Towards Comprehensible ASP Reasoning by Means of Abstraction

Zeynep G. Saribatur

Institute of Logic and Computation
TU Wien, Vienna, Austria

XLoKR @ KR 2020
Answer Set Programming (ASP)

- A knowledge representation and reasoning paradigm widely used in problem solving.
  - Non-monotonic semantics, expressive power
  - Availability of efficient solvers (e.g., Clingo, WASP, DLV)
- Applications in many areas of AI including planning, diagnosis and commonsense reasoning.
- A convenient tool for investigating ways of applying human-inspired problem solving methods.
Background: ASP

- Syntax: rules of form

\[ \alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m, \text{not } \alpha_{m+1}, \ldots, \text{not } \alpha_n, \text{ for } 0 \leq m \leq n \]

Semantics:
- Herbrand universe: constant symbols
- Herbrand base: ground literals

Stable models [Gelfond and Lifschitz 1991]

Example: Graph coloring

\[
\begin{align*}
\text{color}(\text{red}). \\
\text{color}(\text{green}). \\
\text{color}(\text{blue}).
\end{align*}
\]

\[
\begin{align*}
\{ & \text{chosenColor}(N, C) \} \leftarrow \text{node}(N), \\
\text{colored}(N) \leftarrow \text{chosenColor}(N, C) . \\
\perp \leftarrow \text{not colored}(N), \text{node}(N) . \\
\perp \leftarrow \text{chosenColor}(N, C_1), \text{chosenColor}(N, C_2), C_1 \neq C_2 . \\
\perp \leftarrow \text{chosenColor}(N_1, C), \text{chosenColor}(N_2, C), \text{edge}(N_1, N_2) .
\end{align*}
\]

Answer sets:

\[
\{ \text{chosenColor}(1, \text{blue}), \text{chosenColor}(2, \text{green}), \text{chosenColor}(3, \text{red}), \ldots \}
\]
Background: ASP

- Syntax: rules of form
  \[ \alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m, \text{not } \alpha_{m+1}, \ldots, \text{not } \alpha_n, \text{ for } 0 \leq m \leq n \]

Example: Graph coloring

\texttt{color(red). color(green). color(blue). node(1...6).}
\texttt{\{chosenColor(N, C)\} \leftarrow node(N), color(C).}
\texttt{colored(N) \leftarrow chosenColor(N, C).}
\texttt{\bot \leftarrow \text{not colored}(N), node(N).}
\texttt{\bot \leftarrow chosenColor(N, C_1), chosenColor(N, C_2), C_1 \neq C_2.}
\texttt{\bot \leftarrow chosenColor(N_1, C), chosenColor(N_2, C), edge(N_1, N_2).}
Background: ASP

- Syntax: rules of form
  \[ \alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m, not \alpha_{m+1}, \ldots, not \alpha_n, \quad \text{for } 0 \leq m \leq n \]

- Semantics:
  - Herbrand universe: constant symbols, Herbrand base: ground literals
  - Stable models [Gelfond and Lifschitz 1991]

Example: Graph coloring

\[
\text{color(red). color(green). color(blue). node(1...6).}
\]
\[
\{\text{chosenColor}(N, C)\} \leftarrow \text{node}(N), \text{color}(C).
\]
\[
\text{colored}(N) \leftarrow \text{chosenColor}(N, C).
\]
\[
\bot \leftarrow \text{not colored}(N), \text{node}(N).
\]
\[
\bot \leftarrow \text{chosenColor}(N, C_1), \text{chosenColor}(N, C_2), C_1 \neq C_2.
\]
\[
\bot \leftarrow \text{chosenColor}(N_1, C), \text{chosenColor}(N_2, C), \text{edge}(N_1, N_2).
\]
Background: ASP

- Syntax: rules of form
  \[ \alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m, \text{not } \alpha_{m+1}, \ldots, \text{not } \alpha_n, \text{ for } 0 \leq m \leq n \]

