ASP Solving for Expanding Universes

Martin Gebser   Tomi Janhunen   Holger Jost
Roland Kaminski   Torsten Schaub

Aalto University

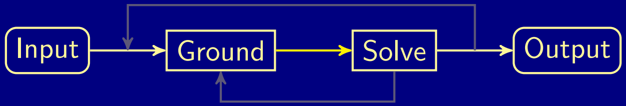
INRIA Rennes

University of Potsdam

Outline

Outline

Multi-shot Solving



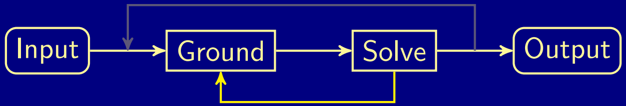
- Traditional ASP systems were devised for one-shot solving
- Modern ASP systems allow for multi-shot solving in a reactive way
- New properties or objects must be integrated dynamically
- Due to non-monotonicity, new information can invalidate conclusions

$$\begin{aligned} \mathit{flies}(X) &\leftarrow \mathit{bird}(X), {\sim}\mathit{penguin}(X) \\ \mathit{bird}(\mathit{tweety}) &\leftarrow \\ \mathit{penguin}(\mathit{tweety}) &\leftarrow \end{aligned}$$

$$\models \{\mathit{bird}(\mathit{tweety}), \neg\mathit{penguin}(\mathit{tweety}), \mathit{flies}(\mathit{tweety})\}$$

- General approach to integrate new information into reasoning process?

Multi-shot Solving



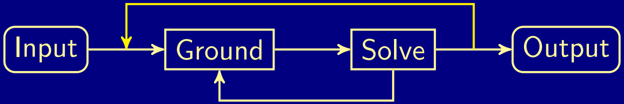
- Traditional ASP systems were devised for one-shot solving
- Modern ASP systems allow for multi-shot solving in a reactive way
- New properties or objects must be integrated dynamically
- Due to non-monotonicity, new information can invalidate conclusions

$$\begin{aligned} \mathit{flies}(X) &\leftarrow \mathit{bird}(X), {\sim}\mathit{penguin}(X) \\ \mathit{bird}(\mathit{tweety}) &\leftarrow \\ \mathit{penguin}(\mathit{tweety}) &\leftarrow \end{aligned}$$

$$\models \{\mathit{bird}(\mathit{tweety}), \neg\mathit{penguin}(\mathit{tweety}), \mathit{flies}(\mathit{tweety})\}$$

- General approach to integrate new information into reasoning process?

Multi-shot Solving



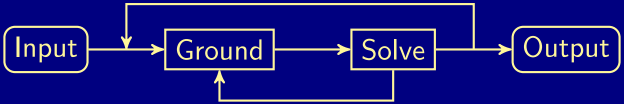
- Traditional ASP systems were devised for one-shot solving
- Modern ASP systems allow for multi-shot solving in a reactive way
- New properties or objects must be integrated dynamically
- Due to non-monotonicity, new information can invalidate conclusions

$$\begin{aligned} \mathit{flies}(X) &\leftarrow \mathit{bird}(X), {\sim}\mathit{penguin}(X) \\ \mathit{bird}(\mathit{tweety}) &\leftarrow \\ \mathit{penguin}(\mathit{tweety}) &\leftarrow \end{aligned}$$

$$\models \{\mathit{bird}(\mathit{tweety}), \neg\mathit{penguin}(\mathit{tweety}), \mathit{flies}(\mathit{tweety})\}$$

- General approach to integrate new information into reasoning process?

Multi-shot Solving



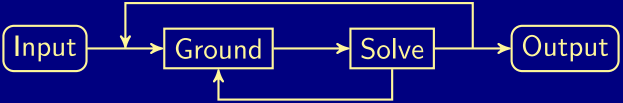
- Traditional ASP systems were devised for one-shot solving
- Modern ASP systems allow for multi-shot solving in a reactive way
- New properties or objects must be integrated dynamically
- Due to non-monotonicity, new information can invalidate conclusions

$$\begin{aligned} \mathit{flies}(X) &\leftarrow \mathit{bird}(X), \sim\mathit{penguin}(X) \\ \mathit{bird}(\mathit{tweety}) &\leftarrow \\ \mathit{penguin}(\mathit{tweety}) &\leftarrow \end{aligned}$$

$$\models \{\mathit{bird}(\mathit{tweety}), \neg\mathit{penguin}(\mathit{tweety}), \mathit{flies}(\mathit{tweety})\}$$

- General approach to integrate new information into reasoning process?

