Logic Programming with Function Symbols: Checking Termination of Bottom-up Evaluation Through Program Adornments

Sergio Greco, Cristian Molinaro, Irina Trubitsyna

DIMES, Università della Calabria, Italy
Logic Programs with Function Symbols

• Function symbols overcome some modeling limitations of traditional ASP systems:
  – Make modeling easier and the resulting encodings more readable and concise.
  – Increase the expressive power (Turing complete).
  – Allow us to overcome the inability of handling infinite domains.

• **Problem**: Common inference tasks become undecidable.

• **Solution**: Restrict the use of function symbols while guaranteeing decidability of common inference tasks.
Finitely-ground programs

• Finitely-ground programs [Calimeri et al. ICLP’08]
  – A finitely-ground program has a finite set of stable models, each of finite size.
  – Stable models of such programs can be computed and thus common inference tasks become decidable.
  – But deciding whether a program is finitely-ground is semi-decidable.
Finitely-ground programs

- Decidable criteria providing sufficient conditions for a program to be finitely-ground:
  - $\omega$-restricted programs [Syrjänen LPNMR’01]
  - $\lambda$-restricted programs [Gesber et al. LPNMR’07]
  - Finite domain programs [Calimeri et al. ICLP’08]
  - Argument-restricted programs [Lierler and Lifschitz ICLP’09]
  - Safe and $\Gamma$-acyclic programs [Greco et al. ICLP’12]
  - Bounded programs [Greco et al. IJCAI’13]
Contribution

• We propose a new technique that, used in conjunction with current criteria, allows us to detect more programs as finitely-ground.
• The technique transforms a program $P$ into an (adorned) “equivalent” program $P'$.
• The aim is to apply current criteria to the adorned program $P'$ rather than the original program $P$. 
Program Adornment

• Suppose we want to check if a program $P$ is finitely-ground by applying a criterion $C$.
• We first transform $P$ into an adorned program $P'$.
• Then, we apply criterion $C$ to $P'$ (rather than the original program $P$).
• (Soundness) If $P'$ satisfies criterion $C$ then the original program $P$ is finitely-ground.
• This approach strictly enlarges the class of programs recognized as finitely-ground by criterion $C$. 
Example

**Original program**

\[
p(X, X) \leftarrow base(X) \\
q(X, Y) \leftarrow p(X, Y) \\
p(f(X), g(X)) \leftarrow q(X, X)
\]

• The bottom-up evaluation of the program terminates whatever finite set of facts for *base* is added to the program.

• However, none of the current criteria is able to detect termination.
Example (Argument-restricted criterion)

**Original program**

\[ p(X, X) \leftarrow base(X) \]

\[ q(X, Y) \leftarrow p(X, Y) \]

**Argument-restricted criterion**

\[ p(f(X), g(X)) \leftarrow q(X, X) \]
## Example (Argument-restricted criterion)

<table>
<thead>
<tr>
<th><strong>Original program</strong></th>
<th><strong>Argument-restricted criterion</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$p(X, X) \leftarrow \text{base}(X)$</td>
<td>Find a function $\varphi$ that assigns a natural number to each of $p[1], p[2], q[1], q[2]$ so that the following four conditions are all satisfied:</td>
</tr>
<tr>
<td>$q(X, Y) \leftarrow p(X, Y)$</td>
<td></td>
</tr>
<tr>
<td>$p(f(X), g(X)) \leftarrow q(X, X)$</td>
<td></td>
</tr>
</tbody>
</table>
Example (Argument-restricted criterion)

**Original program**

\[ p(X,X) \leftarrow base(X) \]

\[ q(X,Y) \leftarrow p(X,Y) \]

\[ p(f(X),g(X)) \leftarrow q(X,X) \]

**Argument-restricted criterion**

Find a function \( \varphi \) that assigns a natural number to each of \( p[1], p[2], q[1], q[2] \) so that the following four conditions are all satisfied:

- \( \varphi(q[1]) \geq \varphi(p[1]) \),
Example (Argument-restricted criterion)

Original program

\[ p(X, X) \leftarrow \text{base}(X) \]

\[ q(X, Y) \leftarrow p(X, Y) \]

\[ p(f(X), g(X)) \leftarrow q(X, X) \]

Argument-restricted criterion

Find a function \( \varphi \) that assigns a natural number to each of \( p[1], p[2], q[1], q[2] \) so that the following four conditions are all satisfied:

- \( \varphi(q[1]) \geq \varphi(p[1]) \),
- \( \varphi(q[2]) \geq \varphi(p[2]) \),
Example (Argument-restricted criterion)

**Original program**

\[ p(X, X) \leftarrow \text{base}(X) \]

\[ q(X, Y) \leftarrow p(X, Y) \]

\[ p(f(X), g(X)) \leftarrow q(X, X) \]

**Argument-restricted criterion**

Find a function \( \varphi \) that assigns a natural number to each of \( p[1], p[2], q[1], q[2] \) so that the following four conditions are all satisfied:

- \( \varphi(q[1]) \geq \varphi(p[1]) \),
- \( \varphi(q[2]) \geq \varphi(p[2]) \),
- \( \varphi(p[1]) > \varphi(q[1]) \) or \( \varphi(p[1]) > \varphi(q[2]) \),
Example (Argument-restricted criterion)

**Original program**

\[ p(X, X) \leftarrow \text{base}(X) \]

\[ q(X, Y) \leftarrow p(X, Y) \]

\[ p(f(X), g(X)) \leftarrow q(X, X) \]

**Argument-restricted criterion**

Find a function \( \varphi \) that assigns a natural number to each of \( p[1], p[2], q[1], q[2] \) so that the following four conditions are all satisfied:

- \( \varphi(q[1]) \geq \varphi(p[1]) \),
- \( \varphi(q[2]) \geq \varphi(p[2]) \),
- \( \varphi(p[1]) > \varphi(q[1]) \) or \( \varphi(p[1]) > \varphi(q[2]) \),
- \( \varphi(p[2]) > \varphi(q[1]) \) or \( \varphi(p[2]) > \varphi(q[2]) \).
Example (Argument-restricted criterion)

Original program

\[ p(X, X) \leftarrow \text{base}(X) \]

\[ q(X, Y) \leftarrow p(X, Y) \]

\[ p(f(X), g(X)) \leftarrow q(X, X) \]

Argument-restricted criterion

Find a function \( \phi \) that assigns a natural number to each of \( p[1], p[2], q[1], q[2] \) so that the following four conditions are all satisfied:

- \( \phi(q[1]) \geq \phi(p[1]) \),
- \( \phi(q[2]) \geq \phi(p[2]) \),
- \( \phi(p[1]) > \phi(q[1]) \) or \( \phi(p[1]) > \phi(q[2]) \),
- \( \phi(p[2]) > \phi(q[1]) \) or \( \phi(p[2]) > \phi(q[2]) \).

