Constraint Logic Programming

Sylvain Soliman, François Fages and Nicolas Beldiceanu
{Sylvain.Soliman,Francois.Fages}@inria.fr

INRIA – Projet CONTRAINTES

MPRI C-2-4-1 Course – September-November, 2006

Part I: CLP - Introduction and Logical Background

Part II: Constraint Logic Programs

Part III

Operational and Fixpoint Semantics

Part III: Operational and Fixpoint Semantics

Operational semantics: CSLD Resolution

A successful derivation is a derivation of the form

$$G \longrightarrow G_1 \longrightarrow G_2 \longrightarrow \ldots \longrightarrow c|\square$$

c is called a for G.

Operational semantics: CSLD Resolution

$$\frac{(p(t_1,t_2) \leftarrow c'|A_1,...,A_n)\theta \in P \qquad \mathcal{X} \models \exists(c \wedge s_1 = t_1 \wedge s_2 = t_2 \wedge c')}{(c|\alpha, p(s_1,s_2), \alpha') \longrightarrow (c, s_1 = t_1, s_2 = t_2, c' \mid \alpha, A_1, ..., A_n, \alpha')}$$

where θ is a renaming substitution of the program clause with new variables.

A successful derivation is a derivation of the form

$$G \longrightarrow G_1 \longrightarrow G_2 \longrightarrow ... \longrightarrow c|\Box$$

c is called a for G.

Operational semantics: CSLD Resolution

$$\frac{(p(t_1,t_2) \leftarrow c'|A_1,\ldots,A_n)\theta \in P \qquad \mathcal{X} \models \exists(c \wedge s_1 = t_1 \wedge s_2 = t_2 \wedge c')}{(c|\alpha, p(s_1,s_2), \alpha') \longrightarrow (c, s_1 = t_1, s_2 = t_2, c' \mid \alpha, A_1, \ldots, A_n, \alpha')}$$

where θ is a renaming substitution of the program clause with new variables.

A successful derivation is a derivation of the form

$$G \longrightarrow G_1 \longrightarrow G_2 \longrightarrow \ldots \longrightarrow c|\square$$

c is called a computed answer constraint for G.

$\wedge$-Compositionality of CSLD-derivations

Lemma ($\wedge$-compositionality)

c is a computed answer for the goal $(d|A_1, ..., A_n)$
iff
there exist computed answers $c_1, ..., c_n$ for the goals $true|A_1, ..., true|A_n$, such that $c = d \wedge \bigwedge_{i=1}^{n} c_i$ is satisfiable.

Corollary

Independance of the selection strategy.

$\wedge$-Compositionality of CSLD-derivations

Proof.

($\Leftarrow$) $d|A_1, \ldots, A_n \rightarrow^* d \wedge c_1|A_2, \ldots, A_n \ldots \rightarrow^* d \wedge c_1 \wedge \ldots \wedge c_n|\Box$.

($\Rightarrow$) By induction on the length l of the derivation.

If $l = 1$ we have $true|A_1 \rightarrow c_1|\Box$.

Otherwise, suppose A_1 is the selected atom, there exists a rule $(A_1 \leftarrow d_1|B_1, \ldots, B_k) \in P$ such that
$d|A_1, \ldots, A_n \rightarrow d \wedge d_1|B_1, \ldots, B_k, A_2, \ldots, A_n \rightarrow^* c|\Box$.

By induction, there exist computed answers $e_1, \ldots, e_l, c_2, \ldots, c_n$ for the goals $B_1, \ldots, B_l, A_2, \ldots, A_n$ such that
$c = d \wedge d_1 \wedge \bigwedge_{i=1}^{l} e_i \wedge \bigwedge_{j=2}^{n} c_j$. Now let $c_1 = d_1 \wedge \bigwedge_{i=1}^{l} e_i$, c_1 is a computed answer for $true|A_1$. $\Box$

Operational Semantics of CLP($\mathcal{X}$) Programs

Observation of the sets of *projected computed answer constraints*

$$O(P) = \{(\exists X\ c)|A \ :\ true|A \longrightarrow^* c|\Box,\ \mathcal{X} \models \exists(c),\ X = V(c)\setminus V(A)\}$$

Program equivalence: $P \equiv P'$ iff $O(P) = O(P')$ iff for every goal G, P and P' have the same sets of computed answer constraints.

Finer observables: the multisets of computed answer constraints or the sets of succesful CSLD derivations (equivalence of traces)

More abstract observable: the set of goals having a success (theorem proving versus programming point of view).

Operational Semantics of CLP($\mathcal{X}$) Programs

Observation of computed answer constraints

$$O_{ca}(P) = \{c|A \ : \ true|A \longrightarrow^* c|\Box, \ \mathcal{X} \models \exists(c)\}$$

$P \equiv_{ca} P'$ iff for every goal G, P and P' have the same sets of computed answer constraints.

Observation of ground successes

$$O_{gs}(P) = \{A\rho \in B_{\mathcal{X}} \ : \ true|A \longrightarrow^* c|\Box, \ \mathcal{X} \models c\rho\}$$

$P \equiv_{gs} P'$ iff P and P' have the same ground success sets, iff for every goal G, G has a CSLD refutation in P iff G has one in P'.

Definitions

Let $(S, \leq)$ be a partial order. Let $X \subseteq S$ be a subset of S.
An upper bound of X is an element $a \in S$ such that $\forall x \in X\ x \leq a$.
The maximum element of X, if it exists, is the unique upper bound of X belonging to X.
The least upper bound (lub) of X, if it exists, is the minimum of the upper bounds of X.
A sup-semi-lattice is a partial order such that every finite part admits a lub.
A lattice is a sup-semi-lattice and an inf-semi-lattice.
A chain is an increasing sequence $x_1 \leq x_2 \leq \ldots$
A partial order is complete if every chain admits a lub.
A function $f : S \rightarrow S$ is monotonic if $x \leq y \Rightarrow f(x) \leq f(y)$.
continuous if $f(lub(X)) = lub(f(X))$ for every chain X.