Multi-shot Solving



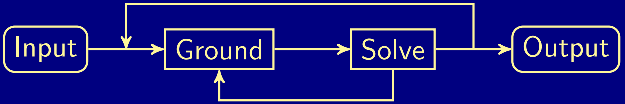
- Traditional ASP systems were devised for one-shot solving
- Modern ASP systems allow for multi-shot solving in a reactive way
- New properties or objects must be integrated dynamically
- Due to non-monotonicity, new information can invalidate conclusions

$$\begin{aligned} \mathit{flies}(\mathit{tweety}) &\leftrightarrow \mathit{bird}(\mathit{tweety}) \land \neg \mathit{penguin}(\mathit{tweety}) \\ \mathit{bird}(\mathit{tweety}) &\leftrightarrow \top \\ \mathit{penguin}(\mathit{tweety}) &\leftrightarrow \bot \end{aligned}$$

$$\models \{\mathit{bird}(\mathit{tweety}), \neg \mathit{penguin}(\mathit{tweety}), \mathit{flies}(\mathit{tweety})\}$$

- General approach to integrate new information into reasoning process?

Multi-shot Solving



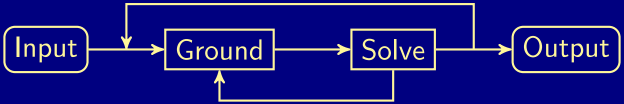
- Traditional ASP systems were devised for one-shot solving
- Modern ASP systems allow for multi-shot solving in a reactive way
- New properties or objects must be integrated dynamically
- Due to non-monotonicity, new information can invalidate conclusions

$$\begin{aligned} \mathit{flies}(X) &\leftarrow \mathit{bird}(X), {\sim}\mathit{penguin}(X) \\ \mathit{bird}(\mathit{tweety}) &\leftarrow \\ \mathit{penguin}(\mathit{tweety}) &\leftarrow \end{aligned}$$

$$\models \{\mathit{bird}(\mathit{tweety}), \neg\mathit{penguin}(\mathit{tweety}), \mathit{flies}(\mathit{tweety})\}$$

- General approach to integrate new information into reasoning process?

Multi-shot Solving



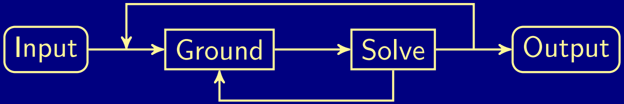
- Traditional ASP systems were devised for one-shot solving
- Modern ASP systems allow for multi-shot solving in a reactive way
- New properties or objects must be integrated dynamically
- Due to non-monotonicity, new information can invalidate conclusions

$$\begin{aligned} \mathit{flies}(X) &\leftarrow \mathit{bird}(X), {\sim}\mathit{penguin}(X) \\ \mathit{bird}(\mathit{tweety}) &\leftarrow \\ \mathit{penguin}(\mathit{tweety}) &\leftarrow \end{aligned}$$

$$\models \{\mathit{bird}(\mathit{tweety}), \neg\mathit{penguin}(\mathit{tweety}), \mathit{flies}(\mathit{tweety})\}$$

- General approach to integrate new information into reasoning process?

Multi-shot Solving



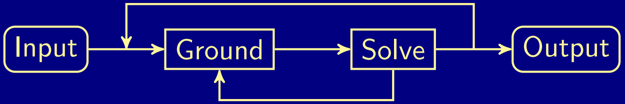
- Traditional ASP systems were devised for one-shot solving
- Modern ASP systems allow for multi-shot solving in a reactive way
- New properties or objects must be integrated dynamically
- Due to non-monotonicity, new information can invalidate conclusions

$$\begin{aligned} \mathit{flies}(\mathit{tweety}) &\leftrightarrow \mathit{bird}(\mathit{tweety}) \wedge \neg \mathit{penguin}(\mathit{tweety}) \\ \mathit{bird}(\mathit{tweety}) &\leftrightarrow \top \\ \mathit{penguin}(\mathit{tweety}) &\leftrightarrow \top \end{aligned}$$

$$\models \{\mathit{bird}(\mathit{tweety}), \mathit{penguin}(\mathit{tweety}), \neg \mathit{flies}(\mathit{tweety})\}$$

- General approach to integrate new information into reasoning process?

Multi-shot Solving



- Traditional ASP systems were devised for one-shot solving
- Modern ASP systems allow for multi-shot solving in a reactive way
- New properties or objects must be integrated dynamically
- Due to non-monotonicity, new information can invalidate conclusions

$$\begin{aligned} \mathit{flies}(\mathit{tweety}) &\leftrightarrow \mathit{bird}(\mathit{tweety}) \wedge \neg \mathit{penguin}(\mathit{tweety}) \\ \mathit{bird}(\mathit{tweety}) &\leftrightarrow \top \\ \mathit{penguin}(\mathit{tweety}) &\leftrightarrow \top \end{aligned}$$

$$\models \{\mathit{bird}(\mathit{tweety}), \mathit{penguin}(\mathit{tweety}), \neg \mathit{flies}(\mathit{tweety})\}$$

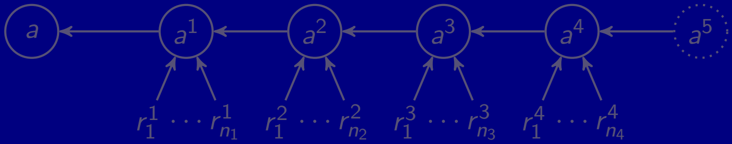
- General approach to integrate new information into reasoning process?