As the conditions above are unsatisfiable, the program is not argument-restricted.
Example (Γ-acyclicity criterion)

Original program

\[ p(X, X) \leftarrow base(X) \]
\[ q(X, Y) \leftarrow p(X, Y) \]

\[ p(f(X), g(X)) \leftarrow q(X, X) \]

Γ-acyclicity criterion
Example (Γ-acyclicity criterion)

Original program

\[ p(X, X) \leftarrow \text{base}(X) \]
\[ q(X, Y) \leftarrow p(X, Y) \]
\[ p(f(X), g(X)) \leftarrow q(X, X) \]

Γ-acyclicity criterion

\[
\begin{align*}
p[1] & \quad \epsilon \\
q[1] & \quad \epsilon
\end{align*}
\]
Example (Γ-acyclicity criterion)

Original program

\[
p(X, X) \leftarrow \text{base}(X)
\]

\[
q(X, Y) \leftarrow p(X, Y)
\]

\[
p(f(X), g(X)) \leftarrow q(X, X)
\]

Γ-acyclicity criterion

\[
\begin{array}{c}
p[1] \\
\downarrow \text{f} \\
q[1] \\
\downarrow \varepsilon \\
p[1]
\end{array}
\]
Example (Γ-acyclicity criterion)

**Original program**

\[ p(X, X) \leftarrow base(X) \]
\[ q(X, Y) \leftarrow p(X, Y) \]
\[ p(f(X), g(X)) \leftarrow q(X, X) \]

**Γ-acyclicity criterion**

The program is not Γ-acyclic
Example

**Original program**

\[ p(X, X) \leftarrow base(X) \]
\[ q(X, Y) \leftarrow p(X, Y) \]
\[ p(f(X), g(X)) \leftarrow q(X, X) \]
Example

**Original program**

\[ p(X, X) \leftarrow base(X) \]

\[ q(X, Y) \leftarrow p(X, Y) \]

\[ p(f(X), g(X)) \leftarrow q(X, X) \]

**Adorned program**

\[ p^{\varepsilon\varepsilon}(X, X) \leftarrow base^\varepsilon(X) \]
Example

**Original program**

\[
p(X,X) \leftarrow \text{base}(X)
\]

\[
q(X,Y) \leftarrow p(X,Y)
\]

\[
p(f(X),g(X)) \leftarrow q(X,X)
\]

**Adorned program**

\[
p^{ee}(X,X) \leftarrow \text{base}^{e}(X)
\]

\[
q^{ee}(X,Y) \leftarrow p^{ee}(X,Y)
\]
Example

Original program

\[ p(X, X) \leftarrow \text{base}(X) \]
\[ q(X, Y) \leftarrow p(X, Y) \]
\[ p(f(X), g(X)) \leftarrow q(X, X) \]

Adorned program

\[ p^{\varepsilon \varepsilon}(X, X) \leftarrow \text{base}^{\varepsilon}(X) \]
\[ q^{\varepsilon \varepsilon}(X, Y) \leftarrow p^{\varepsilon \varepsilon}(X, Y) \]
\[ p^{f_{i1}}(f(X), g(X)) \leftarrow q^{\varepsilon \varepsilon}(X, X) \]
Example

Original program

\[
\begin{align*}
p(X, X) & \leftarrow \text{base}(X) \\
q(X, Y) & \leftarrow p(X, Y) \\
p(f(X), g(X)) & \leftarrow q(X, X)
\end{align*}
\]

Adorned program

\[
\begin{align*}
p^{\varepsilon\varepsilon}(X, X) & \leftarrow \text{base}^\varepsilon(X) \\
q^{\varepsilon\varepsilon}(X, Y) & \leftarrow p^{\varepsilon\varepsilon}(X, Y) \\
p^{f_1g_1}(f(X), g(X)) & \leftarrow q^{\varepsilon\varepsilon}(X, X) \\
q^{f_1g_1}(X, Y) & \leftarrow p^{f_1g_1}(X, Y)
\end{align*}
\]
Example

Original program

\[
p(X, X) \leftarrow base(X)
q(X, Y) \leftarrow p(X, Y)
p(f(X), g(X)) \leftarrow q(X, X)
\]

Adorned program

\[
p^{\varepsilon \varepsilon}(X, X) \leftarrow base^{\varepsilon}(X)
q^{\varepsilon \varepsilon}(X, Y) \leftarrow p^{\varepsilon \varepsilon}(X, Y)
p^{f_{1g_1}}(f(X), g(X)) \leftarrow q^{\varepsilon \varepsilon}(X, X)
q^{f_{1g_1}}(X, Y) \leftarrow p^{f_{1g_1}}(X, Y)
p^{??}(f(X), g(X)) \leftarrow q^{f_{1g_1}}(X, X)
\]
Example

Original program
\[
\begin{align*}
  & p(X, X) \leftarrow \text{base}(X) \\
  & q(X, Y) \leftarrow p(X, Y) \\
  & p(f(X), g(X)) \leftarrow q(X, X)
\end{align*}
\]