Fixpoint theorems

Theorem (Knaster-Tarski)

Let S be a complete partial order. Let $f : S \to S$ be a continuous operator over S. Then f admits a least fixed point $lfp(f) = f \uparrow \omega$.

Proof.

First, as f is continuous, f is monotonic, hence $\perp \leq f(\perp) \leq f(f(\perp)) \leq \ldots$ forms an increasing chain. Let $a = lub(\{f^n(\perp) | n \in \mathbb{N}\}) = f \uparrow \omega$. By continuity $f(a) = lub(\{f^{n+1}(\perp) \mid n \in \mathbb{N}\}) = a$, hence a is a fixed point of f.
Let e be any fixed point of f. We show that for all integer n, $f^n(\perp) \leq e$, by induction on n. Clearly $\perp \leq e$. Furthermore if $f^n(\perp) \leq e$ then by monotonicity, $f^{n+1}(\perp) \leq f(e) = e$.
Thus $f^n(\perp) \leq e$ for all n, hence $a \leq e$. □

Least Post-Fixed Point

Theorem

Let S be a complete sup-semi-lattice. Let f be a continuous operator over S. Then f admits a least post-fixed point (i.e. an element e satisfying $f(e) \leq e$) which is equal to $lfp(f)$.

Proof.

Let $g(x) = lub(x, f(x))$.
An element e is a post fixed point of f, i.e. $f(e) \leq e$, if and only if e is a fixed point of g, $g(e) = e$.
Now g is continuous, hence $lfp(g)$ is the least fixed point of g and the least post-fixed point of f.
Furthermore, $lfp(g) = lub\{f^n(\perp)\} = lfp(f)$. □

Fixpoint semantics of O_{gs}

Consider the complete lattice of $\mathcal{X}$-interpretations $(2^{\mathcal{B}_\mathcal{X}}, \subseteq)$
The bottom element is the empty $\mathcal{X}$-interpretation (all atoms false)
The top element is $\mathcal{B}_\mathcal{X}$ (all atoms true).

A chain X is an increasing sequence $I_1 \subseteq I_2 \subseteq \ldots$
$lub(X) = \bigcup_{i \geq 1} I_i$.

Define the semantics $O_{gs}(P)$ as the least solution of a fixpoint equation over $2^{\mathcal{B}_\mathcal{X}}$: $I = T(I)$.

$T_P^{\mathcal{X}}$ immediate consequence operator

$T_P^{\mathcal{X}} : 2^{\mathcal{B}_\mathcal{X}} \rightarrow 2^{\mathcal{B}_\mathcal{X}}$ is defined by:

$$\begin{array}{rcl} T_P^{\mathcal{X}}(I) & = & \{A\rho \in \mathcal{B}_\mathcal{X} | \text{ there exists a renamed clause in normal form} \\ & & (A \leftarrow c | A_1, \ldots, A_n) \in P, \text{ and a valuation } \rho \text{ s.t.} \\ & & \mathcal{X} \models c\rho \text{ and } \{A_1\rho, \ldots, A_n\rho\} \subseteq I\} \end{array}$$

Example

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

$$\begin{array}{rcl} T_P^{\mathcal{H}}(\emptyset) & = & \{append([], B, B) \mid B \in \mathcal{H}\} \\ T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset)) & = & T_P^{\mathcal{H}}(\emptyset) \cup \{append([X], B, [X|B]) \mid X, B \in \mathcal{H}\} \\ T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset))) & = & T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset)) \cup \\ & & \{append([X, Y], B, [X, Y|B]) \mid X, Y, B \in \mathcal{H}\} \end{array}$$

INRIA

$T_P^{\mathcal{X}}$ immediate consequence operator

$T_P^{\mathcal{X}} : 2^{\mathcal{B}_{\mathcal{X}}} \to 2^{\mathcal{B}_{\mathcal{X}}}$ is defined by:

$$T_P^{\mathcal{X}}(I) = \{A\rho \in \mathcal{B}_{\mathcal{X}} | \text{ there exists a renamed clause in normal form } (A \leftarrow c|A_1,\ldots,A_n) \in P, \text{ and a valuation } \rho \text{ s.t. } \mathcal{X} \models c\rho \text{ and } \{A_1\rho,\ldots,A_n\rho\} \subseteq I\}$$

Example

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

$$\begin{array}{lcl} T_P^{\mathcal{H}}(\emptyset) & = & \{append([], B, B) \mid B \in \mathcal{H}\} \\ T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset)) & = & T_P^{\mathcal{H}}(\emptyset) \cup \{append([X], B, [X|B]) \mid X, B \in \mathcal{H}\} \\ T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset))) & = & T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset)) \cup \\ & & \{append([X, Y], B, [X, Y|B]) \mid X, Y, B \in \mathcal{H}\} \end{array}$$

$T_P^{\mathcal{X}}$ immediate consequence operator

$T_P^{\mathcal{X}} : 2^{\mathcal{B}_\mathcal{X}} \rightarrow 2^{\mathcal{B}_\mathcal{X}}$ is defined by:

$$T_P^{\mathcal{X}}(I) = \{A\rho \in \mathcal{B}_\mathcal{X} | \text{ there exists a renamed clause in normal form } (A \leftarrow c | A_1, ..., A_n) \in P, \text{ and a valuation } \rho \text{ s.t. } \mathcal{X} \models c\rho \text{ and } \{A_1\rho, ..., A_n\rho\} \subseteq I\}$$

Example

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

$$
\begin{aligned}
T_P^{\mathcal{H}}(\emptyset) &= \{append([], B, B) \mid B \in \mathcal{H}\} \\
T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset)) &= T_P^{\mathcal{H}}(\emptyset) \cup \{append([X], B, [X|B]) \mid X, B \in \mathcal{H}\} \\
T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset))) &= T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset)) \cup \\
&\quad \{append([X, Y], B, [X, Y|B]) \mid X, Y, B \in \mathcal{H}\}
\end{aligned}
$$