Basic Idea

- View arrival of new objects as addition of new constants
  - ➥ Successively expanding Herbrand universe
- New constants induce new ground instances of rules
  - ➥ Disjoint partition and modular composition of ground program
- ✗ New ground instances defining older atoms invalidate completion!

Contribution

- ✔ Translation approach guaranteeing modularity at level of completion
  - New ground instances of rules define new expansion atoms
  - Expansion atoms are interconnected to accumulate derivations
  - Accumulated derivations are propagated to original ground atoms

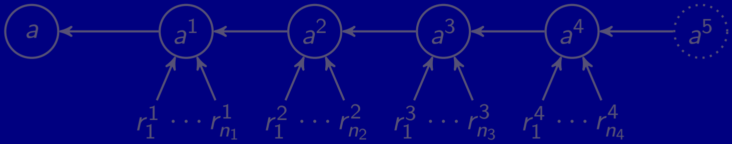


Basic Idea

- View arrival of new objects as addition of new constants
  - ➥ Successively expanding Herbrand universe
- New constants induce new ground instances of rules
  - ➥ Disjoint partition and modular composition of ground program
- ✘ New ground instances defining older atoms invalidate completion!

Contribution

- ✔ Translation approach guaranteeing modularity at level of completion
  - New ground instances of rules define new expansion atoms
  - Expansion atoms are interconnected to accumulate derivations
  - Accumulated derivations are propagated to original ground atoms

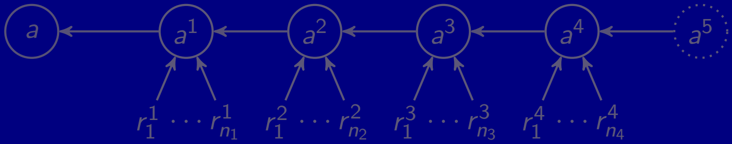


Basic Idea

- View arrival of new objects as addition of new constants
  - ➥ Successively expanding Herbrand universe
- New constants induce new ground instances of rules
  - ➥ Disjoint partition and modular composition of ground program
- ✘ New ground instances defining older atoms invalidate completion!

Contribution

- ✔ Translation approach guaranteeing modularity at level of completion
  - New ground instances of rules define new expansion atoms
  - Expansion atoms are interconnected to accumulate derivations
  - Accumulated derivations are propagated to original ground atoms



Basic Idea

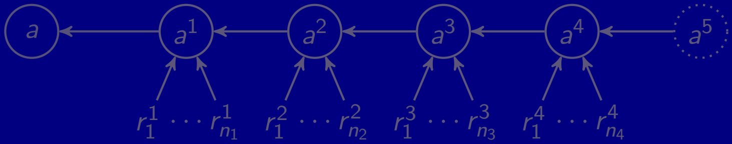
- View arrival of new objects as addition of new constants
  - ➥ Successively expanding Herbrand universe
- New constants induce new ground instances of rules
  - ➥ Disjoint partition and modular composition of ground program

✘ New ground instances defining older atoms invalidate completion!

Contribution

✔ Translation approach guaranteeing modularity at level of completion

1. New ground instances of rules define new expansion atoms
2. Expansion atoms are interconnected to accumulate derivations
3. Accumulated derivations are propagated to original ground atoms



Basic Idea

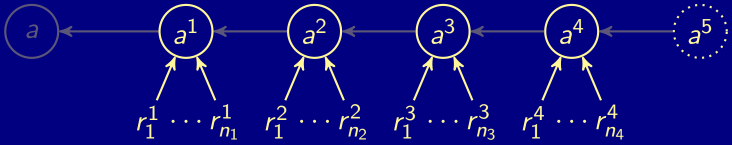
- View arrival of new objects as addition of new constants
  - ➥ Successively expanding Herbrand universe
- New constants induce new ground instances of rules
  - ➥ Disjoint partition and modular composition of ground program

✘ New ground instances defining older atoms invalidate completion!

Contribution

✔ Translation approach guaranteeing modularity at level of completion

1. New ground instances of rules define new expansion atoms
2. Expansion atoms are interconnected to accumulate derivations
3. Accumulated derivations are propagated to original ground atoms



Basic Idea

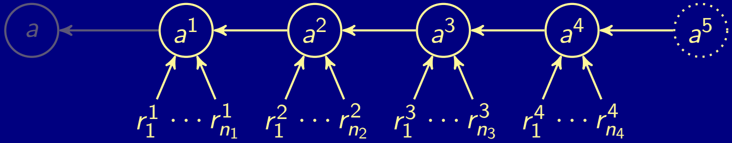
- View arrival of new objects as addition of new constants
  - ➥ Successively expanding Herbrand universe
- New constants induce new ground instances of rules
  - ➥ Disjoint partition and modular composition of ground program

✘ New ground instances defining older atoms invalidate completion!

