \leftarrow node(1) \wedge color(red).$

$\{pickedCol(1, green)\} \leftarrow node(1) \wedge color(green).$

$\{pickedCol(1, blue)\} \leftarrow node(1) \wedge color(blue).$

$\{pickedCol(1, yellow)\} \leftarrow node(1) \wedge color(yellow).$

$\{pickedCol(2, red)\} \leftarrow node(2) \wedge color(red).$

$\vdots$

$\{pickedCol(2, yellow)\} \leftarrow node(2) \wedge color(yellow).$

Lazy-Grounding

- Grounding explosion, problem in practice.
- ⇒ Avoid grounding bottleneck.
- Lazy grounding:
  - Interleave grounding and solving phases.
  - Several solvers available (GASP, ASPeRiX, Omiga, Alpha).
  - New foundation for solving ⇒ brings own challenges.
- Alpha combines lazy-grounding with CDCL (conflict-driven clause learning).
- But: sometimes search gets stuck.

Lazy-Grounding

- Grounding explosion, problem in practice.
- ⇒ Avoid grounding bottleneck.
- Lazy grounding:
  - Interleave grounding and solving phases.
  - Several solvers available (GASP, ASPeRiX, Omiga, Alpha).
  - New foundation for solving ⇒ brings own challenges.
- Alpha combines lazy-grounding with CDCL (conflict-driven clause learning).
- But: sometimes search gets stuck.

Lazy-Grounding

- Grounding explosion, problem in practice.
- ⇒ Avoid grounding bottleneck.
- Lazy grounding:
  - Interleave grounding and solving phases.
  - Several solvers available (GASP, ASPeRiX, Omiga, Alpha).
  - New foundation for solving ⇒ brings own challenges.
- Alpha combines lazy-grounding with CDCL (conflict-driven clause learning).
- But: sometimes search gets stuck.

Lazy-Grounding

- Grounding explosion, problem in practice.
- $\Rightarrow$ Avoid grounding bottleneck.
- Lazy grounding:
  - Interleave grounding and solving phases.
  - Several solvers available (GASP, ASPeRiX, Omiga, Alpha).
  - New foundation for solving $\Rightarrow$ brings own challenges.
- Alpha combines lazy-grounding with CDCL (conflict-driven clause learning).
- But: sometimes search gets stuck.

Alpha's Core Algorithm

Alpha Algorithm: perform iteratively these steps by priority:

1. (conflict): if clause violated, analzye conflict (1UIP), learn new clause, backjump (CDCL).
2. (propagate): unit propagation assign false/true (BCP).
3. (justify): set rule head justified-true if all positive body atoms justified-true.
4. (ground): ground new rules based on atoms assigned true.
5. (decide): pick one atom and assign it true or false.
6. (justification-conflict): if all atoms assigned and some atom true but not justified-true, backtrack last decision.

- Novel characterization based on justifications.
- Previously, three truth values: false/must-be-true/true.
- Using justification: false/true/justified-true.

Alpha's Core Algorithm

Alpha Algorithm: perform iteratively these steps by priority:

1. (conflict): if clause violated, analzye conflict (1UIP), learn new clause, backjump (CDCL).
2. (propagate): unit propagation assign false/true (BCP).
3. (justify): set rule head justified-true if all positive body atoms justified-true.
4. (ground): ground new rules based on atoms assigned true.
5. (decide): pick one atom and assign it true or false.
6. (justification-conflict): if all atoms assigned and some atom true but not justified-true, backtrack last decision.

- Novel characterization based on justifications.
- Previously, three truth values: false/must-be-true/true.
- Using justification: false/true/justified-true.

Problem in Justification-Conflict

Example (Graph Coloring, again)

If $colored(2)$ is true but not justified, what caused it?

$$\begin{aligned} colored(N) &\leftarrow pickedCol(N, C). \\ &\leftarrow node(N) \land \neg colored(N). \end{aligned}$$

Trivial in the ground case. Hard to say without grounding.

- $\Rightarrow$ Solver cannot backjump and revert the wrong guess.
- $\Rightarrow$ Chronological backtracking, exponential time overhead.

Problem in Justification-Conflict

Example (Graph Coloring, again)

If $colored(2)$ is true but not justified, what caused it?

$$colored(N) \leftarrow pickedCol(N, C).$$
$$\leftarrow node(N) \land \neg colored(N).$$

Trivial in the ground case. Hard to say without grounding.

- $\Rightarrow$ Solver cannot backjump and revert the wrong guess.
- $\Rightarrow$ Chronological backtracking, exponential time overhead.

Justifications

- **Justification** J for $\neg p$ explains for each rule that could derive p, why it does not fire in interpretation I.

Example

$$colored(N) \leftarrow pickedCol(N, C).$$

Justifications

- Justification J for $\neg p$ explains for each rule that could derive p, why it does not fire in interpretation I.

Example

$$colored(N) \leftarrow pickedCol(N, C).$$

$\neg colored(2) \rightarrow \neg pickedColor(2, red)$

$\neg colored(2) \rightarrow \neg pickedColor(2, blue)$

$\neg colored(2) \rightarrow \neg pickedColor(2, green)$

$\neg colored(2) \rightarrow \neg pickedColor(2, yellow)$

Justifications (2)

Theorem

If p is true but not justified in justification-conflict, then ¬p is justified.

- Problem: justifications consider ground rules.

⇒ Lift justifications.

