Motivation

HEX-Programs

- Extend ASP by external sources
- Traditional safety not sufficient due to value invention
- Current notion of strong safety is unnecessarily restrictive

Example

$$\Pi = \left\{ r_1 : t(a). \quad r_3 : s(Y) \leftarrow t(X), \text{cat}[X, a](Y). \right\}$$
$$\left\{ r_2 : \text{dom}(aa). \quad r_4 : t(X) \leftarrow s(X), \text{dom}(X). \right\}$$

Contribution

- New more liberal safety criteria
- Still guarantee finite groundability
- Based on a modular framework $\Rightarrow$ extensibility of the approach
Liberal Safety: Basic Concepts

Monotone Grounding Operator

\[ G_\Pi(\Pi') = \bigcup_{r \in \Pi} \{ r \theta \mid A \subseteq A(\Pi'), A \not\models \bot, A \models B^+(r\theta) \}, \]

where \( A(\Pi') = \{ Ta, Fa \mid a \in A(\Pi') \} \setminus \{ Fa \mid a \leftarrow . \in \Pi \} \) and \( r\theta \) is the instance of \( r \) under variable substitution \( \theta : V \rightarrow C \).

Example

Program \( \Pi \):

\[ r_1 : s(a). \quad r_2 : \text{dom}(ax). \quad r_3 : \text{dom}(axx). \]
\[ r_4 : s(Y) \leftarrow s(X), \& \text{cat}[X, x](Y), \text{dom}(Y). \]

Least fixpoint of \( G_\Pi \):

\[ r'_1 : s(a). \quad r'_2 : \text{dom}(ax). \quad r'_3 : \text{dom}(axx). \]
Monotone Grounding Operator

\[ G_{\Pi}(\Pi') = \bigcup_{r \in \Pi} \{ r\theta \mid A \subseteq A(\Pi'), A \not\models \bot, A \models B^+(r\theta) \}, \]

where \( A(\Pi') = \{Ta, Fa \mid a \in A(\Pi')\} \setminus \{Fa \mid a \leftarrow . \in \Pi\} \)
and \( r\theta \) is the instance of \( r \) under variable substitution \( \theta: V \rightarrow C \).

Example

Program \( \Pi \):

\[
\begin{align*}
    r_1 &: s(a). \\
    r_2 &: dom(ax). \\
    r_3 &: dom(axx). \\
    r_4 &: s(Y) \leftarrow s(X), \&cat[X, x](Y), dom(Y).
\end{align*}
\]

Least fixpoint of \( G_{\Pi} \):

\[
\begin{align*}
    r_1' &: s(a). \\
    r_2' &: dom(ax). \\
    r_3' &: dom(axx). \\
    r_4' &: s(ax) \leftarrow s(a), \&cat[a, x](ax), dom(ax).
\end{align*}
\]
Monotone Grounding Operator

\[ G_{\Pi}(\Pi') = \bigcup_{r \in \Pi} \{ r\theta \mid A \subseteq A(\Pi'), A \not\models \bot, A \models B^+(r\theta) \}, \]

where \( A(\Pi') = \{ \text{T}a, \text{F}a \mid a \in A(\Pi') \} \setminus \{ \text{F}a \mid a \leftarrow \text{.} \in \Pi \} \)

and \( r\theta \) is the instance of \( r \) under variable substitution \( \theta : \mathcal{V} \to \mathcal{C} \).

Example

Program \( \Pi \):

\[
\begin{align*}
    r_1 & : s(a). \quad r_2 : \text{dom}(ax). \quad r_3 : \text{dom}(axx). \\
    r_4 & : s(Y) \leftarrow s(X), \&\text{cat}[X, x](Y), \text{dom}(Y).
\end{align*}
\]

Least fixpoint of \( G_{\Pi} \):

\[
\begin{align*}
    r'_1 & : s(a). \quad r'_2 : \text{dom}(ax). \quad r'_3 : \text{dom}(axx). \\
    r'_4 & : s(ax) \leftarrow s(a), \&\text{cat}[a, x](ax), \text{dom}(ax). \\
    r'_5 & : s(axx) \leftarrow s(ax), \&\text{cat}[ax, x](axx), \text{dom}(axx).
\end{align*}
\]
# Liberal Safety: Basic Concepts

## Monotone Grounding Operator

\[ G_\Pi(\Pi') = \bigcup_{r \in \Pi} \{ r\theta \mid A \subseteq A(\Pi'), A \not\models \bot, A \models B^+(r\theta) \}, \]

where \( A(\Pi') = \{ Ta, Fa \mid a \in A(\Pi') \} \setminus \{ Fa \mid a \leftarrow . \in \Pi \} \)

and \( r\theta \) is the instance of \( r \) under variable substitution \( \theta : V \rightarrow C \).