Contribution

✔ Translation approach guaranteeing modularity at level of completion

1. New ground instances of rules define new expansion atoms
2. Expansion atoms are interconnected to accumulate derivations
3. Accumulated derivations are propagated to original ground atoms



Basic Idea

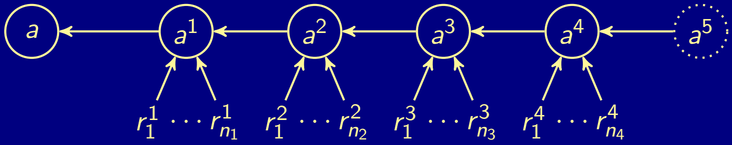
- View arrival of new objects as addition of new constants
  - ➥ Successively expanding Herbrand universe
- New constants induce new ground instances of rules
  - ➥ Disjoint partition and modular composition of ground program

✘ New ground instances defining older atoms invalidate completion!

Contribution

✔ Translation approach guaranteeing modularity at level of completion

1. New ground instances of rules define new expansion atoms
2. Expansion atoms are interconnected to accumulate derivations
3. Accumulated derivations are propagated to original ground atoms



Outline

Translation Approach

- Given a set R of rules, defining intensional predicates $\mathcal{P}_I$, let:

$$\Phi(R) = \{p^k(X_1, \ldots, X_n) \leftarrow B \mid (p(X_1, \ldots, X_n) \leftarrow B) \in R\},$$
$$\Pi(\mathcal{P}_I) = \{p(X_1, \ldots, X_n) \leftarrow p^k(X_1, \ldots, X_n) \mid p/n \in \mathcal{P}_I\},$$
$$\Delta(\mathcal{P}_I) = \{p^k(X_1, \ldots, X_n) \leftarrow p^{k+1}(X_1, \ldots, X_n) \mid p/n \in \mathcal{P}_I\}.$$

Example

$$R = \left\{ \begin{array}{l} ok(C) \leftarrow cs(C), st(S), in(S, C) \\ ko(C) \leftarrow cs(C), \sim ok(C) \end{array} \right\}$$

$$\Phi(R) = \left\{ \begin{array}{l} ok^k(C) \leftarrow cs(C), st(S), in(S, C) \\ ko^k(C) \leftarrow cs(C), \sim ok(C) \end{array} \right\}$$

$$\Pi(\mathcal{P}_I) = \left\{ \begin{array}{l} ok(C) \leftarrow ok^k(C) \\ ko(C) \leftarrow ko^k(C) \end{array} \right\} \qquad \Delta(\mathcal{P}_I) = \left\{ \begin{array}{l} ok^k(C) \leftarrow ok^{k+1}(C) \\ ko^k(C) \leftarrow ko^{k+1}(C) \end{array} \right\}$$

Translation Approach

- Given a set R of rules, defining intensional predicates $\mathcal{P}_I$, let:

$$\Phi(R) = \{p^k(X_1, \ldots, X_n) \leftarrow B \mid (p(X_1, \ldots, X_n) \leftarrow B) \in R\},$$
$$\Pi(\mathcal{P}_I) = \{p(X_1, \ldots, X_n) \leftarrow p^k(X_1, \ldots, X_n) \mid p/n \in \mathcal{P}_I\},$$
$$\Delta(\mathcal{P}_I) = \{p^k(X_1, \ldots, X_n) \leftarrow p^{k+1}(X_1, \ldots, X_n) \mid p/n \in \mathcal{P}_I\}.$$

Example

$$R = \left\{ \begin{array}{l} ok(C) \leftarrow cs(C), st(S), in(S, C) \\ ko(C) \leftarrow cs(C), \sim ok(C) \end{array} \right\}$$

$$\Phi(R) = \left\{ \begin{array}{l} ok^k(C) \leftarrow cs(C), st(S), in(S, C) \\ ko^k(C) \leftarrow cs(C), \sim ok(C) \end{array} \right\}$$

$$\Pi(\mathcal{P}_I) = \left\{ \begin{array}{l} ok(C) \leftarrow ok^k(C) \\ ko(C) \leftarrow ko^k(C) \end{array} \right\} \qquad \Delta(\mathcal{P}_I) = \left\{ \begin{array}{l} ok^k(C) \leftarrow ok^{k+1}(C) \\ ko^k(C) \leftarrow ko^{k+1}(C) \end{array} \right\}$$

Translation Approach

- Given a set R of rules, defining intensional predicates $\mathcal{P}_I$, let:

$$\Phi(R) = \{p^k(X_1,\dots,X_n) \leftarrow B \mid (p(X_1,\dots,X_n) \leftarrow B) \in R\},$$
$$\Pi(\mathcal{P}_I) = \{p(X_1,\dots,X_n) \leftarrow p^k(X_1,\dots,X_n) \mid p/n \in \mathcal{P}_I\},$$
$$\Delta(\mathcal{P}_I) = \{p^k(X_1,\dots,X_n) \leftarrow p^{k+1}(X_1,\dots,X_n) \mid p/n \in \mathcal{P}_I\}.$$