Adorned program
\[
\begin{align*}
  & p^{\varepsilon \varepsilon}(X, X) \leftarrow \text{base}^{\varepsilon}(X) \\
  & q^{\varepsilon \varepsilon}(X, Y) \leftarrow p^{\varepsilon \varepsilon}(X, Y) \\
  & p^{f_{g_1}}(f(X), g(X)) \leftarrow q^{\varepsilon \varepsilon}(X, X) \\
  & q^{f_{g_1}}(X, Y) \leftarrow p^{f_{g_1}}(X, Y)
\end{align*}
\]
Each adorned rule is obtained from a rule in the original program by adding adornments which keep track of the structure of the terms that can be propagated during the bottom-up evaluation.
**Example**

Original program

\[
\begin{align*}
p(X,X) & \leftarrow \text{base}(X) \\
qu(X,Y) & \leftarrow p(X,Y) \\
p(f(X),g(X)) & \leftarrow q(X,X)
\end{align*}
\]

Adorned program

\[
\begin{align*}
p^{\varepsilon\varepsilon}(X,X) & \leftarrow \text{base}^{\varepsilon}(X) \\
q^{\varepsilon\varepsilon}(X,Y) & \leftarrow p^{\varepsilon\varepsilon}(X,Y) \\
p^{f_1g_1}(f(X),g(X)) & \leftarrow q^{\varepsilon\varepsilon}(X,X) \\
q^{f_1g_1}(X,Y) & \leftarrow p^{f_1g_1}(X,Y)
\end{align*}
\]

The adorned program is “equivalent” to the original one in the following sense: the minimal model of the original program can be obtained from the minimal model of the adorned program by dropping adornments.
Example

**Original program**

\[
\begin{align*}
  p(X, X) &\leftarrow \text{base}(X) \\
  q(X, Y) &\leftarrow p(X, Y) \\
  p(f(X), g(X)) &\leftarrow q(X, X)
\end{align*}
\]

**Adorned program**

\[
\begin{align*}
  p^{\varepsilon}(X, X) &\leftarrow \text{base}^{\varepsilon}(X) \\
  q^{\varepsilon}(X, Y) &\leftarrow p^{\varepsilon}(X, Y) \\
  p^{f_{1g_1}}(f(X), g(X)) &\leftarrow q^{\varepsilon}(X, X) \\
  q^{f_{1g_1}}(X, Y) &\leftarrow p^{f_{1g_1}}(X, Y)
\end{align*}
\]

- The bottom-up evaluation of the original program always ends.
- None of the current criteria is able to realize it (when applied to the original program).
- But all current criteria realize that the bottom-up evaluation of the **adorned program** ends.
- This allows us to conclude that the bottom-up evaluation of the original program always ends (by the soundness of the proposed adornment technique).
Some terminology

- The set of **adornment symbols** is:
  \[ \{ f_i \mid f \text{ is a function symbol and } i \text{ is a natural number} \} \cup \{ \varepsilon \} \]

- \( \varepsilon \) means that the corresponding term can be a simple term (a variable or a constant).

- \( f_i \) means that the corresponding term can be a complex term of the form \( f(\ldots) \).
Some terminology

• An *adornment* for a predicate symbol $p$ of arity $n$ is a string of $n$ adornment symbols.

• An *adornment definition* for an adornment symbol $f_i$ is an expression of the form

$$f_i = f(\alpha_1, \ldots, \alpha_m)$$

where $m$ is the arity of function symbol $f$ and the $\alpha_j$'s are adornment symbols.
Some terminology

- **Adornments and adornment definitions** are used to keep track of the structure of the terms that can be propagated during the program evaluation.
Some terminology

- **Adornments and adornment definitions** are used to keep track of the structure of the terms that can be propagated during the program evaluation.

- As an example, the adorned predicate symbol $p_{f_1g_1}$
Some terminology

- Adornments and adornment definitions are used to keep track of the structure of the terms that can be propagated during the program evaluation.

- As an example, the adorned predicate symbol $p^{f_1g_1}$ with adornment definitions $f_1 = f(\varepsilon)$ and $g_1 = g(f_1)$
Some terminology

• Adornments and adornment definitions are used to keep track of the structure of the terms that can be propagated during the program evaluation.

• As an example, the adorned predicate symbol $p^{f_1g_1}$ with adornment definitions $f_1 = f(\varepsilon)$ and $g_1 = g(f_1)$ means that the evaluation of the considered program might yield atoms of the form $p(f(a),g(f(b)))$ with $a$ and $b$ being constants.
Some terminology

- **Adornments and adornment definitions** are used to keep track of the structure of the terms that can be propagated during the program evaluation.

- As an example, the adorned predicate symbol $p^{f_1g_1}$ with adornment definitions $f_1 = f(\varepsilon)$ and $g_1 = g(f_1)$ means that the evaluation of the considered program might yield atoms of the form $p(f(a),g(f(b)))$ with $a$ and $b$ being constants.

- **Database facts** are facts for base predicate symbols where function symbols do not appear.
Some terminology

• **Adornments and adornment definitions are used to keep track of the structure of the terms that can be propagated during the program evaluation.**

• As an example, the adorned predicate symbol $p^{f_1g_1}$ with adornment definitions $f_1 = f(\varepsilon)$ and $g_1 = g(f_1)$ means that the evaluation of the considered program might yield atoms of the form $p(f(a), g(f(b)))$ with $a$ and $b$ being constants.

• **Database facts** are facts for base predicate symbols where function symbols do not appear.

We initially focus on positive normal programs
Adornment Algorithm

• The adornment algorithm relies on two steps:
  1. Checking if an adorned body is *coherent*
  2. Propagating adornments from a coherently adorned body to the head
Adornment coherency

Checking if an adorned conjunction is *coherent*

1. For every adorned atom $p^{\alpha_1...\alpha_n}(t_1,...,t_n)$ in the *body* conjunction, check if every $t_i$ complies with the term structure described by $\alpha_i$
Adornment coherency

Checking if an adorned conjunction is coherent

1. For every adorned atom $p^{\alpha_1 \ldots \alpha_n}(t_1, \ldots, t_n)$ in the body conjunction, check if every $t_i$ complies with the term structure described by $\alpha_i$

   – Consider the adorned atom $p^{f_1}(g(X))$
Adornment coherency

**Checking if an adorned conjunction is coherent**

1. For every adorned atom $p^{\alpha_1\ldots\alpha_n}(t_1,\ldots,t_n)$ in the body conjunction, check if every $t_i$ complies with the term structure described by $\alpha_i$

