Semantics for Prolog with Cut – Revisited

Kanazawa, Japan

Jael Kriener
INRIA-Microsoft Research
Paris, France

Andy King
University of Kent
Canterbury, UK

What this talk is about

- Denotational semantics for Prolog with cut, designed to be "oven ready" for abstract interpretation;
- Correct mistakes in [Kriener, TPLP, 2011] that we discovered with a proof assistant;
- Draw conclusions about designing logic programming semantics with Coq

Semantics for cut over sequences [de Vink, SCP, 1989]

```
p(a, _).
p(_, b).
```

$[\![p(x,y)]\!] \quad \langle\rangle = \langle\rangle$

$$\theta_1 = \{x \mapsto a\}$$
$$\theta_2 = \{x \mapsto c\}$$
$$\theta_3 = \{y \mapsto b\}$$

$$\theta_4 = \{x \mapsto a, y \mapsto b\}$$
$$\theta_5 = \{x \mapsto c, y \mapsto b\}$$

Semantics for cut over sequences [de Vink, SCP, 1989]

```
p(a, _).
p(_, b).
```

$$[\![p(x,y)]\!] \; \langle\rangle = \langle\rangle$$

$$[\![p(x,y)]\!] \; \langle\epsilon\rangle = \langle\theta_1, \theta_3\rangle$$

$$\theta_1 = \{x \mapsto a\}$$
$$\theta_2 = \{x \mapsto c\}$$
$$\theta_3 = \{y \mapsto b\}$$

$$\theta_4 = \{x \mapsto a, y \mapsto b\}$$
$$\theta_5 = \{x \mapsto c, y \mapsto b\}$$

Semantics for cut over sequences [de Vink, SCP, 1989]

```
p(a, _).
p(_, b).
```

$$[\![p(x,y)]\!] \langle\rangle = \langle\rangle$$
$$[\![p(x,y)]\!] \langle\epsilon\rangle = \langle\theta_1, \theta_3\rangle$$
$$[\![p(x,y)]\!] \langle\theta_1\rangle = \langle\theta_1, \theta_4\rangle$$

$$\theta_1 = \{x \mapsto a\}$$
$$\theta_2 = \{x \mapsto c\}$$
$$\theta_3 = \{y \mapsto b\}$$

$$\theta_4 = \{x \mapsto a, y \mapsto b\}$$
$$\theta_5 = \{x \mapsto c, y \mapsto b\}$$

Semantics for cut over sequences [de Vink, SCP, 1989]

```
p(a, _).
p(_, b).
```

$$[\![p(x,y)]\!]\ \langle\rangle = \langle\rangle$$
$$[\![p(x,y)]\!]\ \langle\epsilon\rangle = \langle\theta_1,\theta_3\rangle$$
$$[\![p(x,y)]\!]\langle\theta_1\rangle = \langle\theta_1,\theta_4\rangle$$
$$[\![p(x,y)]\!]\langle\theta_2\rangle = \langle\theta_5\rangle$$

$$\theta_1 = \{x \mapsto a\}$$
$$\theta_2 = \{x \mapsto c\}$$
$$\theta_3 = \{y \mapsto b\}$$

$$\theta_4 = \{x \mapsto a, y \mapsto b\}$$
$$\theta_5 = \{x \mapsto c, y \mapsto b\}$$

Semantics for cut over sequences [de Vink, SCP, 1989]

```
p(a, _).
p(_, b).
```

$$[\![p(x,y)]\!]\ \langle\rangle = \langle\rangle$$
$$[\![p(x,y)]\!]\ \langle\epsilon\rangle = \langle\theta_1, \theta_3\rangle$$
$$[\![p(x,y)]\!]\langle\theta_1\rangle = \langle\theta_1, \theta_4\rangle$$
$$[\![p(x,y)]\!]\langle\theta_2\rangle = \langle\theta_5\rangle$$
$$[\![p(x,y)]\!]\langle\theta_3\rangle = \langle\theta_4, \theta_3\rangle$$

$$\theta_1 = \{x \mapsto a\}$$
$$\theta_2 = \{x \mapsto c\}$$
$$\theta_3 = \{y \mapsto b\}$$

$$\theta_4 = \{x \mapsto a, y \mapsto b\}$$
$$\theta_5 = \{x \mapsto c, y \mapsto b\}$$

Semantics for cut over sequences [de Vink, SCP, 1989]

p(a, _).
p(_, b).