$T_P^{\mathcal{X}}$ immediate consequence operator

$T_P^{\mathcal{X}} : 2^{\mathcal{B}_{\mathcal{X}}} \to 2^{\mathcal{B}_{\mathcal{X}}}$ is defined by:

$$T_P^{\mathcal{X}}(I) = \{A\rho \in \mathcal{B}_{\mathcal{X}} | \text{ there exists a renamed clause in normal form } (A \leftarrow c|A_1, \ldots, A_n) \in P, \text{ and a valuation } \rho \text{ s.t. } \mathcal{X} \models c\rho \text{ and } \{A_1\rho, \ldots, A_n\rho\} \subseteq I\}$$

Example

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

$$\begin{array}{lcl} T_P^{\mathcal{H}}(\emptyset) & = & \{append([], B, B) \mid B \in \mathcal{H}\} \\ T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset)) & = & T_P^{\mathcal{H}}(\emptyset) \cup \{append([X], B, [X|B]) \mid X, B \in \mathcal{H}\} \\ T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset))) & = & T_P^{\mathcal{H}}(T_P^{\mathcal{H}}(\emptyset)) \cup \\ & & \{append([X, Y], B, [X, Y|B]) \mid X, Y, B \in \mathcal{H}\} \end{array}$$

Continuity of $T_P^{\mathcal{X}}$ operator

Proposition

$T_P^{\mathcal{X}}$ is a continuous operator on the complete lattice of $\mathcal{X}$-interpretations.

Proof.

Let X be a chain of $\mathcal{X}$-interpretations. $A\rho \in T_P^{\mathcal{X}}(lub(X))$,
iff $(A \leftarrow c|A_1, ..., A_n) \in P$, $\mathcal{X} \models c\rho$ and $\{A_1\rho, ..., A_n\rho\} \subset lub(X)$,
iff $(A \leftarrow c|A_1, ..., A_n) \in P$, $\mathcal{X} \models c\rho$ and $\{A_1\rho, ..., A_n\rho\} \subset I$,
for some $I \in X$ (as X is a chain)
iff $A\rho \in T_P^{\mathcal{X}}(I)$ for some $I \in X$, iff $A\rho \in lub(T_P^{\mathcal{X}}(X))$. □

Corollary

$T_P^{\mathcal{X}}$ admits a least (post) fixed point $T_P^{\mathcal{X}} \uparrow \omega$.

Full abstraction

Let $F_1(P) = \text{lfp}(T_P^{\mathcal{X}}) = T_P^{\mathcal{X}} \uparrow \omega = \ldots T_P^{\mathcal{X}}(T_P^{\mathcal{X}}(\emptyset))\ldots$

Theorem ([JL87])

$F_1(P) = O_{gs}(P)$.

$F_1(P) \subseteq O_{gs}(P)$ is proved by induction on the powers n of $T_P^{\mathcal{X}}$. $n = 0$ is trivial. Let $A\rho \in T_P^{\mathcal{X}} \uparrow n$, there exists a rule $(A \leftarrow c|A_1, \ldots, A_n) \in P$, s.t. $\{A_1\rho, \ldots, A_n\rho\} \subseteq T_P^{\mathcal{X}} \uparrow n-1$ and $\mathcal{X} \models c\rho$. By induction $\{A_1\rho, \ldots, A_n\rho\} \subseteq O_{gs}(P)$. By definition of O_{gs} we get $A\rho \in O_{gs}(P)$.
$O_{gs}(P) \subseteq F_1(P)$ is proved by induction on the length of derivations. Successes with derivation of length 0 are ground facts in $T_P^{\mathcal{X}} \uparrow 1$. Let $A\rho \in O_{gs}(P)$ with a derivation of length n. By definition of O_{gs} there exists $(A \leftarrow c|A_1, \ldots, A_n) \in P$ s.t. $\{A_1\rho, \ldots, A_n\rho\} \subseteq O_{gs}(P)$ and $\mathcal{X} \models c\rho$. By induction $\{A_1\rho, \ldots, A_n\rho\} \subseteq F_1(P)$. Hence by definition of $T_P^{\mathcal{X}}$ we get $A\rho \in F_1(P)$.

$T_P^{\mathcal{X}}$ and $\mathcal{X}$ models

Proposition

I is a $\mathcal{X}$-model of P iff I is a post-fixed point of $T_P^{\mathcal{X}}$, $T_P^{\mathcal{X}}(I) \subseteq I$.

Proof.

I is a $\mathcal{X}$-model of P,
iff for each clause $A \leftarrow c|A_1, \ldots, A_n \in P$ and for each $\mathcal{X}$-valuation ρ, if $\mathcal{X} \models c\rho$ and $\{A_1\rho, \ldots, A_n\rho\} \subseteq I$ then $A\rho \in I$,
iff $T_P^{\mathcal{X}}(I) \subseteq I$. $\square$

$T_P^{\mathcal{X}}$ and $\mathcal{X}$ models

Theorem (Least $\mathcal{X}$-model [JL87])

Let P be a constraint logic program on $\mathcal{X}$. P has a least $\mathcal{X}$-model, denoted by $M_P^{\mathcal{X}}$ satisfying:

$$M_P^{\mathcal{X}} = F_1(P)$$

Proof.

$F_1(P) = lfp(T_P^{\mathcal{X}})$ is also the least post-fixed point of $T_P^{\mathcal{X}}$, thus by Prop. 9, $lfp(T_P^{\mathcal{X}})$ is the least $\mathcal{X}$-model of P. □

Fixpoint semantics of O_{ca}

Consider the set of constrained atoms
$\mathcal{B}'_{\mathcal{X}} = \{c|A \ : \ A$ is an atom and $\mathcal{X} \models \exists(c)\}$ modulo renaming.