- Semantics:
  - Herbrand universe: constant symbols, Herbrand base: ground literals
  - Stable models [Gelfond and Lifschitz 1991]

Example: Graph coloring

```prolog
color(red). color(green). color(blue). node(1...6).

\{chosenColor(N, C)\} \leftarrow node(N), color(C).
colored(N) \leftarrow chosenColor(N, C).
\bot \leftarrow \text{not} \ colored(N), node(N).
\bot \leftarrow chosenColor(N, C_1), chosenColor(N, C_2), C_1 \neq C_2.
\bot \leftarrow chosenColor(N_1, C), chosenColor(N_2, C), edge(N_1, N_2).
```

Answer sets:

\{chosenColor(1, blue), chosenColor(2, green), chosenColor(3, red), \ldots \}
Background: ASP

- Syntax: rules of form
  \[ \alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m, \text{not } \alpha_{m+1}, \ldots, \text{not } \alpha_n, \text{ for } 0 \leq m \leq n \]

- Semantics:
  - Herbrand universe: constant symbols, Herbrand base: ground literals
  - Stable models [Gelfond and Lifschitz 1991]

Example: Graph coloring

```
color(red). color(green). color(blue). node(1 \ldots 8).
{chosenColor(N, C)} \leftarrow node(N), color(C).
colored(N) \leftarrow chosenColor(N, C).
\bot \leftarrow \text{not colored}(N), node(N).
\bot \leftarrow chosenColor(N, C_1), chosenColor(N, C_2), C_1 \neq C_2.
\bot \leftarrow chosenColor(N_1, C), chosenColor(N_2, C), edge(N_1, N_2).
```

No answer sets.
Possibilities of Comprehensible ASP

- **Simplifications**

- **Verifying ASP programs** [Lifschitz et al. 2020]

- **Explanations** (see survey [Fandinno and Schulz 2019])
  - Debugging [Brain et al. 2007, Gebser et al. 2008, Oetsch et al. 2010]
Possibilities of Comprehensible ASP

- Simplifications
- Verifying ASP programs [Lifschitz et al. 2020]
- Explanations (see survey [Fandinno and Schulz 2019])
  - Debugging [Brain et al. 2007, Gebser et al. 2008, Oetsch et al. 2010]

→ The obtained explanations may contain too many details which prevent one from seeing the crucial parts.
Abstraction in KR

- It is commonly agreed that abstraction plays a key role in representing knowledge and in reasoning
  - By focusing on the key details and disregarding the rest
- Solve the problem in the abstract space, then guide the search for an original solution [Newell and Simon 1972, Sacerdoti 1974]
  - Abstraction heuristics [Edelkamp 2001, Helmert et al. 2007]
  - Hierarchical planning [Bercher et al. 2019]
- Desired properties for abstractions
  [Giunchiglia and Walsh 1992, Nayak and Levy 1995]
- Abstraction layers in ASP-related languages for robotics
  [Zhang et al. 2015, Sridharan et al. 2019]
Abstraction in ASP

- Constructing an over-approximation of a given program
  - through omitting atoms from the vocabulary, or
  - clustering the elements of the domain.
- CEGAR-inspired abstraction-&-refinement methodology
  - Automatically finding an abstraction that gives concrete solutions.
- Implemented prototypical tools and applied to several benchmarks.
- Resulting abstractions can be used to get an understanding of the problem at hand.
Outline

1. Background: Abstraction in ASP
2. Potential in Comprehensibility
   - Removing Irrelevant Details
   - Generalization
3. Conclusion
Outline

1. Background: Abstraction in ASP

2. Potential in Comprehensibility
   - Removing Irrelevant Details
   - Generalization

3. Conclusion
Abstraction

- Over-approximate the problem into a smaller or simpler state space.
  - Deliberately lose information.