Example

$$R = \left\{ \begin{array}{l} ok(C) \leftarrow cs(C), st(S), in(S,C) \\ ko(C) \leftarrow cs(C), {\sim}ok(C) \end{array} \right\}$$

$$\Phi(R) = \left\{ \begin{array}{l} ok^k(C) \leftarrow cs(C), st(S), in(S,C) \\ ko^k(C) \leftarrow cs(C), {\sim}ok(C) \end{array} \right\}$$

$$\Pi(\mathcal{P}_I) = \left\{ \begin{array}{l} ok(C) \leftarrow ok^k(C) \\ ko(C) \leftarrow ko^k(C) \end{array} \right\} \qquad \Delta(\mathcal{P}_I) = \left\{ \begin{array}{l} ok^k(C) \leftarrow ok^{k+1}(C) \\ ko^k(C) \leftarrow ko^{k+1}(C) \end{array} \right\}$$

Translation Approach

- Given a set R of rules, defining intensional predicates $\mathcal{P}_I$, let:

$$\Phi(R) = \{p^k(X_1,\dots,X_n) \leftarrow B \mid (p(X_1,\dots,X_n) \leftarrow B) \in R\},$$
$$\Pi(\mathcal{P}_I) = \{p(X_1,\dots,X_n) \leftarrow p^k(X_1,\dots,X_n) \mid p/n \in \mathcal{P}_I\},$$
$$\Delta(\mathcal{P}_I) = \{p^k(X_1,\dots,X_n) \leftarrow p^{k+1}(X_1,\dots,X_n) \mid p/n \in \mathcal{P}_I\}.$$

Example

$$R = \left\{ \begin{array}{l} ok(C) \leftarrow cs(C), st(S), in(S,C) \\ ko(C) \leftarrow cs(C), {\sim}ok(C) \end{array} \right\}$$

$$\Phi(R) = \left\{ \begin{array}{l} ok^k(C) \leftarrow cs(C), st(S), in(S,C) \\ ko^k(C) \leftarrow cs(C), {\sim}ok(C) \end{array} \right\}$$

$$\Pi(\mathcal{P}_I) = \left\{ \begin{array}{l} ok(C) \leftarrow ok^k(C) \\ ko(C) \leftarrow ko^k(C) \end{array} \right\} \qquad \Delta(\mathcal{P}_I) = \left\{ \begin{array}{l} ok^k(C) \leftarrow ok^{k+1}(C) \\ ko^k(C) \leftarrow ko^{k+1}(C) \end{array} \right\}$$

Expansible Instantiation

- Given a set R of rules and a constant stream $(c_1, \ldots, c_i, \ldots, c_j, \ldots)$, the expansible instantiation of R for $j \geq 0$ is $\bigcup_{i=0}^{j} R^i$, where:

$$R^i = \{(r[i])\sigma \mid r \in \Phi(R) \cup \Pi(\mathcal{P}_I), \sigma \text{ is new ground substitution for } i\}$$
$$\cup \{(r[i])\sigma \mid r \in \Delta(\mathcal{P}_I), \sigma \text{ is ground substitution for } i\}.$$

Constant stream $(c_1, s_1, \ldots)$

$$\begin{array}{lll}
ok^k(C) \leftarrow cs(C), st(S), in(S, C) & & \\
ko^k(C) \leftarrow cs(C), {\sim}ok(C) & & [\Phi(R)] \\
\hline
ok(C) \leftarrow ok^k(C) & ko(C) \leftarrow ko^k(C) & [\Pi(\mathcal{P}_I)] \\
\hline\hline
ok^k(C) \leftarrow ok^{k+1}(C) & ko^k(C) \leftarrow ko^{k+1}(C) & [\Delta(\mathcal{P}_I)]
\end{array}$$

Expansible Instantiation

- Given a set R of rules and a constant stream $(c_1, \dots, c_i, \dots, c_j, \dots)$, the expansible instantiation of R for $j \geq 0$ is $\bigcup_{i=0}^{j} R^i$, where:

$$R^i = \{(r[i])\sigma \mid r \in \Phi(R) \cup \Pi(\mathcal{P}_I), \sigma \text{ is new ground substitution for } i\}$$
$$\cup \{(r[i])\sigma \mid r \in \Delta(\mathcal{P}_I), \sigma \text{ is ground substitution for } i\}.$$

Constant stream $(c_1, s_1, \dots)$

$$\begin{array}{lll}
ok^k(C) \leftarrow cs(C), st(S), in(S, C) & & \\
ko^k(C) \leftarrow cs(C), {\sim}ok(C) & & [\Phi(R)] \\
\hline
ok(C) \leftarrow ok^k(C) & ko(C) \leftarrow ko^k(C) & [\Pi(\mathcal{P}_I)] \\
\hline\hline
ok^k(C) \leftarrow ok^{k+1}(C) & ko^k(C) \leftarrow ko^{k+1}(C) & [\Delta(\mathcal{P}_I)]
\end{array}$$