## Example

**Program \( \Pi \):**

\[
\begin{align*}
    r_1 &: s(a). \\
    r_2 &: \text{dom}(ax). \\
    r_3 &: \text{dom}(axx). \\
    r_4 &: s(Y) \leftarrow s(X), \&\text{cat}[X, x](Y), \text{dom}(Y).
\end{align*}
\]

**Least fixpoint of \( G_\Pi \):**

\[
\begin{align*}
    r'_1 &: s(a). \\
    r'_2 &: \text{dom}(ax). \\
    r'_3 &: \text{dom}(axx). \\
    r'_4 &: s(ax) \leftarrow s(a), \&\text{cat}[a, x](ax), \text{dom}(ax). \\
    r'_5 &: s(axx) \leftarrow s(ax), \&\text{cat}[ax, x](axx), \text{dom}(axx).
\end{align*}
\]

**Intuition:** We call a program safe if this operator produces a finite grounding.
Liberal Safety

Two concepts

- A term is **bounded** if $G_{\Pi}(\Pi')$ contains only finitely many substitutions for it
- An attribute is **de-safe** if $G_{\Pi}(\Pi')$ contains only finitely many values at this attribute position

Idea

1. Start with empty set of bounded terms $B_0$ and de-safe attributes $S_0$
2. For all $n \geq 0$ until $B_n$ and $S_n$ do not change anymore
   - a. Identify additional bounded terms $\Rightarrow B_{n+1}$
      (assuming that $B_n$ are bounded and $S_n$ are de-safe)
   - b. Identify additional de-safe attributes $\Rightarrow S_{n+1}$
      (assuming that $B_{n+1}$ are bounded and $S_n$ are de-safe)
Liberal Safety

Two concepts

- A term is **bounded** if $G_{\Pi}(\Pi')$ contains only finitely many substitutions for it.
- An attribute is **de-safe** if $G_{\Pi}(\Pi')$ contains only finitely many values at this attribute position.

Idea

1. Start with empty set of bounded terms $B_0$ and de-safe attributes $S_0$.
2. For all $n \geq 0$ until $B_n$ and $S_n$ do not change anymore
   - a. Identify additional bounded terms $\Rightarrow B_{n+1}$ (assuming that $B_n$ are bounded and $S_n$ are de-safe).
   - b. Identify additional de-safe attributes $\Rightarrow S_{n+1}$ (assuming that $B_{n+1}$ are bounded and $S_n$ are de-safe).

Identification of bounded terms in Step 2a by **term bounding functions (TBFs)**
Concrete safety criteria can be plugged in by specific TBF $b(\Pi, r, S, B)$. 
Liberal Safety

Two concepts

- A term is **bounded** if $G_{\Pi}(\Pi')$ contains only finitely many substitutions for it.
- An attribute is **de-safe** if $G_{\Pi}(\Pi')$ contains only finitely many values at this attribute position.

Idea

1. Start with empty set of bounded terms $B_0$ and de-safe attributes $S_0$.
2. For all $n \geq 0$ until $B_n$ and $S_n$ do not change anymore:
   - a. Identify additional bounded terms $\Rightarrow B_{n+1}$ (assuming that $B_n$ are bounded and $S_n$ are de-safe).
   - b. Identify additional de-safe attributes $\Rightarrow S_{n+1}$ (assuming that $B_{n+1}$ are bounded and $S_n$ are de-safe).

Identification of bounded terms in Step 2a by term bounding functions (TBFs). Concrete safety criteria can be plugged in by specific TBF $b(\Pi, r, S, B)$ $\Rightarrow$ TBFs are a flexible means that however must fulfill certain conditions.
Liberal Safety: Concrete TBF

Definition (Syntactic Term Bounding Function)

\[ t \in b_{\text{syn}}(\Pi, r, S, B) \text{ iff } \]

(i) \( t \) is a constant in \( r \); or

(ii) there is an ordinary atom \( q(s_1, \ldots, s_{\text{ar}(q)}) \in B^+(r) \) s.t. \( t = s_j \), for some \( 1 \leq j \leq \text{ar}(q) \) and \( q \upharpoonright j \in S \); or

(iii) for some external atom \( \&g[\vec{X}](\vec{Y}) \in B^+(r) \), we have that \( t = Y_i \) for some \( Y_i \in \vec{Y} \), and for each \( X_i \in \vec{X} \),

\[
\begin{cases} 
X_i \in B, & \text{if } \tau(\&g, i) = \text{const}, \\
X_i \upharpoonright 1, \ldots, X_i \upharpoonright \text{ar}(X_i) \in S, & \text{if } \tau(\&g, i) = \text{pred}. 
\end{cases}
\]
Example