Consider the lattice of constrained interpretations $(2^{\mathcal{B}'_{\mathcal{X}}}, \subseteq)$.

For a constrained interpretation I, let us define the closed $\mathcal{X}$-interpretation:
$[I]_{\mathcal{X}} = \{A\rho \ :$ there exists a valuation ρ and $c|A \in I$ s.t. $\mathcal{X} \models c\rho\}$.

Define the semantics $O_{ca}(P)$ as the least solution of a fixpoint equation over $2^{\mathcal{B}'_{\mathcal{X}}}$.

Non-ground immediate consequence operator

$S_P^{\mathcal{X}} : 2^{\mathcal{B}'_{\mathcal{X}}} \longrightarrow 2^{\mathcal{B}'_{\mathcal{X}}}$ is defined as:

$$S_P^{\mathcal{X}}(I) = \{c|A \in \mathcal{B}'_{\mathcal{X}} \mid \text{there exists a renamed clause in normal form } (A \leftarrow d|A_1, \ldots, A_n) \in P, \text{ and constrained atoms } \{c_1|A_1, \ldots, c_n|A_n\} \subseteq I, \text{ s.t. } c = d \wedge \bigwedge_{i=1}^{n} c_i \text{ is } \mathcal{X}\text{-satisfiable}\}$$

Proposition

For any $\mathcal{B}'_{\mathcal{X}}$-interpretation I, $[S_P^{\mathcal{X}}(I)]_{\mathcal{X}} = T_P^{\mathcal{X}}([I]_{\mathcal{X}})$.

Proof.

$A\rho \in [S_P^{\mathcal{X}}(I)]_{\mathcal{X}}$
iff $(A \leftarrow d|A_1, \ldots, A_n) \in P$, $c = d \wedge \bigwedge_{i=1}^{n} c_i$, $\mathcal{X} \models c\rho$ and
$\{c_1|A_1, \ldots, c_n|A_n\} \subset I$
iff $(A \leftarrow d|A_1, \ldots, A_n) \in P$, $c = d \wedge \bigwedge_{i=1}^{n} c_i$, $\mathcal{X} \models c\rho$ and
$\{A_1\rho, \ldots, A_n\rho\} \subset [I]_{\mathcal{X}}$ iff $A\rho \in T_P^{\mathcal{X}}([I]_{\mathcal{X}})$. □

Continuity of $S_P^{\mathcal{X}}$ operator

Proposition

$S_P^{\mathcal{X}}$ *is continuous.*

Proof.

Let X be a chain of constrained interpretations. $c|A \in S_P^{\mathcal{X}}(lub(X))$,
iff $(A \leftarrow d|A_1, \ldots, A_n) \in P$, $c = d \wedge \bigwedge_{i=1}^{n} c_i$, $\mathcal{X} \models \exists(c)$ and $\{c_1|A_1, \ldots, c_n|A_n\} \subset lub(X)$.
iff $(A \leftarrow d|A_1, \ldots, A_n) \in P$, $c = d \wedge \bigwedge_{i=1}^{n} c_i$, $\mathcal{X} \models \exists(c)$ and $\{c_1|A_1, \ldots, c_n|A_n\} \subset I$, for some $I \in X$ (as X is a chain)
iff $c|A \in S_P^{\mathcal{X}}(I)$ for some $I \in X$, iff $c|A \in lub(S_P^{\mathcal{X}}(X))$. $\square$

Corollary

$S_P^{\mathcal{X}}$ *admits a least (post) fixed point* $F_2(P) = lfp(S_P^{\mathcal{X}}) = S_P^{\mathcal{X}} \uparrow \omega$.

Example CLP($\mathcal{H}$)

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

Example

$$
\begin{aligned}
S_P^{\mathcal{H}} \uparrow 0 &= \emptyset \\
S_P^{\mathcal{H}} \uparrow 1 &= \{A = [], B = C | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 2 &= S_P^{\mathcal{H}} \uparrow 1 \cup \\
&\{A = [X|L], C = [X|R], L = [], B = R | append(A, B, C)\} \\
&= S_P^{\mathcal{H}} \uparrow 1 \cup \{A = [X], C = [X|B] | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 3 &= S_P^{\mathcal{H}} \uparrow 2 \cup \\
&\{A = [X, Y], C = [X, Y|B] | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 4 &= S_P^{\mathcal{H}} \uparrow 3 \cup \\
&\{A = [X, Y, Z], C = [X, Y, Z|B] | append(A, B, C)\} \\
\ldots &= \ldots
\end{aligned}
$$

Example CLP($\mathcal{H}$)

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

Example

$$
\begin{aligned}
S_P^{\mathcal{H}} \uparrow 0 &= \emptyset \\
S_P^{\mathcal{H}} \uparrow 1 &= \{A = [], B = C | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 2 &= S_P^{\mathcal{H}} \uparrow 1 \cup \\
&\{A = [X|L], C = [X|R], L = [], B = R | append(A, B, C)\} \\
&= S_P^{\mathcal{H}} \uparrow 1 \cup \{A = [X], C = [X|B] | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 3 &= S_P^{\mathcal{H}} \uparrow 2 \cup \\
&\{A = [X, Y], C = [X, Y|B] | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 4 &= S_P^{\mathcal{H}} \uparrow 3 \cup \\
&\{A = [X, Y, Z], C = [X, Y, Z|B] | append(A, B, C)\} \\
\ldots &= \ldots
\end{aligned}
$$

Example CLP($\mathcal{H}$)