$$
\begin{aligned}
[\![p(x,y)]\!]\ \langle\rangle &= \langle\rangle \\
[\![p(x,y)]\!]\ \langle\epsilon\rangle &= \langle\theta_1,\theta_3\rangle \\
[\![p(x,y)]\!]\langle\theta_1\rangle &= \langle\theta_1,\theta_4\rangle \\
[\![p(x,y)]\!]\langle\theta_2\rangle &= \langle\theta_5\rangle \\
[\![p(x,y)]\!]\langle\theta_3\rangle &= \langle\theta_4,\theta_3\rangle
\end{aligned}
$$

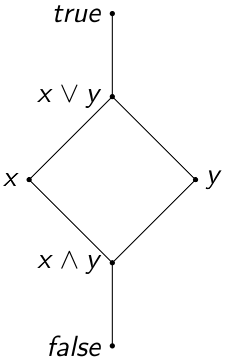
$$
\begin{aligned}
\theta_1 &= \{x \mapsto a\} \\
\theta_2 &= \{x \mapsto c\} \\
\theta_3 &= \{y \mapsto b\} \\
\theta_4 &= \{x \mapsto a, y \mapsto b\} \\
\theta_5 &= \{x \mapsto c, y \mapsto b\}
\end{aligned}
$$

Domain constructed using the ordering: $\vec{\theta} \sqsubseteq \vec{\kappa}$ iff $\vec{\theta}$ is a prefix $\vec{\kappa}$

α γ ?

$\alpha \qquad \gamma \qquad ?$

An off-the-shelf abstract domain D



$$
\begin{aligned}
\gamma(\mathit{true}) &= \mathit{Sub} \\
\gamma(x \vee y) &= \Theta_1 \cup \Theta_2 \\
\gamma(x) &= \Theta_1 \\
\gamma(y) &= \Theta_2 \\
\gamma(x \wedge y) &= \Theta_1 \cap \Theta_2 \\
\gamma(\mathit{false}) &= \emptyset
\end{aligned}
$$

$$
\begin{aligned}
\text{where } \Theta_1 &= \{\theta \in \mathit{Sub} \mid \theta(x) \text{ is ground}\} \\
\Theta_2 &= \{\theta \in \mathit{Sub} \mid \theta(y) \text{ is ground}\}
\end{aligned}
$$

Sequences over D

Problem:

$$
\begin{array}{ccc}
\langle \theta_1 \rangle & \xrightarrow{\alpha} & \langle f_1 \rangle \\
{\scriptstyle [\![\cdot]\!]} \downarrow & & \downarrow {\scriptstyle [\![\cdot]\!]^A} \\
\langle \theta_1, \theta_2 \rangle & \xleftarrow[\gamma]{} & \langle f_1, f_2 \rangle
\end{array}
$$

$$\theta_1 = \{x \mapsto a\}$$
$$\theta_2 = \{x \mapsto a, y \mapsto b\}$$

$$f_1 = x$$
$$f_2 = x \wedge y$$

Sequences over D

Problem:

$$\begin{CD}
\langle \theta_1 \rangle @>{\alpha}>> \langle f_1 \rangle \\
@V{[\![\cdot]\!]}VV @VV{[\![\cdot]\!]^A}V \\
\langle \theta_1, \theta_2 \rangle @<<{\gamma}< \langle f_1, f_2 \rangle
\end{CD}$$

$$\begin{aligned}
\theta_1 &= \{x \mapsto a\} \\
\theta_2 &= \{x \mapsto a, y \mapsto b\}
\end{aligned}$$

$$\begin{aligned}
f_1 &= x \\
f_2 &= x \wedge y
\end{aligned}$$

$$\begin{aligned}
\theta_3 &= \{x \mapsto c\} \\
\theta_4 &= \{x \mapsto c, y \mapsto b\}
\end{aligned}$$

$$\begin{CD}
\langle \theta_3 \rangle @>{\alpha}>> \langle f_1 \rangle \\
@V{[\![\cdot]\!]}VV @VV{[\![\cdot]\!]^A}V \\
\langle \theta_4 \rangle @<<{\gamma}< \langle f_2 \rangle
\end{CD}$$