   – Consider the adorned atom $p^{f_1(g(X))}$ with adornment definition $f_1 = f(\varepsilon)$. 
Adornment coherency

Checking if an adorned conjunction is **coherent**

1. For every adorned atom $p^{\alpha_1\ldots\alpha_n}(t_1,\ldots,t_n)$ in the body conjunction, check if every $t_i$ complies with the term structure described by $\alpha_i$

   – Consider the adorned atom $p^{f_1}(g(X))$ with adornment definition $f_1 = f(\varepsilon)$.
     The adorned atom is **NOT coherently adorned** because $g(X)$ does not comply with the term structure $f(\varepsilon)$ corresponding to $f_1$. 
Adornment coherency

Checking if an adorned conjunction is coherent

1. For every adorned atom $p^{\alpha_1...\alpha_n}(t_1,...,t_n)$ in the body conjunction, check if every $t_i$ complies with the term structure described by $\alpha_i$

   – Consider the adorned atom $p^{f_1}(g(X))$
     with adornment definition $f_1 = f(\epsilon)$.
     The adorned atom is NOT coherently adorned because $g(X)$ does not comply with the term structure $f(\epsilon)$ corresponding to $f_1$.
     The expected adornment should be of the form $p^{g_i}(g(X))$ with adornment definition $g_i = g(...)$.
Adornment coherency

Checking if an adorned conjunction is coherent

1. For every adorned atom $p^{\alpha_1...\alpha_n}(t_1,...,t_n)$ in the body conjunction, check if every $t_i$ complies with the term structure described by $\alpha_i$

   – Consider the adorned atom $p^{f_1}(g(X))$ with adornment definition $f_1 = f(\varepsilon)$. The adorned atom is NOT coherently adorned because $g(X)$ does not comply with the term structure $f(\varepsilon)$ corresponding to $f_1$. The expected adornment should be of the form $p^{g_i}(g(X))$ with adornment definition $g_i = g(...)$.

   – $p^{f_1}(f(X))$ and $p^{f_1}(X)$ are coherently adorned.
Adornment coherency

Checking if an adorned conjunction is coherent

1. For every adorned atom $p^{\alpha_1...\alpha_n}(t_1,...,t_n)$ in the body conjunction, check if every $t_i$ complies with the term structure described by $\alpha_i$

   – Consider the adorned atom $p^{f_1}(g(X))$ with adornment definition $f_1 = f(\varepsilon)$. The adorned atom is **NOT coherently adorned** because $g(X)$ does not comply with the term structure $f(\varepsilon)$ corresponding to $f_1$. The expected adornment should be of the form $p^{g_1}(g(X))$ with adornment definition $g_1 = g(...)$. 

   – $p^{f_1}(f(X))$ and $p^{f_1}(X)$ are **coherently adorned**.

   – $p^{f_1}(f(f(X)))$ is **NOT coherently adorned**. The expected adornment should be of the form $p^{f_i}(f(f(X)))$ with $f_i = f(f_j)$ and $f_j = f(...)$.
Adornment coherency

Checking if an adorned conjunction is coherent

2. Check if every variable is associated with only one adornment symbol.
Adornment coherency

Checking if an adorned conjunction is **coherent**

2. Check if every variable is associated with only one adornment symbol.

   – The adorned atom $p^{f_1 g_1}(X, X)$ with adornment definitions $f_1 = f(\varepsilon)$ and $g_1 = g(\varepsilon)$ is **NOT coherently adorned** because variable $X$ is associated with two different adornment symbols.
Adornment coherency

Checking if an adorned conjunction is *coherent*

2. Check if every variable is associated with only one adornment symbol.

   – The adorned atom $p^{f_1g_1}(X,X)$ with adornment definitions $f_1 = f(\varepsilon)$ and $g_1 = g(\varepsilon)$ is **NOT coherently adorned** because variable $X$ is associated with two different adornment symbols.

   – $p^{f_1g_1}(f(X),g(X))$ is *coherently adorned*. 


Adornment propagation

• Adornment propagation (from the body to the head): given a coherently adorned body, determine the adornment of the head.

• This is done on the basis of
  – the structure of the terms in the head, and
  – the adornments in the body.
Adornment propagation

Original rule

\[ p(X, f(X, g(X))) \leftarrow b(X) \]
Adornment propagation

Original rule

\[ p(X, f(X, g(X))) \leftarrow b(X) \]

Suppose we have derived the adorned predicate symbol \( b^\epsilon \).
This means that the program evaluation may yield atoms of the form \( b(c) \), where \( c \) is a constant.
Adornment propagation

Original rule

\[ p(X, f(X, g(X))) \leftarrow b(X) \]

Adorned body

\[ \leftarrow b^\varepsilon(X) \]

Suppose we have derived the adorned predicate symbol \( b^\varepsilon \). This means that the program evaluation may yield atoms of the form \( b(c) \), where \( c \) is a constant.
Adornment propagation

Original rule
\[ p(X, f(X, g(X))) \leftarrow b(X) \]

Adorned body
\[ \leftarrow b^\epsilon(X) \]

Thus, the original rule may yield atoms of the form \( p(c, f(c, g(c))) \).

Suppose we have derived the adorned predicate symbol \( b^\epsilon \). This means that the program evaluation may yield atoms of the form \( b(c) \), where \( c \) is a constant.
Adornment propagation

Original rule
\[ p(X, f(X, g(X))) \leftarrow b(X) \]

Adorned rule
\[ p^{\varepsilon f_1}(X, f(X, g(X))) \leftarrow b^\varepsilon(X) \]

Adornment definitions
\[ f_1 = f(\varepsilon, g_1) \]
\[ g_1 = g(\varepsilon) \]

Thus, the original rule may yield atoms of the form \( p(c, f(c, g(c))) \).

Suppose we have derived the adorned predicate symbol \( b^\varepsilon \). This means that the program evaluation may yield atoms of the form \( b(c) \), where \( c \) is a constant.

