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

Example (Encoding Graph Coloring)

\[ \text{pickedCol}(N, C) \leftarrow \text{node}(N) \land \text{color}(C). \]
\[ \text{colored}(N) \leftarrow \text{pickedCol}(N, C). \]
\[ \text{node}(N) \land \neg \text{colored}(N). \]
\[ \text{node}(N) \land \text{pickedCol}(N, C_1) \land \text{pickedCol}(N, C_2) \land C_1 \neq C_2. \]
\[ \text{edge}(N_1, N_2) \land \text{pickedCol}(N_1, C) \land \text{pickedCol}(N_2, C). \]

• Formal semantics: answer sets.
• Answer-Set Programming (ASP) a KR formalism.
• Rule-based, nonmonotonic, expressive (NP-hard).

Example (Encoding Graph Coloring)

\[
\{ \text{pickedCol}(N, C) \} \leftarrow \text{node}(N) \land \text{color}(C).
\]
\[
\text{colored}(N) \leftarrow \text{pickedCol}(N, C).
\]
\[
\leftarrow \text{node}(N) \land \neg \text{colored}(N).
\]
\[
\leftarrow \text{node}(N) \land \text{pickedCol}(N, C1) \land \text{pickedCol}(N, C2) \land C1 \neq C2.
\]
\[
\leftarrow \text{edge}(N1, N2) \land \text{pickedCol}(N1, C) \land \text{pickedCol}(N2, C).
\]

• 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)

\[
\{ \text{pickedCol}(N, C) \} \leftarrow \text{node}(N) \land \text{color}(C).
\]

\[
\text{color(red). color(blue). color(green). color(yellow).}
\]

\[
\text{node(1). node(2).}
\]

\[
\{ \text{pickedCol}(1, \text{red}) \} \leftarrow \text{node}(1) \land \text{color(red)}.
\]

\[
\{ \text{pickedCol}(1, \text{green}) \} \leftarrow \text{node}(1) \land \text{color(green)}.
\]

\[
\{ \text{pickedCol}(1, \text{blue}) \} \leftarrow \text{node}(1) \land \text{color(blue)}.
\]

\[
\{ \text{pickedCol}(1, \text{yellow}) \} \leftarrow \text{node}(1) \land \text{color(yellow)}.
\]

\[
\{ \text{pickedCol}(2, \text{red}) \} \leftarrow \text{node}(2) \land \text{color(red)}.
\]

\[
\ldots
\]

\[
\{ \text{pickedCol}(2, \text{yellow}) \} \leftarrow \text{node}(2) \land \text{color(yellow)}.
\]
ASP Evaluation

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

**Example (Grounding)**

\[
\{ \text{pickedCol}(N, C) \} \leftarrow \text{node}(N) \land \text{color}(C).
\]

\begin{align*}
\text{color}(\text{red}). \quad \text{color}(\text{blue}). \quad \text{color}(\text{green}). \quad \text{color}(\text{yellow}). \\
\text{node}(1). \quad \text{node}(2). \\
\{ \text{pickedCol}(1, \text{red}) \} \leftarrow \text{node}(1) \land \text{color}(\text{red}). \\
\{ \text{pickedCol}(1, \text{green}) \} \leftarrow \text{node}(1) \land \text{color}(\text{green}). \\
\{ \text{pickedCol}(1, \text{blue}) \} \leftarrow \text{node}(1) \land \text{color}(\text{blue}). \\
\{ \text{pickedCol}(1, \text{yellow}) \} \leftarrow \text{node}(1) \land \text{color}(\text{yellow}). \\
\{ \text{pickedCol}(2, \text{red}) \} \leftarrow \text{node}(2) \land \text{color}(\text{red}). \\
\vdots \\
\{ \text{pickedCol}(2, \text{yellow}) \} \leftarrow \text{node}(2) \land \text{color}(\text{yellow}).
\end{align*}
Traditional two-step evaluation: ground-and-solve.
- Grounding: replace variables by ground terms.
- Solving: mainly SAT techniques.