Example

Justifications (2)

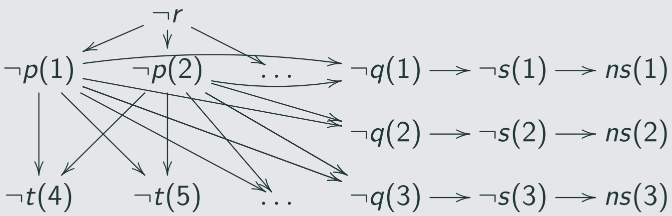
Theorem

If p is true but not justified in justification-conflict, then $\neg p$ is justified.

- Problem: justifications consider ground rules.

⇒ Lift justifications.

Example



Justifications (2)

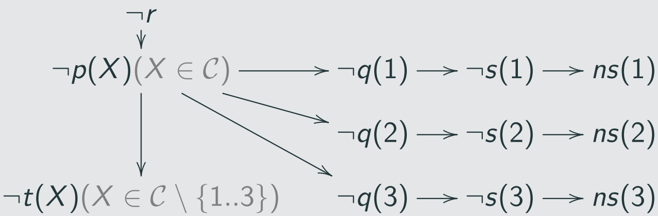
Theorem

If p is true but not justified in justification-conflict, then $\neg p$ is justified.

- Problem: justifications consider ground rules.

$\Rightarrow$ Lift justifications.

Example



Algorithm

- In justification-conflict, compute justification J.
- Turn justification J for $\neg p$ into new clause:
    - Leaves L of J influence p being not justified.
    - New clause: $\neg p \vee \bigvee_{\ell \in L} \ell$

Theorem

New clause is in conflict with current solver state, and satisfied in all answer sets.

- Add clause $\Rightarrow$ standard conflict analysis does backjumping.
- Computing J: top-down analysis (details: paper, poster).

Algorithm

- In justification-conflict, compute justification J.
- Turn justification J for $\neg p$ into new clause:
  - Leaves L of J influence p being not justified.
  - New clause: $\neg p \vee \bigvee_{\ell \in L} \ell$

Theorem

New clause is in conflict with current solver state, and satisfied in all answer sets.

- Add clause $\Rightarrow$ standard conflict analysis does backjumping.
- Computing J: top-down analysis (details: paper, poster).

Algorithm

- In justification-conflict, compute justification J.
- Turn justification J for $\neg p$ into new clause:
  - Leaves L of J influence p being not justified.
  - New clause: $\neg p \vee \bigvee_{\ell \in L} \ell$

Theorem

New clause is in conflict with current solver state, and satisfied in all answer sets.

- Add clause $\Rightarrow$ standard conflict analysis does backjumping.
- Computing J: top-down analysis (details: paper, poster).

Algorithm

- In justification-conflict, compute justification J.
- Turn justification J for $\neg p$ into new clause:
  - Leaves L of J influence p being not justified.
  - New clause: $\neg p \vee \bigvee_{\ell \in L} \ell$

Theorem

New clause is in conflict with current solver state, and satisfied in all answer sets.

- Add clause $\Rightarrow$ standard conflict analysis does backjumping.
- Computing J: top-down analysis (details: paper, poster).

Evaluation (1)

Size	Alpha	*Alpha*$_J$	Clingo
10	0.81	0.79	0.00
20	2.55	0.81	0.00
30	300.00(5)	0.85	0.00
40	300.00(5)	0.92	0.00
50	300.00(5)	0.90	0.00
65	300.00(5)	0.86	0.00
100	300.00(5)	1.02	0.00
200	300.00(5)	1.04	0.01
400	300.00(5)	1.23	0.01
1000	300.00(5)	1.56	0.01

Table 1: Benchmark results for Two-way-derivation. Runtime is in seconds, timeouts in parentheses.

Evaluation (2)

Size	Alpha	*Alpha_J*	Alpha	*Alpha_J*	Clingo
	Original (no constraint)		With constraint		Both
10	5.58	1.10	1.11	1.07	0.01
20	39.20(1)	1.46	1.31	1.25	0.01
30	69.31(2)	1.92	1.59	1.62	0.01
40	252.74(8)	2.33	1.88	1.97	0.01
75	300.00(10)	3.96	3.35	3.38	0.02
100	300.00(10)	5.90	4.76	5.03	0.03
200	300.00(10)	13.44	10.27	9.96	0.08
400	300.00(10)	33.96	22.15	24.85	0.27
500	300.00(10)	44.62	32.27	33.55	0.39
750	300.00(10)	82.97	68.20	66.50	0.87
1000	300.00(10)	131.17	101.88	105.93	1.54

Table 2: Benchmark results for Graph-5-coloring. Runtime in seconds, timeouts in parentheses.

Evaluation (3)

Size	Alpha	*Alpha*$_J$	Clingo
10	0.88	0.89	0.01
20	1.04	1.05	0.03
40	11.46	1.91	0.26
80	60.99(2)	3.39	2.62
100	90.92(3)	4.47	5.53
200	91.23(3)	13.64	47.16
400	32.29(1)	32.31(1)	276.18(8 memout)
1000	3.80	3.69	300.00(10 memout)
2000	92.90(3)	92.86(3)	300.00(10 memout)
4000	97.16(3)	97.05(3)	300.00(10 memout)

Table 3: Benchmark results for Non-partition-removal-coloring. Runtime in seconds, timeouts in parentheses.

Conclusion

- Addressed inherent problem of lazy grounding.
- Benchmarks: Justification analysis can avoid exponential overhead of chronological backtracking.
- Implemented in the lazy-grounding ASP solver Alpha.

  `github.com/alpha-asp/alpha`
- More details on the poster.

Thanks.

Conclusion

- Addressed inherent problem of lazy grounding.
- Benchmarks: Justification analysis can avoid exponential overhead of chronological backtracking.
- Implemented in the lazy-grounding ASP solver Alpha.

  `github.com/alpha-asp/alpha`
- More details on the poster.

Thanks.