The rule head is adorned to keep track of this. We get an adorned rule and two adornment definitions.
Adornment Algorithm

• The adornment algorithm maintains
  – a set $P'$ of adorned rules,
  – a set $AP$ of adorned predicate symbols, and
  – a set $AD$ of adornment definitions.
Adornment Algorithm

• The adornment algorithm maintains
  – a set $P'$ of adorned rules,
  – a set $AP$ of adorned predicate symbols, and
  – a set $AD$ of adornment definitions.

• Given a positive normal program $P$
  1. Base predicate symbols are adorned with $\varepsilon$’s.
Adornment Algorithm

• The adornment algorithm maintains
  – a set $P'$ of adorned rules,
  – a set $AP$ of adorned predicate symbols, and
  – a set $AD$ of adornment definitions.

• Given a positive normal program $P$
  1. Base predicate symbols are adorned with $\varepsilon$’s.
  2. If there is a rule $r$ of $P$ whose body can be coherently adorned using adorned predicate symbols in $AP$, then
     a. Adorn the head of $r$ according to the chosen adorned body (this may yield new adorned predicate symbols and new adornment definitions that are added to $AP$ and $AD$).
     b. Add the obtained adorned rule to $P'$.
Adornment Algorithm

*Original program*

\[
p(X, f(X)) \leftarrow \text{base}(X)
\]

\[
p(X, f(X)) \leftarrow p(Y, X), \text{base}(Y)
\]

\[
p(X, Y) \leftarrow p(f(X), f(Y))
\]
Adornment Algorithm

Original program

\[ p(X, f(X)) \leftarrow \text{base}(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), \text{base}(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

Adorned program

*Adorned predicate symbols*

\[ \text{base}^e \]

*Adornment definitions*
Adornment Algorithm

Original program

\[
p(X, f(X)) \leftarrow base(X)
\]
\[
p(X, f(X)) \leftarrow p(Y, X), base(Y)
\]
\[
p(X, Y) \leftarrow p(f(X), f(Y))
\]

Adorned program

Adorned predicate symbols

\[ base^e \]

Adornment definitions
Adornment Algorithm

Original program

\[ p(X, f(X)) \leftarrow \text{base}(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), \text{base}(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

Adorned program

\[ \leftarrow \text{base}^\varepsilon(X) \]
Adornment Algorithm

Original program

\[
p(X, f(X)) \leftarrow base(X)
\]

\[
p(X, f(X)) \leftarrow p(Y, X), base(Y)
\]

\[
p(X, Y) \leftarrow p(f(X), f(Y))
\]

Adorned program

Adorned predicate symbols

\[
p^{εf_1}(X, f(X)) \leftarrow base^{ε}(X)
\]

Adornment definitions

\[
base^ε
\]

\[
p^{εf_1}
\]

\[
f_1 = f(ε)
\]
Adornment Algorithm

Original program
\[ p(X, f(X)) \leftarrow \text{base}(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), \text{base}(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

Adorned program
\[ p^{\varepsilon_{f_1}}(X, f(X)) \leftarrow \text{base}^\varepsilon(X) \]

Adorned predicate symbols
\[ \text{base}^\varepsilon \]
\[ p^{\varepsilon_{f_1}} \]
\[ f_1 = f(\varepsilon) \]

Adornment definitions
Adornment Algorithm

Original program

\[ p(X, f(X)) \leftarrow base(X) \]

\[ p(X, f(X)) \leftarrow p(Y, X), base(Y) \]

\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

Adorned program

\[ p^{\varepsilon f_1}(X, f(X)) \leftarrow base^\varepsilon(X) \]

Adorned predicate symbols

- \( base^\varepsilon \)
- \( p^{\varepsilon f_1} \)

Adornment definitions

\[ f_1 = f(\varepsilon) \]
Adornment Algorithm

**Original program**

\[ p(X, f(X)) \leftarrow base(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), base(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

**Adorned program**

\[ p^{\varepsilon f_1}(X, f(X)) \leftarrow base^{\varepsilon}(X) \]
\[ \leftarrow p^{\varepsilon f_1}(Y, X), base^{\varepsilon}(Y) \]

**Adorned predicate symbols**

- \( base^{\varepsilon} \)
- \( p^{\varepsilon f_1} \)

**Adornment definitions**

\( f_1 = f(\varepsilon) \)
Adornment Algorithm

Original program

\[ p(X, f(X)) \leftarrow base(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), base(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

Adorned program

\[ p^{e_{f_1}}(X, f(X)) \leftarrow base^{e}(X) \]
\[ p^{e_{f_1}}(X, f(X)) \leftarrow p^{e_{f_1}}(Y, X), base^{e}(Y) \]

Adorned predicate symbols

\[ base^e \]
\[ p^{e_{f_1}} \]
\[ p^{f_1 f_2} \]

Adornment definitions

\[ f_1 = f(e) \]
\[ f_2 = f(f_1) \]
Adornment Algorithm

Original program

\[ p(X, f(X)) \leftarrow \text{base}(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), \text{base}(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

Adorned program

\[ p^{\varepsilon f_1}(X, f(X)) \leftarrow \text{base}^\varepsilon(X) \]
\[ p^{\varepsilon f_1}(X, f(X)) \leftarrow p^{\varepsilon f_1}(Y, X), \text{base}^\varepsilon(Y) \]

Adorned predicate symbols

\[ \begin{align*}
  \text{base}^\varepsilon \\
  p^{\varepsilon f_1} \\
  p^{f_1 f_2}
\end{align*} \]

Adornment definitions

\[ \begin{align*}
  f_1 &= f(\varepsilon) \\
  f_2 &= f(f_1)
\end{align*} \]
Adornment Algorithm

**Original program**

\[ p(X, f(X)) \leftarrow \text{base}(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), \text{base}(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

**Adorned program**

**Adorned predicate symbols**

- \( \text{base}^\varepsilon \)
- \( p^{\varepsilon f_1} \)
- \( p^{f_1 f_2} \)

**Adornment definitions**

\[ f_1 = f(\varepsilon) \]
\[ f_2 = f(f_1) \]
Adornment Algorithm

Original program

\[
\begin{align*}
p(X, f(X)) &\leftarrow \text{base}(X) \\
p(X, f(X)) &\leftarrow p(Y, X), \text{base}(Y) \\
p(X, Y) &\leftarrow p(f(X), f(Y))
\end{align*}
\]