All original transitions must be preserved. Spurious transitions may be introduced. Refinement of the abstraction is necessary in order to get rid of spurious transitions. E.g., CEGAR method [Clarke et al. 2003]
Abstraction in Model Checking

- Over-approximate the problem into a smaller or simpler state space.
  - Deliberately lose information.
- All original transitions must be preserved.
  - The abstract system can simulate the original system.
- Spurious transitions may be introduced.
Abstraction in Model Checking

- Over-approximate the problem into a smaller or simpler state space.
  - Deliberately lose information.
- All original transitions must be preserved.
  - The abstract system can simulate the original system.
- Spurious transitions may be introduced.
Abstraction in Model Checking

- Over-approximate the problem into a smaller or simpler state space.
  - Deliberately lose information.
- All original transitions must be preserved.
  - The abstract system can simulate the original system.
- Spurious transitions may be introduced.

- Refinement of the abstraction is necessary in order to get rid of spurious transitions.
  - E.g., CEGAR method [Clarke et al. 2003]
Abstraction in Model Checking

- Over-approximate the problem into a smaller or simpler state space.
  - Deliberately lose information.
- All original transitions must be preserved.
  - The abstract system can simulate the original system.
- Spurious transitions may be introduced.

- Refinement of the abstraction is necessary in order to get rid of spurious transitions.
  - E.g., CEGAR method [Clarke et al. 2003]
Abstraction in Model Checking

- **Over-approximate** the problem into a smaller or simpler state space.
  - Deliberately lose information.
- All original transitions must be preserved.
  - The abstract system can simulate the original system.
- **Spurious** transitions may be introduced.

- **Refinement** of the abstraction is necessary in order to get rid of spurious transitions.
  - E.g., CEGAR method [Clarke et al. 2003]
Abstraction in Model Checking

- Over-approximate the problem into a smaller or simpler state space.
  - Deliberately lose information.
- All original transitions must be preserved.
  - The abstract system can simulate the original system.
- Spurious transitions may be introduced.

- Refinement of the abstraction is necessary in order to get rid of spurious transitions.
  - E.g., CEGAR method [Clarke et al. 2003]
Abstraction in Model Checking

- Over-approximate the problem into a smaller or simpler state space.
  - Deliberately lose information.
- All original transitions must be preserved.
  - The abstract system can simulate the original system.
- Spurious transitions may be introduced.

- Refinement of the abstraction is necessary in order to get rid of spurious transitions.
  - E.g., CEGAR method [Clarke et al. 2003]
Abstraction of Answer Set Programs

**Definition**

$\Pi'$ is an abstraction of $\Pi$, where $|A| \geq |A'|$, if there exists a mapping $m : A \rightarrow A' \cup \{\top\}$ s.t. for any answer set $I$ of $\Pi$, $I' = \{m(\alpha) | \alpha \in I\}$ is an answer set of $\Pi'$.
Abstraction of Answer Set Programs

**Definition**

$\Pi'$ is an abstraction of $\Pi$, where $|A| \geq |A'|$, if there exists a mapping $m : A \rightarrow A' \cup \{\top\}$ such that for any answer set $I$ of $\Pi$, $I' = \{m(\alpha) \mid \alpha \in I\}$ is an answer set of $\Pi'$.

$$\text{AS}(\Pi') = \{\ldots I_k' \ldots I_n' \ldots \}$$

$$\text{AS}(\Pi) = \{I_1 \ I_2 \ \ldots \ \ I_n \}$$
Abstraction of Answer Set Programs

**Definition**

$\Pi'$ is an abstraction of $\Pi$, where $|A| \geq |A'|$, if there exists a mapping $m : A \rightarrow A' \cup \{\top\}$ s.t. for any answer set $I$ of $\Pi$, $I' = \{m(\alpha) \mid \alpha \in I\}$ is an answer set of $\Pi'$.

\[
\begin{align*}
\text{AS}(\Pi') &= \{... \ I_k' \ ... \ I_n' \ ... \} \\
\text{AS}(\Pi) &= \{I_1 \ I_2 \ ... \ I_n \}
\end{align*}
\]

- Size of $I'$ $\leq$ size of $I$.
- **Spurious** $I'$ may exist that cannot be mapped back to some $I$.
**Definition**

$\Pi'$ is an abstraction of $\Pi$, where $|A| \geq |A'|$, if there exists a mapping $m : A \rightarrow A' \cup \{\top\}$ s.t. for any answer set $I$ of $\Pi$, $I' = \{m(\alpha) \mid \alpha \in I\}$ is an answer set of $\Pi'$.