Sequences over D

Problem:

$$\begin{array}{ccc} \langle\theta_1\rangle & \xrightarrow{\alpha} & \langle f_1\rangle \\ {\scriptstyle [\![\cdot]\!]}\downarrow & & \downarrow{\scriptstyle [\![\cdot]\!]^A} \\ \langle\theta_1,\theta_2\rangle & \xleftarrow[\gamma]{} & \langle f_1, f_2\rangle \end{array}$$

$$\theta_1 = \{x \mapsto a\}$$
$$\theta_2 = \{x \mapsto a, y \mapsto b\}$$

$$f_1 = x$$
$$f_2 = x \wedge y$$

$$\theta_3 = \{x \mapsto c\}$$
$$\theta_4 = \{x \mapsto c, y \mapsto b\}$$

$$\begin{array}{ccc} \langle\theta_3\rangle & \xrightarrow{\alpha} & \langle f_1\rangle \\ {\scriptstyle [\![\cdot]\!]}\downarrow & & \downarrow{\scriptstyle [\![\cdot]\!]^A} \\ \langle\theta_4\rangle & \xleftarrow[\gamma]{} & \langle f_2\rangle \end{array}$$

Solution: design $\sqsubseteq_{seq}$ so that $\langle f_2\rangle \sqsubseteq_{seq} \langle f_1, f_2\rangle$ (non-prefix)

Ordering sequences over D

Subsequence ordering:

$$\begin{array}{lllrl} \langle\rangle & \sqsubseteq_{sub} & \vec{f} & & \\ \vec{f} & \sqsubseteq_{sub} & \vec{f}' & \Rightarrow & g :: \vec{f} \sqsubseteq_{sub} g :: \vec{f}' \\ \vec{f} & \sqsubseteq_{sub} & \vec{f}' & \Rightarrow & \vec{f} \sqsubseteq_{sub} g :: \vec{f}' \end{array}$$

Mix in the domain ordering:

$$\vec{f} \sqsubseteq_{seq} \vec{g} \Leftrightarrow \exists \vec{g}' \cdot \vec{f} \sqsubseteq_{pw} \vec{g}' \wedge \vec{g}' \sqsubseteq_{sub} \vec{g}$$

Example: $\langle x \vee y, y \rangle \sqsubseteq_{seq} \langle true, x \vee y, x, x \vee y, false \rangle$ since

$$\langle x \vee y, y \rangle \sqsubseteq_{pw} \langle x \vee y, x \vee y \rangle \sqsubseteq_{sub} \langle true, x \vee y, x, x \vee y, false \rangle$$

Candidate meet [Kriener, TPLP, 2011]

$\vec{f} \otimes \vec{g} =$ remove *false* $\vec{h}$ where

$$\vec{h} = \begin{cases} \bigsqcup_{pw} \left\{ \vec{f} \sqcap_{pw} \vec{g}' \;\middle|\; \vec{g}' \sqsubseteq_{sub} \vec{g} \wedge |\vec{g}'| = |\vec{f}| \right\} & \text{if } |\vec{f}| \leq |\vec{g}| \\ \bigsqcup_{pw} \left\{ \vec{f}' \sqcap_{pw} \vec{g} \;\middle|\; \vec{f}' \sqsubseteq_{sub} \vec{f} \wedge |\vec{f}'| = |\vec{g}| \right\} & \text{otherwise} \end{cases}$$

Candidate meet [Kriener, TPLP, 2011]

$\vec{f} \otimes \vec{g} =$ remove *false* $\vec{h}$ where

$$\vec{h} = \begin{cases} \bigsqcup_{pw} \left\{ \vec{f} \sqcap_{pw} \vec{g}' \;\middle|\; \vec{g}' \sqsubseteq_{sub} \vec{g} \wedge |\vec{g}'| = |\vec{f}| \right\} & \text{if } |\vec{f}| \leq |\vec{g}| \\ \bigsqcup_{pw} \left\{ \vec{f}' \sqcap_{pw} \vec{g} \;\middle|\; \vec{f}' \sqsubseteq_{sub} \vec{f} \wedge |\vec{f}'| = |\vec{g}| \right\} & \text{otherwise} \end{cases}$$

Then join can be constructed from meet to give a complete lattice

Candidate meet [Kriener, TPLP, 2011]

$\vec{f} \otimes \vec{g} =$ remove *false* $\vec{h}$ where

$$\vec{h} = \begin{cases} \bigsqcup_{pw} \left\{ \vec{f} \sqcap_{pw} \vec{g}' \;\middle|\; \vec{g}' \sqsubseteq_{sub} \vec{g} \wedge |\vec{g}'| = |\vec{f}| \right\} & \text{if } |\vec{f}| \leq |\vec{g}| \\ \bigsqcup_{pw} \left\{ \vec{f}' \sqcap_{pw} \vec{g} \;\middle|\; \vec{f}' \sqsubseteq_{sub} \vec{f} \wedge |\vec{f}'| = |\vec{g}| \right\} & \text{otherwise} \end{cases}$$

Then join can be constructed from meet to give a complete lattice

Or can it?