Program II:

\[ r_1 : s(a). \quad r_2 : \text{dom}(ax). \quad r_3 : \text{dom}(axx). \]
\[ r_4 : s(Y) \leftarrow s(X), \& \text{cat}[X, x](Y), \text{dom}(Y). \]

\[ B_1(r_2, \Pi, b_{syn}) = \{ax\}, \quad B_1(r_3, \Pi, b_{syn}) = \{axx\}, \quad B_1(r_4, \Pi, b_{syn}) = \{x\} \]
Liberal Safety: Concrete TBF

Example

Program $\Pi$:

\[
\begin{align*}
    r_1 &: s(a). \\
    r_2 &: \text{dom}(ax). \\
    r_3 &: \text{dom}(axx). \\
    r_4 &: s(Y) \leftarrow s(X), \&\text{cat}[X, x](Y), \text{dom}(Y).
\end{align*}
\]

- $B_1(r_2, \Pi, b_{syn}) = \{ax\}$, $B_1(r_3, \Pi, b_{syn}) = \{axx\}$, $B_1(r_4, \Pi, b_{syn}) = \{x\}$
- $\Rightarrow S_1(\Pi) = \{\text{dom}|1, \&\text{cat}[X, x]_{r_4}|_2\}$
Liberal Safety: Concrete TBF

Example

Program Π:

\[ r_1 : s(a). \quad r_2 : \text{dom}(ax). \quad r_3 : \text{dom}(axx). \]
\[ r_4 : s(Y) \leftarrow s(X), \text{&cat}[X,x](Y), \text{dom}(Y). \]

\[ B_1(r_2, \Pi, b_{syn}) = \{ax\}, B_1(r_3, \Pi, b_{syn}) = \{axx\}, B_1(r_4, \Pi, b_{syn}) = \{x\} \]

\[ \Rightarrow S_1(\Pi) = \{\text{dom}\upharpoonright 1, \text{&cat}[X,x]_{r_4}\upharpoonright 2\} \]

\[ B_2(r_4, \Pi, b_{syn}) = \{Y\}, B_2(r_1, \Pi, b_{syn}) = \{a\} \]
Liberal Safety: Concrete TBF

Example

Program Π:

\[
\begin{align*}
r_1 &: s(a). \\
r_2 &: \text{dom}(ax). \\
r_3 &: \text{dom}(axx). \\
r_4 &: s(Y) \leftarrow s(X), \&\text{cat}[X,x](Y), \text{dom}(Y).
\end{align*}
\]

- \( B_1(r_2, \Pi, b_{syn}) = \{ax\} \)
- \( B_1(r_3, \Pi, b_{syn}) = \{axx\} \)
- \( B_1(r_4, \Pi, b_{syn}) = \{x\} \)

\Rightarrow \( S_1(\Pi) = \{\text{dom}\|1, \&\text{cat}[X,x]_{r_4}\|1\} \)

- \( B_2(r_4, \Pi, b_{syn}) = \{Y\} \)
- \( B_2(r_1, \Pi, b_{syn}) = \{a\} \)

\Rightarrow \( S_2(\Pi) \supseteq \{s\|1, \&\text{cat}[X,x]_{r_4}\|o1\} \)

We also provide a TBF which exploits semantic properties of external sources.
Liberal Safety: Concrete TBF

Example

Program Π:

\[ r_1 : s(a). \quad r_2 : \text{dom}(ax). \quad r_3 : \text{dom}(axx). \]

\[ r_4 : s(Y) \leftarrow s(X), \text{\&cat}[X, x](Y), \text{dom}(Y). \]

\[ B_1(r_2, \Pi, b_{syn}) = \{ax\}, B_1(r_3, \Pi, b_{syn}) = \{axx\}, B_1(r_4, \Pi, b_{syn}) = \{x\} \]

\[ \Rightarrow S_1(\Pi) = \{\text{dom} \upharpoonright 1, \text{\&cat}[X, x]_{r_4} \upharpoonright 2\} \]

\[ B_2(r_4, \Pi, b_{syn}) = \{Y\}, B_2(r_1, \Pi, b_{syn}) = \{a\} \]

\[ \Rightarrow S_2(\Pi) \supseteq \{s \upharpoonright 1, \text{\&cat}[X, x]_{r_4} \upharpoonright 0 \upharpoonright 1\} \]

\[ X \in B_3(r_4, \Pi, b_{syn}) \]
Example

Program Π:

\[ r_1 : s(a). \quad r_2 : \text{dom}(ax). \quad r_3 : \text{dom}(axx). \]
\[ r_4 : s(Y) \leftarrow s(X), \quad \&\text{cat}[X, x](Y), \text{dom}(Y). \]