**Example (Grounding)**

\[
\{\text{pickedCol}(N, C)\} \leftarrow \text{node}(N) \land \text{color}(C).
\]

\[
\text{color(} \text{red}\text{)}. \text{color(} \text{blue}\text{)}. \text{color(} \text{green}\text{)}. \text{color(} \text{yellow}\text{)}.
\]

\[
\text{node(}1\text{)}. \text{node(}2\text{)}.
\]

\[
\{\text{pickedCol}(1, \text{red})\} \leftarrow \text{node}(1) \land \text{color(} \text{red}\text{)}.
\]

\[
\{\text{pickedCol}(1, \text{green})\} \leftarrow \text{node}(1) \land \text{color(} \text{green}\text{)}.
\]

\[
\{\text{pickedCol}(1, \text{blue})\} \leftarrow \text{node}(1) \land \text{color(} \text{blue}\text{)}.
\]

\[
\{\text{pickedCol}(1, \text{yellow})\} \leftarrow \text{node}(1) \land \text{color(} \text{yellow}\text{)}.
\]

\[
\{\text{pickedCol}(2, \text{red})\} \leftarrow \text{node}(2) \land \text{color(} \text{red}\text{)}.
\]

\[
\vdots
\]

\[
\{\text{pickedCol}(2, \text{yellow})\} \leftarrow \text{node}(2) \land \text{color(} \text{yellow}\text{)}.
\]
• Traditional two-step evaluation: ground-and-solve.
• Grounding: replace variables by ground terms.
• Solving: mainly SAT techniques.

**Example (Grounding)**

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


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)

\[
\{ \text{pickedCol}(N, C) \} \leftarrow \text{node}(N) \land \text{color}(C).
\]

\[
\text{color}(\text{red}). \text{color}(\text{blue}). \text{color}(\text{green}). \text{color}(\text{yellow}).
\]

\[
\text{node}(1). \text{node}(2).
\]

\[
\{ \text{pickedCol}(1, \text{red}) \} \leftarrow \text{node}(1) \land \text{color}(\text{red}).
\]

\[
\{ \text{pickedCol}(1, \text{green}) \} \leftarrow \text{node}(1) \land \text{color}(\text{green}).
\]

\[
\{ \text{pickedCol}(1, \text{blue}) \} \leftarrow \text{node}(1) \land \text{color}(\text{blue}).
\]

\[
\{ \text{pickedCol}(1, \text{yellow}) \} \leftarrow \text{node}(1) \land \text{color}(\text{yellow}).
\]

\[
\{ \text{pickedCol}(2, \text{red}) \} \leftarrow \text{node}(2) \land \text{color}(\text{red}).
\]

\[
\vdots
\]

\[
\{ \text{pickedCol}(2, \text{yellow}) \} \leftarrow \text{node}(2) \land \text{color}(\text{yellow}).
\]
• Traditional two-step evaluation: ground-and-solve.
• Grounding: replace variables by ground terms. (exponential!)
• Solving: mainly SAT techniques.

Example (Grounding)

\[
\{ \text{pickedCol}(N, C) \} \leftarrow \text{node}(N) \land \text{color}(C).
\]

\[
\text{color}(\text{red}). \; \text{color}(\text{blue}). \; \text{color}(\text{green}). \; \text{color}(\text{yellow}). \\
\text{node}(1). \; \text{node}(2).
\]

\[
\{ \text{pickedCol}(1, \text{red}) \} \leftarrow \text{node}(1) \land \text{color}(\text{red}).
\]

\[
\{ \text{pickedCol}(1, \text{green}) \} \leftarrow \text{node}(1) \land \text{color}(\text{green}).
\]

\[
\{ \text{pickedCol}(1, \text{blue}) \} \leftarrow \text{node}(1) \land \text{color}(\text{blue}).
\]

\[
\{ \text{pickedCol}(1, \text{yellow}) \} \leftarrow \text{node}(1) \land \text{color}(\text{yellow}).
\]

\[
\{ \text{pickedCol}(2, \text{red}) \} \leftarrow \text{node}(2) \land \text{color}(\text{red}).
\]

\[
\{ \text{pickedCol}(2, \text{yellow}) \} \leftarrow \text{node}(2) \land \text{color}(\text{yellow}).
\]
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, analyze 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, analyze 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.
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.

- ⇒ Solver cannot backjump and revert the wrong guess.
- ⇒ Chronological backtracking, exponential time overhead.
Example (Graph Coloring, again)

If \textit{colored}(2) is true but not \textit{justified}, what caused it?

\[
\textit{colored}(N) \leftarrow \textit{pickedCol}(N, C).
\]

\[
\leftarrow \textit{node}(N) \land \neg \textit{colored}(N).
\]

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

- ⇒ Solver cannot backjump and revert the wrong guess.
- ⇒ Chronological backtracking, \textit{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**

\[\text{colored}(N) \leftarrow \text{pickedCol}(N, C).\]

- $\neg \text{pickedColor}(2, \text{red})$
- $\neg \text{pickedColor}(2, \text{blue})$
- $\neg \text{colored}(2) \rightarrow \neg \text{pickedColor}(2, \text{green})$
- $\neg \text{pickedColor}(2, \text{yellow})$
• **Justification** \( J \) for \( \neg p \) explains for each rule that could derive \( p \), why it does not fire in interpretation \( I \).