Coq formalisation revealed that $\otimes$ is non-monotonic

Problem:

- Observe $\langle true, true, x\rangle \otimes \langle x, true, true\rangle = \langle x, true, x\rangle$
- Observe too $\langle true, true\rangle \otimes \langle x, true, true\rangle = \langle true, true\rangle$
- Notice $\langle true, true\rangle \sqsubseteq_{seq} \langle true, true, x\rangle$
- By monotonity one would expect $\langle true, true\rangle \sqsubseteq_{seq} \langle x, true, x\rangle$

Coq formalisation revealed that $\otimes$ is non-monotonic

Problem:

- Observe $\langle true, true, x \rangle \otimes \langle x, true, true \rangle = \langle x, true, x \rangle$
- Observe too $\langle true, true \rangle \otimes \langle x, true, true \rangle = \langle true, true \rangle$
- Notice $\langle true, true \rangle \sqsubseteq_{seq} \langle true, true, x \rangle$
- By monotonity one would expect $\langle true, true \rangle \sqsubseteq_{seq} \langle x, true, x \rangle$

Show-stopper?

Coq formalisation revealed that $\otimes$ is non-monotonic

Problem:

- Observe $\langle \mathit{true}, \mathit{true}, x\rangle \otimes \langle x, \mathit{true}, \mathit{true}\rangle = \langle x, \mathit{true}, x\rangle$
- Observe too $\langle \mathit{true}, \mathit{true}\rangle \otimes \langle x, \mathit{true}, \mathit{true}\rangle = \langle \mathit{true}, \mathit{true}\rangle$
- Notice $\langle \mathit{true}, \mathit{true}\rangle \sqsubseteq_{seq} \langle \mathit{true}, \mathit{true}, x\rangle$
- By monotonity one would expect $\langle \mathit{true}, \mathit{true}\rangle \sqsubseteq_{seq} \langle x, \mathit{true}, x\rangle$

Show-stopper?

Solution: employ order ideals induced by $\sqsubseteq_{seq}$ as follows:

- $Seq^{\downarrow}(D) = \{S \mid S = {\downarrow}S\}$ where ${\downarrow}S = \{\vec{f} \mid \vec{f} \sqsubseteq_{seq} \vec{g} \wedge \vec{g} \in S\}$
- Then $\langle Seq^{\downarrow}(D), \subseteq, \cap, \cup\rangle$ is a complete lattice

Non-monotonicity bites back with cut

Problem:

- To illustrate consider the predicate:

  ```
  liar :- liar, !, fail.
  liar.
  ```

  which succeeds if it fails and fails if it succeeds.
- Non-monotonicity has been previously dealt with by observing that predicates such as liar also diverge, which give them a stable value $\perp$ [de Vink, SCP, 1989]
- But stratification [Apt, Blair and Walker, 1988] arguably gives a simpler way to handle non-monotonicity that focusses solely on one concern

Non-monotonicity bites back with cut

Problem:

- To illustrate consider the predicate:

  ```
  liar :- liar, !, fail.
  liar.
  ```

  which succeeds if it fails and fails if it succeeds.
- Non-monotonicity has been previously dealt with by observing that predicates such as liar also diverge, which give them a stable value $\perp$ [de Vink, SCP, 1989]
- But stratification [Apt, Blair and Walker, 1988] arguably gives a simpler way to handle non-monotonicity that focusses solely on one concern

Solution: avoid these vicious circular definitions

Cut-normal programs

Introduce cut-normal form in which each predicate takes the form:

$$p(\vec{x}) \text{ :- } G_1; G_2, !, G_3; G_4$$

where each G_i is a (cut-free) conjunctive goal

Cut-normal programs

Introduce cut-normal form in which each predicate takes the form:

$$p(\vec{x}) \text{ :- } G_1; G_2, !, G_3; G_4$$

where each G_i is a (cut-free) conjunctive goal

```
memberchk(X, L) :-
    member(X, L), !.

member(X, [X|_]).
member(X, [_|L) :-
    member(X, L).
```