- Size of $I' \leq$ size of $I$.
- **Spurious** $I'$ may exist that cannot be mapped back to some $I$.

- $\Pi'$ is faithful if it has no spurious answer sets $\rightarrow m(AS(\Pi)) = AS(\Pi')$
Abstraction of Answer Set Programs

**Definition**

$\Pi'$ is an abstraction of $\Pi$, where $|\mathcal{A}| \geq |\mathcal{A}'|$, if there exists a mapping $m : \mathcal{A} \rightarrow \mathcal{A}' \cup \{\top\}$ s.t. for any answer set $I$ of $\Pi$, $I' = \{m(\alpha) \mid \alpha \in I\}$ is an answer set of $\Pi'$.

\[
\begin{align*}
\text{AS}(\Pi') &= \emptyset \\
\downarrow \\
\text{AS}(\Pi) &= \emptyset
\end{align*}
\]

- Size of $I'$ ≤ size of $I$.
- **Spurious** $I'$ may exist that cannot be mapped back to some $I$.
- If no $I'$ exists in $\Pi'$, then no $I$ exists in $\Pi$.

$\Pi'$ is **faithful** if it has no spurious answer sets $\rightarrow m(\text{AS}(\Pi)) = \text{AS}(\Pi')$
Example: Graph Coloring

- A non-3-colorable graph
Example: Graph Coloring

- A non-3-colorable graph
Example: Graph Coloring

- A non-3-colorable graph

![Non-3-colorable graph](image)

- 3-coloring of a graph

![3-coloring of a graph](image)
Example: Graph Coloring

- A non-3-colorable graph

- 3-coloring of a graph

\[ a_4 = \{4, 5, 6\} \]
Example: Graph Coloring

- A non-3-colorable graph

- 3-coloring of a graph

\[ a_4 = \{4, 5, 6\} \]
Example: Graph Coloring

- A non-3-colorable graph

→ Omission abstraction
[Saribatur and Eiter 2020]

3-coloring of a graph

→ Domain abstraction
[Saribatur et al. 2019]
Abstraction and Refinement Methodology

Input: $\Pi, m$

Construct $\Pi^m$

Output: $\Pi^m, m$

Get $I$ from $\text{AS}(\Pi^m)$

$I$ is spurious?

Refine $m$

Yes

No

No

Yes
Abstraction and Refinement Methodology

Input: $\Pi, m$

Construct $\Pi^m$

Get $I$ from $\text{AS}(\Pi^m)$

$I$ is spurious?

Refine $m$

Output: $\Pi^m, m, I$

Output: $\Pi^m, m$
Example: Graph Coloring

- Adding back omitted nodes

- Dividing abstract node clusters
Example: Graph Coloring

- Adding back omitted nodes

- Dividing abstract node clusters
Example: Graph Coloring

- Adding back omitted nodes

- Dividing abstract node clusters
Example: Graph Coloring

- **Adding back omitted nodes**

  ![Graph Coloring Example](image)

- **Dividing abstract node clusters**

  ![Graph Coloring Example](image)
Abstraction and Refinement Methodology

Input: $\Pi, m$

Construct $\Pi^m$

$AS(\Pi^m) \neq \emptyset$?

yes

no

Output: $\Pi^m, m$

Refine $m$

Get $I$ from $AS(\Pi^m)$

$I$ is spurious?