**Example**

\[
\text{colored}(N) \leftarrow \text{pickedCol}(N, C).
\]

\[
\neg \text{pickedColor}(2, \text{red})
\]

\[
\neg \text{pickedColor}(2, \text{blue})
\]

\[
\neg \text{colored}(2)
\]

\[
\neg \text{pickedColor}(2, \text{green})
\]

\[
\neg \text{pickedColor}(2, \text{yellow})
\]
Theorem

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

- Problem: justifications consider ground rules.

$\implies$ Lift justifications.

Example
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

\[
\begin{align*}
\neg r & \\
\neg p(1) & \rightarrow \neg p(2) \rightarrow \ldots \rightarrow \neg q(1) \rightarrow \neg s(1) \rightarrow ns(1) \\

\neg p(1) & \rightarrow \neg t(4) \rightarrow \neg t(5) \rightarrow \ldots \rightarrow \neg q(3) \rightarrow \neg s(3) \rightarrow ns(3)
\end{align*}
\]
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

$\neg r$

$\downarrow$

$\neg p(X)(X \in C) \quad \rightarrow \quad \neg q(1) \quad \rightarrow \quad \neg s(1) \quad \rightarrow \quad ns(1)$

$\rightarrow$

$\rightarrow$

$\rightarrow$

$\neg q(2) \quad \rightarrow \quad \neg s(2) \quad \rightarrow \quad ns(2)$

$\rightarrow$

$\rightarrow$

$\rightarrow$

$\neg t(X)(X \in C \setminus \{1..3\}) \quad \rightarrow \quad \neg q(3) \quad \rightarrow \quad \neg s(3) \quad \rightarrow \quad ns(3)$
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: $\neg p \lor \bigvee_{\ell \in L} \ell$

Theorem

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

- Add clause $\Rightarrow$ standard conflict analysis does backjumping.
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: $\neg p \lor \bigvee_{\ell \in L} \ell$

Theorem

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

- Add clause $\Rightarrow$ standard conflict analysis does backjumping.
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: $\neg p \lor \bigvee_{\ell \in L} \ell$

Theorem

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

- Add clause $\Rightarrow$ standard conflict analysis does backjumping.
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: $\neg p \lor \bigvee_{\ell \in L} \ell$

Theorem

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

- Add clause $\Rightarrow$ standard conflict analysis does backjumping.
<table>
<thead>
<tr>
<th>Size</th>
<th>Alpha</th>
<th>$\text{Alpha}_J$</th>
<th>Clingo</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.81</td>
<td>0.79</td>
<td>0.00</td>
</tr>
<tr>
<td>20</td>
<td>2.55</td>
<td>0.81</td>
<td>0.00</td>
</tr>
<tr>
<td>30</td>
<td>300.00(5)</td>
<td>0.85</td>
<td>0.00</td>
</tr>
<tr>
<td>40</td>
<td>300.00(5)</td>
<td>0.92</td>
<td>0.00</td>
</tr>
<tr>
<td>50</td>
<td>300.00(5)</td>
<td>0.90</td>
<td>0.00</td>
</tr>
<tr>
<td>65</td>
<td>300.00(5)</td>
<td>0.86</td>
<td>0.00</td>
</tr>
<tr>
<td>100</td>
<td>300.00(5)</td>
<td>1.02</td>
<td>0.00</td>
</tr>
<tr>
<td>200</td>
<td>300.00(5)</td>
<td>1.04</td>
<td>0.01</td>
</tr>
<tr>
<td>400</td>
<td>300.00(5)</td>
<td>1.23</td>
<td>0.01</td>
</tr>
<tr>
<td>1000</td>
<td>300.00(5)</td>
<td>1.56</td>
<td>0.01</td>
</tr>
</tbody>
</table>

**Table 1:** Benchmark results for Two-way-derivation. Runtime is in seconds, timeouts in parentheses.
### Evaluation (2)

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

**Table 2:** Benchmark results for Graph-5-coloring. Runtime in seconds, timeouts in parentheses.
### Evaluation (3)

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

**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.