Adorned program

\[
\begin{align*}
p^{\epsilon f_1}(X, f(X)) &\leftarrow \text{base}^\epsilon(X) \\
p^{f_1 f_2}(X, f(X)) &\leftarrow p^{\epsilon f_1}(Y, X), \text{base}^\epsilon(Y) \\
&\leftarrow p^{f_1 f_2}(f(X), f(Y))
\end{align*}
\]

Adorned predicate symbols

\[
\begin{align*}
\text{base}^\epsilon \\
p^{\epsilon f_1} \\
p^{f_1 f_2}
\end{align*}
\]

Adornment definitions

\[
\begin{align*}
f_1 = f(\epsilon) \\
f_2 = f(f_1)
\end{align*}
\]
Adornment Algorithm

Original program

\[ p(X, f(X)) \leftarrow base(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), base(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

Adorned program

**Adorned predicate symbols**

\[ p^{\epsilon f_1}(X, f(X)) \leftarrow base^{\epsilon}(X) \]
\[ p_{f_1f_2}(X, f(X)) \leftarrow p^{\epsilon f_1}(Y, X), base^{\epsilon}(Y) \]
\[ p^{\epsilon f_1}(X, Y) \leftarrow p_{f_1f_2}(f(X), f(Y)) \]

**Adorned definitions**

\[ base^{\epsilon} \]
\[ p^{\epsilon f_1} \]
\[ p_{f_1f_2} \]

\[ f_1 = f(\epsilon) \]
\[ f_2 = f(f_1) \]
Adornment Algorithm

Original program

\[ p(X, f(X)) \leftarrow base(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), base(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

Adorned program

\[ p^{ef_1}(X, f(X)) \leftarrow base^e(X) \]
\[ p^{ef_1}(X, f(X)) \leftarrow p^{ef_1}(Y, X), base^e(Y) \]
\[ p^{ef_1}(X, Y) \leftarrow p^{f_1 f_2}(f(X), f(Y)) \]

Adorned predicate symbols

- \( base^e \)
- \( p^{ef_1} \)
- \( p^{f_1 f_2} \)

Adornment definitions

- \( f_1 = f(\epsilon) \)
- \( f_2 = f(f_1) \)

The adornment algorithm terminates because no new coherently adorned body conjunction can be generated.
Adornment Algorithm

**Original program**

\[ p(X, f(X)) \leftarrow \text{base}(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), \text{base}(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

**Adorned program**

\[ p^{\varepsilon f_1}(X, f(X)) \leftarrow \text{base}^\varepsilon(X) \]
\[ p^{f_1 f_2}(X, f(X)) \leftarrow p^{\varepsilon f_1}(Y, X), \text{base}^\varepsilon(Y) \]
\[ p^{\varepsilon f_1}(X, Y) \leftarrow p^{f_1 f_2}(f(X), f(Y)) \]

**Adorned predicate symbols**

- \[ \text{base}^\varepsilon \]
- \[ p^{\varepsilon f_1} \]
- \[ p^{f_1 f_2} \]

**Adornment definitions**

\[ f_1 = f(\varepsilon) \]
\[ f_2 = f(f_1) \]

\[ p^{f_1 f_2}(Y, X), \text{base}^\varepsilon(Y) \] is not coherently adorned because \( Y \) is associated with the two different adornment symbols \( f_1 \) and \( \varepsilon \)
Adornment Algorithm

Original program

\begin{align*}
p(X, f(X)) & \leftarrow base(X) \\
p(X, f(X)) & \leftarrow p(Y, X), base(Y) \\
p(X, Y) & \leftarrow p(f(X), f(Y))
\end{align*}

Adorned program

\begin{align*}
p^{\varepsilon f_1}(X, f(X)) & \leftarrow base^\varepsilon (X) \\
p^{f_1 f_2}(X, f(X)) & \leftarrow p^{\varepsilon f_1}(Y, X), base^\varepsilon (Y) \\
p^{\varepsilon f_1}(X, Y) & \leftarrow p^{f_1 f_2}(f(X), f(Y))
\end{align*}

Adorned predicate symbols

\begin{align*}
base^\varepsilon & \\
p^{\varepsilon f_1} & \\
p^{f_1 f_2} &
\end{align*}

Adornment definitions

\begin{align*}
f_1 & = f(\varepsilon) \\
f_2 & = f(f_1)
\end{align*}

\( p^{\varepsilon f_1}(f(X), f(Y)) \) is not coherently adorned because \( f(X) \) does not comply with the term structure described by \( \varepsilon \)
Adornment Algorithm

Original program

\[ p(X, f(X)) \leftarrow \text{base}(X) \]
\[ p(X, f(X)) \leftarrow p(Y, X), \text{base}(Y) \]
\[ p(X, Y) \leftarrow p(f(X), f(Y)) \]

Adorned program

\[ p^{\epsilon f_1}(X, f(X)) \leftarrow \text{base}^\epsilon(X) \]
\[ p^{f_1 f_2}(X, f(X)) \leftarrow p^{\epsilon f_1}(Y, X), \text{base}^\epsilon(Y) \]
\[ p^{\epsilon f_1}(X, Y) \leftarrow p^{f_1 f_2}(f(X), f(Y)) \]

• Both the original and the adorned programs are recursive.

• But while none of the current criteria realize that the bottom-up evaluation of the original program terminates, some of them realize that the evaluation of the adorned program ends.