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

Example

$$
\begin{aligned}
S_P^{\mathcal{H}} \uparrow 0 &= \emptyset \\
S_P^{\mathcal{H}} \uparrow 1 &= \{A = [], B = C | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 2 &= S_P^{\mathcal{H}} \uparrow 1 \cup \\
&\quad \{A = [X|L], C = [X|R], L = [], B = R | append(A, B, C)\} \\
&= S_P^{\mathcal{H}} \uparrow 1 \cup \{A = [X], C = [X|B] | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 3 &= S_P^{\mathcal{H}} \uparrow 2 \cup \\
&\quad \{A = [X, Y], C = [X, Y|B] | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 4 &= S_P^{\mathcal{H}} \uparrow 3 \cup \\
&\quad \{A = [X, Y, Z], C = [X, Y, Z|B] | append(A, B, C)\} \\
\ldots &= \ldots
\end{aligned}
$$

Example CLP($\mathcal{H}$)

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

Example

$$
\begin{aligned}
S_P^{\mathcal{H}} \uparrow 0 &= \emptyset \\
S_P^{\mathcal{H}} \uparrow 1 &= \{A = [], B = C | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 2 &= S_P^{\mathcal{H}} \uparrow 1 \cup \\
&\quad \{A = [X|L], C = [X|R], L = [], B = R | append(A, B, C)\} \\
&= S_P^{\mathcal{H}} \uparrow 1 \cup \{A = [X], C = [X|B] | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 3 &= S_P^{\mathcal{H}} \uparrow 2 \cup \\
&\quad \{A = [X, Y], C = [X, Y|B] | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 4 &= S_P^{\mathcal{H}} \uparrow 3 \cup \\
&\quad \{A = [X, Y, Z], C = [X, Y, Z|B] | append(A, B, C)\} \\
\ldots &= \ldots
\end{aligned}
$$

Example CLP($\mathcal{H}$)

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

Example

$$
\begin{aligned}
S_P^{\mathcal{H}} \uparrow 0 &= \emptyset \\
S_P^{\mathcal{H}} \uparrow 1 &= \{A = [], B = C | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 2 &= S_P^{\mathcal{H}} \uparrow 1 \cup \\
&\quad \{A = [X|L], C = [X|R], L = [], B = R | append(A, B, C)\} \\
&= S_P^{\mathcal{H}} \uparrow 1 \cup \{A = [X], C = [X|B] | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 3 &= S_P^{\mathcal{H}} \uparrow 2 \cup \\
&\quad \{A = [X, Y], C = [X, Y|B] | append(A, B, C)\} \\
S_P^{\mathcal{H}} \uparrow 4 &= S_P^{\mathcal{H}} \uparrow 3 \cup \\
&\quad \{A = [X, Y, Z], C = [X, Y, Z|B] | append(A, B, C)\} \\
\ldots &= \ldots
\end{aligned}
$$

Relating $S_P^{\mathcal{X}}$ and $T_P^{\mathcal{X}}$ operators

Theorem ([JL87])

For every ordinal α, $T_P^{\mathcal{X}} \uparrow \alpha = [S_P^{\mathcal{X}} \uparrow \alpha]_{\mathcal{X}}$.

Proof.

The base case $\alpha = 0$ is trivial. For a successor ordinal, we have

$$\begin{array}{rcl} [S_P^{\mathcal{X}} \uparrow \alpha]_{\mathcal{X}} & = & [S_P^{\mathcal{X}}(S_P^{\mathcal{X}} \uparrow \alpha - 1)]_{\mathcal{X}} \\ & = & T_P^{\mathcal{X}}([S_P^{\mathcal{X}} \uparrow \alpha - 1]_{\mathcal{X}}) \text{ by Prop. 11} \\ & = & T_P^{\mathcal{X}}(T_P^{\mathcal{X}} \uparrow \alpha - 1) \text{ by induction} \\ & = & T_P^{\mathcal{X}} \uparrow \alpha. \end{array}$$

For a limit ordinal, we have

$$\begin{array}{rcl} [S_P^{\mathcal{X}} \uparrow \alpha]_{\mathcal{X}} & = & [\bigcup_{\beta<\alpha} S_P^{\mathcal{X}} \uparrow \beta]_{\mathcal{X}} \\ & = & \bigcup_{\beta<\alpha}[S_P^{\mathcal{X}} \uparrow \beta]_{\mathcal{X}} \\ & = & \bigcup_{\beta<\alpha} T_P^{\mathcal{X}} \uparrow \beta \text{ by induction} \\ & = & T_P^{\mathcal{X}} \uparrow \alpha \end{array}$$

□

Full abstraction w.r.t. computed constraints

Theorem (Theorem of full abstraction [GL91])

$O_{ca}(P) = F_2(P)$.

$F_2(P) \subseteq O_{ca}(P)$ is proved by induction on the powers n of $S_P^{\mathcal{X}}$. $n = 0$ is trivial. Let $c|A \in S_P^{\mathcal{X}} \uparrow n$, there exists a rule $(A \leftarrow d|A_1, ..., A_n) \in P$, s.t. $\{c_1|A_1, ..., c_n|A_n\} \subseteq S_P^{\mathcal{X}} \uparrow n-1$, $c = d \wedge \bigwedge_{i=1}^n c_i$ and $\mathcal{X} \models \exists c$. By induction $\{c_1|A_1, ..., c_n|A_n\} \subseteq O_{ca}(P)$. By definition of O_{ca} we get $c|A \in O_{ca}(P)$.

$O_{ca}(P) \subseteq F_2(P)$ is proved by induction on the length of derivations. Successes with derivation of length 0 are facts in $S_P^{\mathcal{X}} \uparrow 1$. Let $c|A \in O_{ca}(P)$ with a derivation of length n. By definition of O_{ca} there exists $(A \leftarrow d|A_1, ..., A_n) \in P$ s.t. $\{c_1|A_1, ..., c_n|A_n\} \subseteq O_{ca}(P)$, $c = d \wedge \bigwedge_{i=1}^n c_i$ and $\mathcal{X} \models \exists c$. By induction $\{c_1|A_1, ..., c_n|A_n\} \subseteq F_2(P)$. Hence by definition of $S_P^{\mathcal{X}}$ we get $c|A \in F_2(P)$.