yes

no

Output: $\Pi^m, m, I$
Outline

1. Background: Abstraction in ASP

2. Potential in Comprehensibility
   - Removing Irrelevant Details
   - Generalization

3. Conclusion
How do Humans use Abstraction?

- **Abstract Thinking**: Removing irrelevant details and identifying the “essence” of the problem [Johnson-Laird 1983, Kramer 2007]
- Approaches to tackle large and complex structures
  - Determining the relevance
  - Distinguishing the common properties among the objects
Outline

1. Background: Abstraction in ASP

2. Potential in Comprehensibility
   - Removing Irrelevant Details
   - Generalization

3. Conclusion
Removing Irrelevant Details

Satisfiability blockers

- Focusing on omission abstraction
- \( C \subseteq A \) is an (answer set) blocker set of \( \Pi \), if \( AS(\text{omit}(\Pi, A \setminus C)) = \emptyset \)
- \( \subseteq \)-minimal blocker set \( \rightarrow \) most relevant part of the unsat. program
Focusing on omission abstraction

- $C \subseteq A$ is an (answer set) blocker set of $\Pi$, if $AS(omit(\Pi, A \setminus C)) = \emptyset$
- $\subseteq$-minimal blocker set $\rightarrow$ most relevant part of the unsat. program
Removing Irrelevant Details

Satisfiability blockers

<table>
<thead>
<tr>
<th>Initial State</th>
<th>Goal State</th>
</tr>
</thead>
<tbody>
<tr>
<td>* * *</td>
<td>* * *</td>
</tr>
<tr>
<td>* * *</td>
<td>* * *</td>
</tr>
<tr>
<td>* * *</td>
<td>* * *</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial State</th>
<th>Goal State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 3 *</td>
<td>0 * *</td>
</tr>
<tr>
<td>* 5 4</td>
<td>3 4 5</td>
</tr>
<tr>
<td>* * *</td>
<td>* * *</td>
</tr>
</tbody>
</table>

$m_{\text{init}}$ $m_{\text{final}}$

- Focusing on omission abstraction
- $C \subseteq A$ is an (answer set) blocker set of $\Pi$, if $AS(\text{omit}(\Pi, A \setminus C)) = \emptyset$
- $\subseteq$-minimal blocker set $\rightarrow$ most relevant part of the unsat. program
Removing Irrelevant Details

Satisfiability blockers

Initial State

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Goal State

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Initial State

<table>
<thead>
<tr>
<th>0</th>
<th>3</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Goal State

<table>
<thead>
<tr>
<th>0</th>
<th>*</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

\[ m_{\text{init}} \quad \text{and} \quad m_{\text{final}} \]

- Focusing on omission abstraction
- \( C \subseteq A \) is an (answer set) blocker set of \( \Pi \), if \( AS(\text{omit}(\Pi, A \setminus C)) = \emptyset \)
- \( \subseteq \)-minimal blocker set \( \rightarrow \) most relevant part of the unsat. program
- E.g., explain non-acceptability of arguments in abstract argumentation
  
  [Saribatur et al. 2020]
Removing Irrelevant Details
Finding only relevant details for solvability

- Focusing on domain abstraction
- E.g., distinguishing the key time slots in scheduling
Removing Irrelevant Details

Finding only relevant details for solvability

- Focusing on domain abstraction
- E.g., distinguishing the key time slots in scheduling
Removing Irrelevant Details
Finding only relevant details for unsolvability

- Focusing on multi-dimensional domain abstraction [Eiter et al. 2019]
  - Achieve a hierarchical abstraction over the domain for zooming-in

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & A & & \\
2 & & & \\
3 & B & & \\
4 & & & \\
\end{array} \quad \begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & A & & \\
2 & & & \\
3 & B & & \\
4 & & & \\
\end{array} \quad \begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & A & & \\
2 & & & \\
3 & B & & \\
4 & & & \\
\end{array} \]
Removing Irrelevant Details
Finding only relevant details for unsolvability