Expansible Instantiation

- Given a set R of rules and a constant stream $(c_1, \dots, c_i, \dots, c_j, \dots)$, the expansible instantiation of R for $j \geq 0$ is $\bigcup_{i=0}^{j} R^i$, where:

$$R^i = \{(r[i])\sigma \mid r \in \Phi(R) \cup \Pi(\mathcal{P}_I), \sigma \text{ is new ground substitution for } i\}$$
$$\cup \{(r[i])\sigma \mid r \in \Delta(\mathcal{P}_I), \sigma \text{ is ground substitution for } i\}.$$

Constant stream $(c_1, s_1, \dots)$

$$\begin{array}{lll}
ok^k(C) \leftarrow cs(C), st(S), in(S, C) & & \\
ko^k(C) \leftarrow cs(C), {\sim}ok(C) & & [\Phi(R)] \\
\hline
ok(C) \leftarrow ok^k(C) & ko(C) \leftarrow ko^k(C) & [\Pi(\mathcal{P}_I)] \\
\hline\hline
ok^k(C) \leftarrow ok^{k+1}(C) & ko^k(C) \leftarrow ko^{k+1}(C) & [\Delta(\mathcal{P}_I)]
\end{array}$$

Expansible Instantiation

- Given a set R of rules and a constant stream $(c_1, \dots, c_i, \dots, c_j, \dots)$, the expansible instantiation of R for $j \geq 0$ is $\bigcup_{i=0}^{j} R^i$, where:

$$R^i = \{(r[i])\sigma \mid r \in \Phi(R) \cup \Pi(\mathcal{P}_I), \sigma \text{ is new ground substitution for } i\} \\ \cup \{(r[i])\sigma \mid r \in \Delta(\mathcal{P}_I), \sigma \text{ is ground substitution for } i\}.$$

Constant stream $(c_1, s_1, \dots)$

$$R^1 = \left\{ \begin{array}{lll} ok^1(c_1) \leftarrow cs(c_1), st(c_1), in(c_1, c_1) & & \\ ko^1(c_1) \leftarrow cs(c_1), \sim ok(c_1) & & [\Phi(R)] \\ \hline ok(c_1) \leftarrow ok^1(c_1) & ko(c_1) \leftarrow ko^1(c_1) & [\Pi(\mathcal{P}_I)] \\ \hline\hline ok^1(c_1) \leftarrow ok^2(c_1) & ko^1(c_1) \leftarrow ko^2(c_1) & [\Delta(\mathcal{P}_I)] \end{array} \right\}$$

Expansible Instantiation

- Given a set R of rules and a constant stream $(c_1, \dots, c_i, \dots, c_j, \dots)$, the expansible instantiation of R for $j \geq 0$ is $\bigcup_{i=0}^{j} R^i$, where:

$$R^i = \{(r[i])\sigma \mid r \in \Phi(R) \cup \Pi(\mathcal{P}_I), \sigma \text{ is new ground substitution for } i\}$$
$$\cup \{(r[i])\sigma \mid r \in \Delta(\mathcal{P}_I), \sigma \text{ is ground substitution for } i\}.$$

Constant stream $(c_1, s_1, \dots)$

$$R^2 = \left\{ \begin{array}{lll}
ok^2(c_1) \leftarrow cs(c_1), st(s_1), in(s_1, c_1) & & \\
ok^2(s_1) \leftarrow cs(s_1), st(c_1), in(c_1, s_1) & & \\
ok^2(s_1) \leftarrow cs(s_1), st(s_1), in(s_1, s_1) & & \\
ko^2(s_1) \leftarrow cs(s_1), \sim ok(s_1) & & [\Phi(R)] \\
\hline
ok(s_1) \leftarrow ok^2(s_1) & ko(s_1) \leftarrow ko^2(s_1) & [\Pi(\mathcal{P}_I)] \\
\hline\hline
ok^2(c_1) \leftarrow ok^3(c_1) & ko^2(c_1) \leftarrow ko^3(c_1) & [\Delta(\mathcal{P}_I)] \\
ok^2(s_1) \leftarrow ok^3(s_1) & ko^2(s_1) \leftarrow ko^3(s_1) &
\end{array} \right\}$$

Expansible Instantiation

- Given a set R of rules and a constant stream $(c_1, \dots, c_i, \dots, c_j, \dots)$, the expansible instantiation of R for $j \geq 0$ is $\bigcup_{i=0}^{j} R^i$, where:

$$R^i = \{(r[i])\sigma \mid r \in \Phi(R) \cup \Pi(\mathcal{P}_I), \sigma \text{ is new ground substitution for } i\}$$
$$\cup \{(r[i])\sigma \mid r \in \Delta(\mathcal{P}_I), \sigma \text{ is ground substitution for } i\}.$$