Program analysis by abstract interpretation

$S_P^{\mathcal{H}} \uparrow \omega$ captures the set of computed answer constraints with P, nevertheless this set may be infinite and
it may contain too much information for proving some properties of the computed constraints.

Abstract interpretation [CC77] is a method for proving properties of programs without handling irrelevant information.

The idea is to replace the real computation domain by an abstract computation domain which retains sufficient information w.r.t. the property to prove.

Groundness analysis by abstract interpretation

Consider the CLP($\mathcal{H}$) append program

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

What is the groundness relation between arguments after a success?
The term structure can be abstracted by a boolean structure which expresses the groundness of the arguments.
We thus associate a CLP($\mathcal{B}$) abstract program:

```
append(A,B,C):- A=true, B=C.
append(A,B,C):- A=X/\L, C=X/\R, append(L,B,R).
```

Its least fixed point computed in at most 2^3 steps will express the groundness relation between arguments of the concrete program.

Groundness analysis by abstract interpretation

Consider the CLP($\mathcal{H}$) append program

```
append(A,B,C):- A=[], B=C.
append(A,B,C):- A=[X|L], C=[X|R], append(L,B,R).
```

What is the groundness relation between arguments after a success?
The term structure can be abstracted by a boolean structure which expresses the groundness of the arguments.
We thus associate a CLP($\mathcal{B}$) abstract program:

```
append(A,B,C):- A=true, B=C.
append(A,B,C):- A=X/\L, C=X/\R, append(L,B,R).
```

Its least fixed point computed in at most 2^3 steps will express the groundness relation between arguments of the concrete program.

Groundness analysis (continued)

$$
\begin{aligned}
S_P^{\mathcal{B}} \uparrow 0 &= \emptyset \\
S_P^{\mathcal{B}} \uparrow 1 &= \{A = true, B = C \mid append(A,B,C)\} \\
S_P^{\mathcal{B}} \uparrow 2 &= S_P^{\mathcal{B}} \uparrow 1 \cup \\
&\quad \{A = X \wedge L, C = X \wedge R, L = true, B = R \mid append(A,B,C)\} \\
&= S_P^{\mathcal{B}} \uparrow 1 \cup \{C = A \wedge B \mid append(A,B,C)\} \\
S_P^{\mathcal{B}} \uparrow 3 &= S_P^{\mathcal{B}} \uparrow 2 \cup \\
&\quad \{A = X \wedge L, C = X \wedge R, R = L \wedge B \mid append(A,B,C)\} \\
&= S_P^{\mathcal{B}} \uparrow 2 \cup \{C = A \wedge B \mid append(A,B,C)\} \\
&= S_P^{\mathcal{B}} \uparrow 2 = S_P^{\mathcal{B}} \uparrow \omega
\end{aligned}
$$

In a success of $append(A, B, C)$, C is ground if and only if A and B are ground.

Groundness analysis (continued)

$$
\begin{aligned}
S_P^{\mathcal{B}} \uparrow 0 &= \emptyset \\
S_P^{\mathcal{B}} \uparrow 1 &= \{A = true, B = C | append(A, B, C)\} \\
S_P^{\mathcal{B}} \uparrow 2 &= S_P^{\mathcal{B}} \uparrow 1 \cup \\
&\quad \{A = X \wedge L, C = X \wedge R, L = true, B = R | append(A, B, C)\} \\
&= S_P^{\mathcal{B}} \uparrow 1 \cup \{C = A \wedge B | append(A, B, C)\} \\
S_P^{\mathcal{B}} \uparrow 3 &= S_P^{\mathcal{B}} \uparrow 2 \cup \\
&\quad \{A = X \wedge L, C = X \wedge R, R = L \wedge B | append(A, B, C)\} \\
&= S_P^{\mathcal{B}} \uparrow 2 \cup \{C = A \wedge B | append(A, B, C)\} \\
&= S_P^{\mathcal{B}} \uparrow 2 = S_P^{\mathcal{B}} \uparrow \omega
\end{aligned}
$$

In a success of $append(A, B, C)$, C is ground if and only if A and B are ground.

Groundness analysis (continued)

$$
\begin{aligned}
S_P^{\mathcal{B}} \uparrow 0 &= \emptyset \\
S_P^{\mathcal{B}} \uparrow 1 &= \{A = true, B = C | append(A, B, C)\} \\
S_P^{\mathcal{B}} \uparrow 2 &= S_P^{\mathcal{B}} \uparrow 1 \cup \\
&\{A = X \wedge L, C = X \wedge R, L = true, B = R | append(A, B, C)\} \\
&= S_P^{\mathcal{B}} \uparrow 1 \cup \{C = A \wedge B | append(A, B, C)\} \\
S_P^{\mathcal{B}} \uparrow 3 &= S_P^{\mathcal{B}} \uparrow 2 \cup \\
&\{A = X \wedge L, C = X \wedge R, R = L \wedge B | append(A, B, C)\} \\
&= S_P^{\mathcal{B}} \uparrow 2 \cup \{C = A \wedge B | append(A, B, C)\} \\
&= S_P^{\mathcal{B}} \uparrow 2 = S_P^{\mathcal{B}} \uparrow \omega
\end{aligned}
$$

In a success of $append(A, B, C)$, C is ground if and only if A and B are ground.