- Focusing on multi-dimensional domain abstraction [Eiter et al. 2019]
  - Achieve a hierarchical abstraction over the domain for zooming-in

- Can the automatically obtained abstractions match the intuition behind a human explanation to unsolvability?
User Study on Unsatisfiable Grid-cell Problems

- **Reachability**: Mark the area which shows *the reason for having unreachable cells*

- **Visitall**: Mark the area which shows *the reason for not finding a solution* that visits all the cells
User Study Results for Reachability and Visitall (1/2)

(a) #6 : expected
(b) #6 : unexpected
(c) #6 - mDASPAR

(d) #1: expected
(f) #1 - mDASPAR
User Study Results for Reachability and Visitall (1/2)

(a) #6: expected
(b) #6: unexpected
(c) #6 - mDASPAR

(d) #1: expected
(e) #1: unexpected
(f) #1 - mDASPAR
User Study Results for Reachability and Visitall (2/2)

(a) #10: expected

(d) #10: expected

(c) #10 - mDASPAR

(f) #10 - mDASPAR
User Study Results for Reachability and Visitall (2/2)

(a) #10: expected

(b) #10: unexpected

(c) #10 - mDASPAR

(d) #10: expected

(e) #10: unexpected

(f) #10 - mDASPAR
Outline

1 Background: Abstraction in ASP

2 Potential in Comprehensibility
   - Removing Irrelevant Details
   - Generalization

3 Conclusion
Generalization

- Make use of **faithful abstractions** to reason at the abstract level by distinguishing the common properties of the clustered elements.
Make use of **faithful abstractions** to reason at the abstract level by distinguishing the common properties of the clustered elements.
Make use of **faithful abstractions** to reason at the abstract level by distinguishing the common properties of the clustered elements.
Make use of faithful abstractions to reason at the abstract level by distinguishing the common properties of the clustered elements.

Relation to generalized planning [Srivastava et al. 2011], [Illanes and McIlraith 2019] remains to be investigated.
Outline

1. Background: Abstraction in ASP

2. Potential in Comprehensibility
   - Removing Irrelevant Details
   - Generalization

3. Conclusion
Conclusion

- Abstraction shows potential in finding the “essense” of problem solving in ASP, useful for human-comprehensibility.
- We have an automated way of starting with an initial abstraction and achieving an abstraction with a concrete answer.
- Demonstrates a human-like focus to the key elements in the problem.
- Can be used as a guide to decide on good abstractions for reasoning.

Challenges:
- Finding ways to make use of such abstractions to help the users in understanding the decision-making behavior.
- Achieving the various levels of abstraction that humans unwittingly use.


References II

Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith.
Counterexample-guided abstraction refinement for symbolic model checking.

Stefan Edelkamp.
Planning with pattern databases.

Thomas Eiter and Michael Fink.
Uniform equivalence of logic programs under the stable model semantics.


References IV

Jorge Fandinno and Claudia Schulz.
Answering the “why” in answer set programming - A survey of explanation approaches.

Martin Gebser, Jörg Pührer, Torsten Schaub, and Hans Tompits.
A meta-programming technique for debugging answer-set programs.

Michael Gelfond and Vladimir Lifschitz.
Classical negation in logic programs and disjunctive databases.

Fausto Giunchiglia and Toby Walsh.
A theory of abstraction.


Philip Nicholas Johnson-Laird.  
*Mental models: Towards a cognitive science of language, inference, and consciousness.*  

Jeff Kramer.  
*Is abstraction the key to computing?*  

João Leite.  
*A bird’s-eye view of forgetting in answer-set programming.*  


References VIII


Enrico Pontelli, Tran Cao Son, and Omar Elkhatib.  
Justifications for logic programs under answer set semantics.  

Earl D. Sacerdoti.  
Planning in a hierarchy of abstraction spaces.  

Zeynep G. Saribatur and Thomas Eiter.  
Omission-based abstraction for answer set programs.  
*TPLP*, 2020.  