• This in turn allows us to say that the evaluation of the original program terminates too.
Adornment Algorithm

Original program

\[ p(X) \leftarrow base(X) \]
\[ p(f(X)) \leftarrow p(X) \]
Adornment Algorithm

Original program

\[ p(X) \leftarrow \text{base}(X) \]
\[ p(f(X)) \leftarrow p(X) \]

Adorned program

Adorned predicate symbols

\[ \text{base}^e \]

Adornment definitions
Adornment Algorithm

Original program

\[
\begin{align*}
  p(X) & \leftarrow \text{base}(X) \\
  p(f(X)) & \leftarrow p(X)
\end{align*}
\]

Adorned program

Adorned predicate symbols

\[\text{base}^e\]

Adornment definitions
Adornment Algorithm

Original program

\[
\begin{align*}
p(X) & \leftarrow base(X) \\
p(f(X)) & \leftarrow p(X)
\end{align*}
\]

Adorned program

\[
\leftarrow base^\epsilon(X)
\]
Adornment Algorithm

Original program

\[ p(X) \leftarrow \text{base}(X) \]
\[ p(f(X)) \leftarrow p(X) \]

Adorned program

\[ p^\varepsilon(X) \leftarrow \text{base}^\varepsilon(X) \]

Adorned predicate symbols

\[ \text{base}^\varepsilon \]
\[ p^\varepsilon \]

Adornment definitions
Adornment Algorithm

Original program

\[ p(X) \leftarrow \text{base}(X) \]
\[ p(f(X)) \leftarrow p(X) \]

Adorned program

\[ p^\varepsilon(X) \leftarrow \text{base}^\varepsilon(X) \]

Adorned predicate symbols

\[ \text{base}^\varepsilon \]
\[ p^\varepsilon \]

Adornment definitions
Adornment Algorithm

**Original program**

\[
p(X) \leftarrow \text{base}(X)
\]

\[
p(f(X)) \leftarrow p(X)
\]

**Adorned program**

\[
p^\varepsilon(X) \leftarrow \text{base}^\varepsilon(X)
\]

**Adorned predicate symbols**

- \(\text{base}^\varepsilon\)
- \(p^\varepsilon\)

**Adornment definitions**
Adornment Algorithm

Original program

\[ p(X) \leftarrow \text{base}(X) \]

\[ p(f(X)) \leftarrow p(X) \]

Adorned program

\[ p^\varepsilon(X) \leftarrow \text{base}^\varepsilon(X) \]

\[ \leftarrow p^\varepsilon(X) \]
Adornment Algorithm

Original program

\[
p(X) \leftarrow base(X)
\]

\[
p(f(X)) \leftarrow p(X)
\]

Adorned program

\[
p^\varepsilon(X) \leftarrow base^\varepsilon(X)
\]

\[
p^{f_1}(f(X)) \leftarrow p^\varepsilon(X)
\]

Adorned predicate symbols

\[
\text{base}^\varepsilon
\]

\[
p^\varepsilon
\]

\[
p^{f_1}
\]

Adornment definitions

\[
f_1 = f(\varepsilon)
\]
Adornment Algorithm

Original program

\[
\begin{align*}
p(X) & \leftarrow base(X) \\
p(f(X)) & \leftarrow p(X)
\end{align*}
\]

Adorned program

<table>
<thead>
<tr>
<th>Adorned predicate symbols</th>
<th>Adornment definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p^\epsilon (X) \leftarrow base^\epsilon (X) )</td>
<td>( base^\epsilon )</td>
</tr>
<tr>
<td>( p^{f_1}(f(X)) \leftarrow p^\epsilon (X) )</td>
<td>( p^\epsilon )</td>
</tr>
<tr>
<td>( p^{f_2}(f(X)) \leftarrow p^{f_1}(X) )</td>
<td>( p^{f_1} )</td>
</tr>
<tr>
<td>( f_1 = f(\epsilon) )</td>
<td></td>
</tr>
<tr>
<td>( p^{f_3}(f(X)) \leftarrow p^{f_2}(X) )</td>
<td>( p^{f_2} )</td>
</tr>
<tr>
<td>( f_2 = f(f_1) )</td>
<td></td>
</tr>
<tr>
<td>( p^{f_4}(f(X)) \leftarrow p^{f_3}(X) )</td>
<td>( p^{f_3} )</td>
</tr>
<tr>
<td>( f_3 = f(f_2) )</td>
<td></td>
</tr>
<tr>
<td>( p^{f_4} )</td>
<td></td>
</tr>
<tr>
<td>( f_4 = f(f_3) )</td>
<td></td>
</tr>
</tbody>
</table>
Adornment Algorithm

**Original program**

\[ p(X) \leftarrow \text{base}(X) \]
\[ p(f(X)) \leftarrow p(X) \]

**Adorned program**

<table>
<thead>
<tr>
<th>Adorned predicate symbols</th>
<th>Adornment definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p^\varepsilon (X) \leftarrow \text{base}^\varepsilon (X) )</td>
<td>( \text{base}^\varepsilon )</td>
</tr>
<tr>
<td>( p^{f_1}(f(X)) \leftarrow p^\varepsilon (X) )</td>
<td>( p^{f_1} )</td>
</tr>
<tr>
<td>( p^{f_2}(f(X)) \leftarrow p^{f_1}(X) )</td>
<td>( p^{f_2} )</td>
</tr>
<tr>
<td>( p^{f_3}(f(X)) \leftarrow p^{f_2}(X) )</td>
<td>( p^{f_3} )</td>
</tr>
<tr>
<td>( p^{f_4}(f(X)) \leftarrow p^{f_3}(X) )</td>
<td>( p^{f_4} )</td>
</tr>
</tbody>
</table>

- \( f_1 = f(\varepsilon) \)
- \( f_2 = f(f_1) \)
- \( f_3 = f(f_2) \)
- \( f_4 = f(f_3) \)

- By replacing \( f_4 \) with \( f_2 \) and \( f_3 \) with \( f_1 \) in the last rule, we get the third rule
- We apply such substitutions everywhere
Adorned Algorithm

Original program

\[ p(X) \leftarrow base(X) \]
\[ p(f(X)) \leftarrow p(X) \]

Adorned program

\[ p^ε(X) \leftarrow base^ε(X) \]
\[ p^{f_1}(f(X)) \leftarrow p^ε(X) \]
\[ p^{f_2}(f(X)) \leftarrow p^{f_1}(X) \]
\[ p^{f_3}(f(X)) \leftarrow p^{f_2}(X) \]
\[ p^{f_4}(f(X)) \leftarrow p^{f_3}(X) \]