Groundness analysis (continued)

$$
\begin{aligned}
S_P^{\mathcal{B}} \uparrow 0 &= \emptyset \\
S_P^{\mathcal{B}} \uparrow 1 &= \{A = true, B = C | append(A,B,C)\} \\
S_P^{\mathcal{B}} \uparrow 2 &= S_P^{\mathcal{B}} \uparrow 1 \cup \\
&\{A = X \wedge L, C = X \wedge R, L = true, B = R | append(A,B,C)\} \\
&= S_P^{\mathcal{B}} \uparrow 1 \cup \{C = A \wedge B | append(A,B,C)\} \\
S_P^{\mathcal{B}} \uparrow 3 &= S_P^{\mathcal{B}} \uparrow 2 \cup \\
&\{A = X \wedge L, C = X \wedge R, R = L \wedge B | append(A,B,C)\} \\
&= S_P^{\mathcal{B}} \uparrow 2 \cup \{C = A \wedge B | append(A,B,C)\} \\
&= S_P^{\mathcal{B}} \uparrow 2 = S_P^{\mathcal{B}} \uparrow \omega
\end{aligned}
$$

In a success of $append(A, B, C)$, C is ground if and only if A and B are ground.

Groundness analysis of reverse

Concrete CLP($\mathcal{H}$) program:

```
rev(A,B) :- A=[], B=[].
rev(A,B) :- A=[X|L], rev(L,K), append(K,[X],B).
```

Abstract CLP($\mathcal{B}$) program:

```
rev(A,B) :- A=true, B=true.
rev(A,B) :- A=X/\L, rev(L,K), append(K,X,B).
```

$$\begin{aligned}
S_P^{\mathcal{B}} \uparrow 0 &= \emptyset \\
S_P^{\mathcal{B}} \uparrow 1 &= \{A = true, B = true | rev(A, B)\} \\
S_P^{\mathcal{B}} \uparrow 2 &= S_P^{\mathcal{B}} \uparrow 1 \cup \{A = X, B = X | rev(A, B)\} \\
&= S_P^{\mathcal{B}} \uparrow 1 \cup \{A = B | rev(A, B)\} \\
S_P^{\mathcal{B}} \uparrow 3 &= S_P^{\mathcal{B}} \uparrow 2 \cup \{A = X \wedge L, L = K, B = K \wedge X | rev(A, B)\} \\
&= S_P^{\mathcal{B}} \uparrow 2 \cup \{A = B | rev(A, B)\} = S_P^{\mathcal{B}} \uparrow 2 = S_P^{\mathcal{B}} \uparrow \omega
\end{aligned}$$

Constraint-based Model Checking [DP99]

Analysis of unbounded states concurrent systems by CLP programs.
Concurrent transition systems defined by condition-action rules [Sha93]:

$$\text{condition } \phi(\vec{x}) \text{ action } \vec{x}' = \psi(\vec{x})$$

Translation into CLP clauses over one predicate p (for states)

$$p(\vec{x}) \leftarrow \phi(\vec{x}),\ \psi(\vec{x}', \vec{x}),\ p(\vec{x}').$$

The transitions of the concurrent system are in one-to-one correspondance to the CSLD derivations of the CLP program.

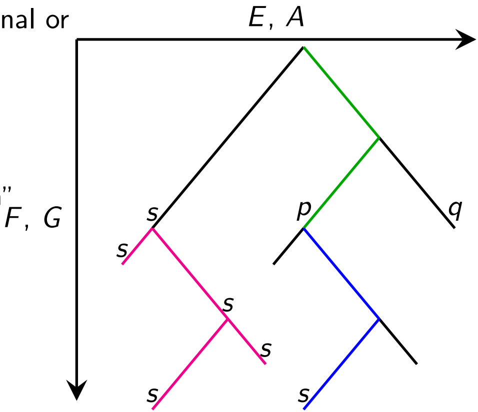
Proposition

The set of states from which a set of states defined by a constraint c is reachable is the set $lfp(T_P)$ where P is the CLP program plus the clause $p(\vec{x}) \leftarrow c(\vec{x})$.

Computation Tree Logic CTL

Temporal logic for branching time:

- States described by propositional or first-order formulas
- Two **path quantifiers** for non-determinism:
  - A "for all transition paths"
  - E "for some transition path"
- Several **temporal operators**:
  - X "next time",
  - F "eventually",
  - G "always",
  - U "until".



Model Checking

Two types of interesting properties:
$AG\neg\phi$ "Safety" property.
$AF\psi$ "Liveness" property.

Duality: for any formula ϕ we have
$EF\phi = \neg AG\neg\phi$ and
$EG\phi = \neg AF\neg\phi$.

Model checking is an algorithm for computing, in a given Kripke structure $K = (S, I, R)$, $I \subset S$, $R \subset S \times S$ (S is the set of states, I the initial states and R the transition relation), the set of states which satisfy a given CTL formula ϕ, i.e. the set $\{s \in S | K, s \models \phi\}$.

(Symbolic) Model Checking

Basic algorithm

When S is finite, represent K as a graph, and iteratively label the nodes with the subformulas of ϕ which are true in that node.

Add A to the states satisfying A ($\neg A$, $A \wedge B$,...)

Add $EF\phi$ ($EX\phi$) to the (immediate) predecessors of states labeled by ϕ

Add $E(\phi U \psi)$ to the predecessor states of ψ while they satisfy ϕ

Add $EG\phi$ to the states for which there exists a path leading to a non trivial strongly connected components of the subgraph restricted to the states satisfying ϕ

Symbolic model checking

Use OBDD's to represent states and transitions as boolean formulas (S is finite).

Constraint-based Model Checking

Constraint-based model checking [DP99] applies to Kripke structures with an infinite set of states.
Numerical constraints provide a finite representation for an infinite set of states.