Constant stream $(c_1, s_1, \dots)$

$$R^2 = \left\{ \begin{array}{lll}
ok^2(c_1) \leftarrow cs(c_1), st(s_1), in(s_1, c_1) & & \\
ok^2(s_1) \leftarrow cs(s_1), st(c_1), in(c_1, s_1) & & \\
ok^2(s_1) \leftarrow cs(s_1), st(s_1), in(s_1, s_1) & & \\
ko^2(s_1) \leftarrow cs(s_1), \sim ok(s_1) & & [\Phi(R)] \\
\hline
ok(s_1) \leftarrow ok^2(s_1) & ko(s_1) \leftarrow ko^2(s_1) & [\Pi(\mathcal{P}_I)] \\
\hline\hline
ok^2(c_1) \leftarrow ok^3(c_1) & ko^2(c_1) \leftarrow ko^3(c_1) & [\Delta(\mathcal{P}_I)] \\
ok^2(s_1) \leftarrow ok^3(s_1) & ko^2(s_1) \leftarrow ko^3(s_1) &
\end{array} \right\}$$

Expansible Instantiation

- Given a set R of rules and a constant stream $(c_1, \dots, c_i, \dots, c_j, \dots)$, the expansible instantiation of R for $j \geq 0$ is $\bigcup_{i=0}^{j} R^i$, where:

$$R^i = \{(r[i])\sigma \mid r \in \Phi(R) \cup \Pi(\mathcal{P}_I), \sigma \text{ is new ground substitution for } i\}$$
$$\cup \{(r[i])\sigma \mid r \in \Delta(\mathcal{P}_I), \sigma \text{ is ground substitution for } i\}.$$

Constant stream $(c_1, s_1, \dots)$

$$R^3 = \left\{ \begin{array}{ll} ok^3(c_1) \leftarrow \dots & \\ \vdots & [\Phi(R)] \\ \hline \vdots & [\Pi(\mathcal{P}_I)] \\ \hline\hline \vdots & [\Delta(\mathcal{P}_I)] \end{array} \right\}$$

Modularity Properties

- ✔ Expansion atoms guarantee disjointness of constraints at ground level
  1. Rules
  2. Completion
  3. Loop formulas
- ✔ Union of local constraints for R^i $(1 \leq i \leq j)$ matches those of $\bigcup_{i=0}^{j} R^i$

- Expansible instantiation can be produced in successive parts that are:
  1. Sound
  2. Complete
  3. Cumulative
- Non-monotone semantics is broken down into monotone constraints
  - ➥ Reasoning process can integrate successive parts in multi-shot solving

Modularity Properties

- ✔ Expansion atoms guarantee disjointness of constraints at ground level
  1. Rules
  2. Completion
  3. Loop formulas
- ✔ Union of local constraints for R^i $(1 \leq i \leq j)$ matches those of $\bigcup_{i=0}^{j} R^i$

- Expansible instantiation can be produced in successive parts that are:
  1. Sound
  2. Complete
  3. Cumulative
- Non-monotone semantics is broken down into monotone constraints
  - ➥ Reasoning process can integrate successive parts in multi-shot solving

Modularity Properties

- ✔ Expansion atoms guarantee disjointness of constraints at ground level
  1. Rules
  2. Completion
  3. Loop formulas
- ✔ Union of local constraints for R^i $(1 \leq i \leq j)$ matches those of $\bigcup_{i=0}^{j} R^i$

- Expansible instantiation can be produced in successive parts that are:
  1. Sound
  2. Complete
  3. Cumulative
- Non-monotone semantics is broken down into monotone constraints
  - ➥ Reasoning process can integrate successive parts in multi-shot solving

Modularity Properties

- ✔ Expansion atoms guarantee disjointness of constraints at ground level
  1. Rules
  2. Completion
  3. Loop formulas
- ✔ Union of local constraints for R^i $(1 \leq i \leq j)$ matches those of $\bigcup_{i=0}^{j} R^i$

- Expansible instantiation can be produced in successive parts that are:
  1. Sound
  2. Complete
  3. Cumulative
- Non-monotone semantics is broken down into monotone constraints
  - ➥ Reasoning process can integrate successive parts in multi-shot solving

Original Semantics

- Associate interpretation I for rules R with extended interpretation I^*, augmenting I with expansion atoms a^i (based on a predicate p^i) such that $I \models B$ for a ground instance $a \leftarrow B$, where i is a stream position in between the maximum of constants in a and those in a or B

Constant stream $(c_1, s_1, \ldots)$ revisited

$$I = \{cs(c_1), st(s_1), in(s_1, c_1), ok(c_1)\}$$
$$I^* = I \cup \{ok^1(c_1), ok^2(c_1)\}$$

1. If I is a stable (or supported) model of R, given constants $\{c_1, \ldots, c_j\}$ and facts over extensional predicates, then I^* is a stable (or supported) model of the expansible instantiation of R for $j \geq 0$
2. If I' is a stable (or supported) model of the expansible instantiation of R for $j \geq 0$, given facts over extensional predicates, then $I' = I^*$ for a stable (or supported) model I of R relative to constants $\{c_1, \ldots, c_j\}$