$\Longrightarrow$

```
memberchk(X, L) :-
    false ;
    member(X, L), !, true ;
    false.

member(X, Y) :-
    Y = [X|_] ;
    false, !, false ;
    Y = [_|L], member(X, L).
```

Cut-normal programs

Introduce cut-normal form in which each predicate takes the form:

$$p(\vec{x}) \text{ :- } G_1;\, G_2, !, G_3;\, G_4$$

where each G_i is a (cut-free) conjunctive goal

```
memberchk(X, L) :-
    member(X, L), !.

member(X, [X|_]).
member(X, [_|L) :-
    member(X, L).
```

$\Longrightarrow$

```
memberchk(X, L) :-
    false ;
    member(X, L), !, true ;
    false.

member(X, Y) :-
    Y = [X|_] ;
    false, !, false ;
    Y = [_|L], member(X, L).
```

Admission: the transform has not been shown to be correct

Cut-stratified programs

A program is cut-stratified if it can be partitioned into strata:

$S_1 = \left\{ \right.$
```
member(X, Y) :-
    Y = [X|_]) ;
    false, !, false ;
    Y = [_|L], member(X, L).
```
$\left. \right\}$ $S_2 = \left\{ \right.$
```
memberchk(X, L) :-
    false ;
    member(X, L), !, true ;
    false.
```
$\left. \right\}$

where for all $p(\vec{x})$:- $G_1; G_2, !, G_3; G_4 \in S_i$ all calls in G_2 are to predicates defined in $S_1 \cup \ldots \cup S_{i-1}$

Cut-stratified programs

A program is cut-stratified if it can be partitioned into strata:

$S_1 =$
```
member(X, Y) :-
    Y = [X|_]) ;
    false, !, false ;
    Y = [_|L], member(X, L).
```
$S_2 =$
```
memberchk(X, L) :-
    false ;
    member(X, L), !, true ;
    false.
```

where for all $p(\vec{x})$:- $G_1; G_2, !, G_3; G_4 \in S_i$ all calls in G_2 are to predicates defined in $S_1 \cup \ldots \cup S_{i-1}$

Restriction seems to capture programming practise:

- not found an non-stratified example in the wild;
- not been able to manufacture an example that puts non-stratfied predicate to good use

Evaluating cut-stratified programs

Denotational semantics then amounts to computing an environment

$$Env = Atom \to Seq^{\downarrow}(D) \to Seq^{\downarrow}(D)$$

over a cut-stratified program $S_1, \ldots, S_n$.

Lift $\subseteq$ ordering on $Seq^{\downarrow}(D)$ to order Env by $\sqsubseteq$

The heart of our semantics is a fixpoint operator $\mathcal{F}_{S_i}$ that will map each stratum S_i into a growing function of type $Env \to Env$

Growing functions

A growing function $f : Env \to Env$ satisfies a weak monotonicity property:

$$\forall \mathfrak{f}\mathfrak{g}\mathfrak{h} \in Env . \mathfrak{f} \sqsubseteq \mathfrak{g} \sqsubseteq \mathfrak{h} \sqsubseteq (f \uparrow \omega)(\mathfrak{f}) \Rightarrow f(\mathfrak{g}) \sqsubseteq f(\mathfrak{h})$$

Then construct

$$\begin{aligned} \mathfrak{f}_1 &= (\mathcal{F}_{S_1} \uparrow \omega)(\bot) \\ \mathfrak{f}_2 &= (\mathcal{F}_{S_2} \uparrow \omega)(\mathfrak{f}_1) \\ &\vdots \\ \mathfrak{f}_n &= (\mathcal{F}_{S_n} \uparrow \omega)(\mathfrak{f}_{n-1}) \end{aligned}$$

CoLoR library only formalises classic fixpoint results

Concluding discussion

Sobering lessons:

- Coq discovered holes in our logic programming semantics that had undergone internal checking and external review;
- Architect of semantics is not the best person to prove their correctness because of false suppositions;
- Repairing join had far reaching implications for semantics

Future work:

- Refine semantics with set abstractions that are pairs, one that is upward closed, the other domain closed, akin to an interval;
- Synthesis our determinacy analysis from our semantics
- Extract abstract interpreter [Blazy et al, SAS, 2012]