Adorned predicate symbols

- \( p^ε \)
- \( p^{f_1} \)
- \( p^{f_2} \)
- \( p^{f_3} \)
- \( p^{f_4} \)
- \( base^ε \)

Adornment definitions

- \( \forall f_1 = f(ε) \)
- \( \forall f_2 = f(f_1) \)
- \( \forall f_3 = f(f_2) \)
- \( \forall f_4 = f(f_3) \)

• By replacing \( f_4 \) with \( f_2 \) and \( f_3 \) with \( f_1 \) in the last rule, we get the third rule
• We apply such substitutions everywhere
## Adornment Algorithm

### Original program

\[
p(X) \leftarrow base(X) \\
p(f(X)) \leftarrow p(X)
\]

### Adorned program

\[
\begin{align*}
p^{\varepsilon}(X) & \leftarrow base^{\varepsilon}(X) \\
p^{f_1}(f(X)) & \leftarrow p^{\varepsilon}(X) \\
p^{f_2}(f(X)) & \leftarrow p^{f_1}(X) \\
p^{f_3}(f(X)) & \leftarrow p^{f_2}(X) \\
p^{f_4}(f(X)) & \leftarrow p^{f_3}(X)
\end{align*}
\]

### Adorned predicate symbols

<table>
<thead>
<tr>
<th>(base^\varepsilon)</th>
<th>(p^\varepsilon)</th>
<th>(p^{f_1})</th>
<th>(p^{f_2})</th>
<th>(p^{f_3})</th>
<th>(p^{f_4})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_1 = f(\varepsilon))</td>
<td>(f_2 = f(f_1))</td>
<td>(f_3 = f(f_2))</td>
<td>(f_4 = f(f_3))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- By replacing \(f_4\) with \(f_2\) and \(f_3\) with \(f_1\) in the last rule, we get the third rule
- We apply such substitutions everywhere
Adornment Algorithm

**Original program**

\[
p(X) \leftarrow \text{base}(X)
\]
\[
p(f(X)) \leftarrow p(X)
\]

**Adorned program**

\[
p^\varepsilon(X) \leftarrow \text{base}^\varepsilon(X)
\]
\[
p_{f_1}(f(X)) \leftarrow p^\varepsilon(X)
\]
\[
p_{f_2}(f(X)) \leftarrow p_{f_1}(X)
\]
\[
p_{f_3}(f(X)) \leftarrow p_{f_2}(X) \quad p_{f_1}(f(X)) \leftarrow p_{f_2}(X)
\]
\[
p_{f_4}(f(X)) \leftarrow p_{f_3}(X)
\]

**Adorned predicate symbols**

| base^\varepsilon | p^\varepsilon | p_{f_1} | p_{f_2} | f_1 = f(\varepsilon) | f_2 = f(f_1) | f_3 = f(f_2) | f_4 = f(f_3) |

**Adornment definitions**

- By replacing \(f_4\) with \(f_2\) and \(f_3\) with \(f_1\) in the last rule, we get the third rule
- We apply such substitutions everywhere
Adornment Algorithm

Original program

\[
p(X) \leftarrow \text{base}(X) \\
p(f(X)) \leftarrow p(X)
\]

Adorned program

\[
p^\varepsilon(X) \leftarrow \text{base}^\varepsilon(X) \\
p^f_1(f(X)) \leftarrow p^\varepsilon(X) \\
p^f_2(f(X)) \leftarrow p^f_1(X) \\
p^f_1(f(X)) \leftarrow p^f_2(X)
\]

Adorned predicate symbols

\[
\text{base}^\varepsilon \\
p^\varepsilon \\
p^f_1 \\
p^f_2
\]

Adornment definitions

\[
f_1 = f(\varepsilon) \\
f_2 = f(f_1) \\
f_1 = f(f_2)
\]

At this point we are not able to generate new rules with the available adorned predicate symbols and the algorithm terminates.
Adornment Algorithm

Original program

\[ p(X) \leftarrow \text{base}(X) \]
\[ p(f(X)) \leftarrow p(X) \]

Adorned program

\[ p^\varepsilon(X) \leftarrow \text{base}^\varepsilon(X) \]
\[ p^f_1(f(X)) \leftarrow p^\varepsilon(X) \]
\[ p^f_2(f(X)) \leftarrow p^f_1(X) \]
\[ p^f_1(f(X)) \leftarrow p^f_2(X) \]

The bottom-up evaluation of both programs does not terminate.

Thus, none of them is recognized as finitely-ground.

However, the adornment algorithm terminates.
General programs

- A logic program $P$ with disjunction (in the head) and negation (in the body) is transformed into a positive normal program $st(P)$ as follows.

Every rule $A_1 \lor A_2 \lor \ldots \lor A_n \leftarrow body$ in $P$ is replaced by $n$ positive normal rules:

\[
\begin{align*}
A_1 & \leftarrow body^+ \\
A_2 & \leftarrow body^+ \\
& \ldots \\
A_n & \leftarrow body^+ 
\end{align*}
\]

where $body^+$ is obtained from $body$ by deleting all negative literals.

- We then apply the adornment algorithm to $st(P)$
Properties

• **Theorem 1.** The adornment algorithm $Adorn$ always terminates.

• **Theorem 2.** Let $P$ be a positive normal program and $P' = Adorn(P)$. The minimal model of $P$ is equal to the minimal model of $P'$ with adornments dropped from predicate symbols.

• **Theorem 3.** Given a positive normal program $P$, if $Adorn(P)$ satisfies a criterion $C$, then $P \cup D$ is finitely-ground for any finite set of database facts $D$. 
Properties

• **Theorem 4.** Given a (general) program $P$, if $\text{Adorn}(st(P))$ satisfies a criterion $C$, then $P \cup D$ is finitely-ground for any finite set of database facts $D$.

• **Theorem 5.** By applying a criterion to adorned programs (rather than the original ones) we are able to recognize more programs as finitely-ground.
Conclusions

• Adornment algorithm to be used in conjunction with current criteria
  – First apply the adornment algorithm to the original program, then apply a criterion to the adorned program.
  – In this way, (strictly) more programs are recognized as finitely-ground.
Thanks!

Questions?