Original Semantics

- Associate interpretation I for rules R with extended interpretation I^*, augmenting I with expansion atoms a^i (based on a predicate p^i) such that $I \models B$ for a ground instance $a \leftarrow B$, where i is a stream position in between the maximum of constants in a and those in a or B

Constant stream $(c_1, s_1, \dots)$ revisited

$$I = \{cs(c_1), st(s_1), in(s_1, c_1), ok(c_1)\}$$
$$I^* = I \cup \{ok^1(c_1), ok^2(c_1)\}$$

1. If I is a stable (or supported) model of R, given constants $\{c_1, \dots, c_j\}$ and facts over extensional predicates, then I^* is a stable (or supported) model of the expansible instantiation of R for $j \geq 0$
2. If I' is a stable (or supported) model of the expansible instantiation of R for $j \geq 0$, given facts over extensional predicates, then $I' = I^*$ for a stable (or supported) model I of R relative to constants $\{c_1, \dots, c_j\}$

Original Semantics

- Associate interpretation I for rules R with extended interpretation I^*, augmenting I with expansion atoms a^i (based on a predicate p^i) such that $I \models B$ for a ground instance $a \leftarrow B$, where i is a stream position in between the maximum of constants in a and those in a or B

Constant stream $(c_1, s_1, \dots)$ revisited

$$
\begin{aligned}
I &= \{cs(c_1), st(s_1), in(s_1, c_1), ok(c_1)\} \\
I^* &= I \cup \{ok^1(c_1), ok^2(c_1)\}
\end{aligned}
$$

1. If I is a stable (or supported) model of R, given constants $\{c_1, \dots, c_j\}$ and facts over extensional predicates, then I^* is a stable (or supported) model of the expansible instantiation of R for $j \geq 0$
2. If I' is a stable (or supported) model of the expansible instantiation of R for $j \geq 0$, given facts over extensional predicates, then $I' = I^*$ for a stable (or supported) model I of R relative to constants $\{c_1, \dots, c_j\}$

Original Semantics

- Associate interpretation I for rules R with extended interpretation I^*, augmenting I with expansion atoms a^i (based on a predicate p^i) such that $I \models B$ for a ground instance $a \leftarrow B$, where i is a stream position in between the maximum of constants in a and those in a or B

Constant stream $(c_1, s_1, \dots)$ revisited

$$I = \{cs(c_1), st(s_1), in(s_1, c_1), ok(c_1)\}$$
$$I^* = I \cup \{ok^1(c_1), ok^2(c_1)\}$$

1. If I is a stable (or supported) model of R, given constants $\{c_1, \dots, c_j\}$ and facts over extensional predicates, then I^* is a stable (or supported) model of the expansible instantiation of R for $j \geq 0$
2. If I' is a stable (or supported) model of the expansible instantiation of R for $j \geq 0$, given facts over extensional predicates, then $I' = I^*$ for a stable (or supported) model I of R relative to constants $\{c_1, \dots, c_j\}$

Outline

Partner Units

- Problem domain and instances from ASP Competition 2014
- Expansible instantiation encoded via predicates providing substitutions
- *clingo* 4 control adding objects to be assigned or resources on demand

	Single-shot solving				Multi-shot solving			
Instance	#S	∅S	#U	∅U	#S	∅S	#U	∅U
026	40	0.10	10	34.69	40	0.04	10	3.00
091	40	0.10	10	3.71	40	0.04	10	8.42
100	40	0.09	10	57.05	40	0.04	10	2.13
127	40	0.10	10	4.99	40	0.04	10	9.38
175	40	0.12	10	48.44	40	0.04	10	4.86
188	40	0.11	10	54.67	40	0.03	10	2.69

- Multi-shot solving can significantly reduce #conflicts and runtime

Partner Units

- Problem domain and instances from ASP Competition 2014
- Expansible instantiation encoded via predicates providing substitutions
- *clingo* 4 control adding objects to be assigned or resources on demand

	Single-shot solving				Multi-shot solving			
Instance	#S	∅S	#U	∅U	#S	∅S	#U	∅U
026	40	0.10	10	34.69	40	0.04	10	3.00
091	40	0.10	10	3.71	40	0.04	10	8.42
100	40	0.09	10	57.05	40	0.04	10	2.13
127	40	0.10	10	4.99	40	0.04	10	9.38
175	40	0.12	10	48.44	40	0.04	10	4.86
188	40	0.11	10	54.67	40	0.03	10	2.69

- Multi-shot solving can significantly reduce #conflicts and runtime

Discussion

- Expansible instantiation induces monotone constraints, enabling successive integration into reasoning process in multi-shot solving
- Translation approach provides scheme for introducing expansion atoms through which later additions take care of non-monotonicity
- New substitutions or ground rules, respectively, must be distinguished to guarantee modular composition of ground program parts
- Future work includes automatic support for introducing expansion atoms by need in multi-shot solving with ASP systems like *clingo* 4