- \[ B_1(r_2, \Pi, b_{syn}) = \{ax\}, \quad B_1(r_3, \Pi, b_{syn}) = \{axx\}, \quad B_1(r_4, \Pi, b_{syn}) = \{x\} \]
- \[ \Rightarrow S_1(\Pi) = \{\text{dom}|1, \&\text{cat}[X, x]_{r_4}|_2\} \]
- \[ B_2(r_4, \Pi, b_{syn}) = \{Y\}, \quad B_2(r_1, \Pi, b_{syn}) = \{a\} \]
- \[ \Rightarrow S_2(\Pi) \supseteq \{s|1, \&\text{cat}[X, x]_{r_4}|_1 1\} \]
- \[ X \in B_3(r_4, \Pi, b_{syn}) \]
- \[ \Rightarrow \&\text{cat}[X, x]_{r_4}|_1 1 \in S_3(\Pi) \]
Liberal Safety: Concrete TBF

Example

Program Π:

- \( r_1 : s(a) \).
- \( r_2 : \text{dom}(ax) \).
- \( r_3 : \text{dom}(axx) \).
- \( r_4 : s(Y) \leftarrow s(X), \&\text{cat}[X, x](Y), \text{dom}(Y) \).

- \( B_1(r_2, \Pi, b_{syn}) = \{ax\}, B_1(r_3, \Pi, b_{syn}) = \{axx\}, B_1(r_4, \Pi, b_{syn}) = \{x\} \)
- \( \Rightarrow S_1(\Pi) = \{\text{dom} \upharpoonright 1, \&\text{cat}[X, x]_{r_4} \upharpoonright 1\} \)
- \( B_2(r_4, \Pi, b_{syn}) = \{Y\}, B_2(r_1, \Pi, b_{syn}) = \{a\} \)
- \( \Rightarrow S_2(\Pi) \supseteq \{s \upharpoonright 1, \&\text{cat}[X, x]_{r_4} \upharpoonright \circ 1\} \)
- \( X \in B_3(r_4, \Pi, b_{syn}) \)
- \( \Rightarrow \&\text{cat}[X, x]_{r_4} \upharpoonright 1 \in S_3(\Pi) \)

We also provide a TBF which exploits semantic properties of external sources
Liberal Safety: Results

Modular composition of TBFs:

**Proposition**

If $b_i(\Pi, r, S, B), 1 \leq i \leq \ell$, are TBFs, then $b(\Pi, r, S, B) = \bigcup_{1\leq i \leq \ell} b_i(\Pi, r, S, B)$ is a TBF.
Liberal Safety: Results

Modular composition of TBFs:

**Proposition**

If \( b_i(\Pi, r, S, B) \), \( 1 \leq i \leq \ell \), are TBFs, then \( b(\Pi, r, S, B) = \bigcup_{1 \leq i \leq \ell} b_i(\Pi, r, S, B) \) is a TBF.

Operator \( G \) is a witness for finite groundability:

**Proposition**

If \( \Pi \) is a de-safe program, then \( G_\Pi^\infty(\emptyset) \) is finite.

**Proposition**

Let \( \Pi \) be a de-safe program. Then \( \Pi \) is finitely restrictable and \( G_\Pi^\infty(\emptyset) \equiv^{pos} \Pi \).

The results hold for any TBF!
Relations to Other Notions of Safety

Using TBF $b_{syn}(\Pi, r, S, B) \cup b_{sem}(\Pi, r, S, B)$, liberal de-safety is strictly more general than many other approaches:

**Proposition**

*Every strongly de-safe [Eiter et al., 2006] program is de-safe.*

**Proposition**

*Every VI-restricted program [Calimeri et al., 2007] is de-safe.*

**Proposition**

*If $\Pi$ is $\omega$-restricted [Syrjänen, 2001], then it corresponds to a rewritten program $F(\Pi)$ which is de-safe.*
Conclusion

ASP Programs with External Sources

- Ordinary safety not sufficient due to value invention
- Traditional strong safety is unnecessarily restrictive

Liberal Safety Criteria

- Based on term bounding functions (TBFs)
- Allows for easy extensibility of the approach
- We also provide concrete TBFs, which are strictly more liberal than many other approaches

Ongoing and Future Work

- Refine and extend existing TBFs (e.g. exploiting domain-specific properties)
- Define and implement grounding algorithms for the new class of programs
References

External Sources of Knowledge and Value Invention in Logic Programming.


Omega-restricted logic programs.

Termination of term rewriting: Interpretation and type elimination.