Constraint logic programming theory:

$$EF(\phi) = lfp(T_{R \cup \{p(\vec{x}) \leftarrow \phi\}})$$

$$EG(\phi) = gfp(T_{R \wedge \phi})$$

Prototype implementation *DMC* in Sicstus Prolog + Simplex, CLP($\mathcal{H}, \mathcal{FD}, \mathcal{R}, \mathcal{B}$)

Part IV

Logical Semantics

Part IV: Logical Semantics

Logical Semantics of CLP($\mathcal{X}$) Programs

- Proper logical semantics

$$(1)\quad P, \mathcal{T} \models \exists(G) \qquad (4)\quad P, \mathcal{T} \models c \supset G,$$

- Logical semantics in a fixed pre-interpretation

$$(2)\quad P \models_{\mathcal{X}} \exists(G) \qquad (5)\quad P \models_{\mathcal{X}} c \supset G,$$

- Algebraic semantics

$$(3)\quad M_P^{\mathcal{X}} \models \exists(G) \qquad (6)\quad M_P^{\mathcal{X}} \models c \supset G.$$

We show (1) $\Leftrightarrow$ (2) $\Leftrightarrow$ (3) and (4) $\Rightarrow$ (5) $\Leftrightarrow$ (6).

Soundness of CSLD Resolution

Theorem ([JL87])

If c is a computed answer for the goal G then $M_P^{\mathcal{X}} \models c \supset G$, $P \models_{\mathcal{X}} c \supset G$ and $P, \mathcal{T} \models c \supset G$.

If $G = (d|A_1, ..., A_n)$, we deduce from the $\wedge$-compositionality lemma, that there exist computed answers $c_1, ..., c_n$ for the goals $A_1, ..., A_n$ such that $c = d \wedge \bigwedge_{i=1}^{n} c_i$ is satisfiable. For every $1 \leq i \leq n$
$c_i|A_i \in S_P^{\mathcal{X}} \uparrow \omega$, by the full abstraction Thm 16,
$[c_i|A_i]_{\mathcal{X}} \subseteq M_P^{\mathcal{X}}$, by Thm. 15, and Prop. 9, hence $M_P^{\mathcal{X}} \models \forall(c_i \supset A_i)$,
$P \models_{\mathcal{X}} \forall(c_i \supset A_i)$ as $M_P^{\mathcal{X}}$ is the least $\mathcal{X}$-model of P,
$P \models_{\mathcal{X}} \forall(c \supset A_i)$ as $\mathcal{X} \models \forall(c \supset c_i)$ for all i, $1 \leq i \leq n$.
Therefore we have $P \models_{\mathcal{X}} \forall(c \supset (d \wedge A_1 \wedge ... \wedge A_n))$,
and as the same reasoning applies to any model $\mathcal{X}$ of $\mathcal{T}$,
$P, \mathcal{T} \models \forall(c \supset (d \wedge A_1 \wedge ... \wedge A_n))$

Completeness of CSLD resolution

Theorem ([Mah87])

If $M_P^{\mathcal{X}} \models_{\mathcal{X}} c \supset G$ then there exists a set $\{c_i\}_{i \geq 0}$ of computed answers for G, such that: $\mathcal{X} \models \forall (c \supset \bigvee_{i \geq 0} \exists Y_i c_i)$.

Proof.

For every solution ρ of c, for every atom A_j in G,
$M_P^{\mathcal{X}} \models A_j\rho$ iff $A_j\rho \in T_P^{\mathcal{X}} \uparrow \omega$, by Thm. 8, iff $A_j\rho \in [S_P^{\mathcal{X}} \uparrow \omega]_{\mathcal{X}}$, by Thm. 15,
iff $c_{j,\rho}|A_j \in S_P^{\mathcal{X}} \uparrow \omega$, for some constraint $c_{j,\rho}$ s.t. ρ is solution of $\exists Y_{j,\rho} c_{j,\rho}$, where $Y_{j,\rho} = V(c_{j,\rho}) \setminus V(A_j)$,
iff $c_{j,\rho}$ is a computed answer for A_j (by 16) and $\mathcal{X} \models \exists Y_{j,\rho} c_{j,\rho}\rho$.
Let c_ρ be the conjunction of $c_{j,\rho}$ for all j. c_ρ is a computed answer for G.
By taking the collection of c_ρ for all ρ we get $\mathcal{X} \models \forall (c \supset \bigvee_{c_\rho} \exists Y_\rho c_\rho)$ □

Completeness w.r.t. the theory of the structure

Theorem ([Mah87])

If $P, \mathcal{T} \models c \supset G$ then there exists a finite set $\{c_1, ..., c_n\}$ of computed answers to G, such that:
$\mathcal{T} \models \forall(c \supset \exists Y_1 c_1 \vee ... \vee \exists Y_n c_n)$.

Proof.

If $P, \mathcal{T} \models c \supset G$ then for every model $\mathcal{X}$ of $\mathcal{T}$, for every $\mathcal{X}$-solution ρ of c, there exists a computed constraint $c_{\mathcal{X},\rho}$ for G s.t. $\mathcal{X} \models c_{\mathcal{X},\rho}\rho$. Let $\{c_i\}_{i\geq 0}$ be the set of these computed answers. Then for every model $\mathcal{X}$ and for every $\mathcal{X}$-valuation ρ, $\mathcal{X} \models c \supset \bigvee_{i\geq 1} \exists Y_i c_i$, therefore
$\mathcal{T} \models c \supset \bigvee_{i\geq 1} \exists Y_i c_i$,
As $\mathcal{T} \cup \{\exists(c \wedge \neg\exists Y_i c_i)\}_i$ is unsatisfiable, by applying the compactness theorem of first-order logic there exists a finite part $\{c_i\}_{1\leq i\leq n}$,
s.t. $\mathcal{T} \models c \supset \bigvee_{i=1}^{n} \exists Y_i c_i$. □

First-order theorem proving in CLP($\mathcal{H}$)

Prolog can be used to find proofs by refutation of Horn clauses (with a complete search meta-interpreter).
$P, \forall(\neg A)$ is unsatisfiable iff $P \models \exists(A)$ iff $A \longrightarrow^* \square$.

Groups can be axiomatized with Horn clauses with a ternary predicate $p(x, y, z)$ meaning $x * y = z$.

```
clause(p(e,X,X)).
clause(p(i(X),X,e)).
clause((p(U,Z,W) :- p(X,Y,U), p(Y,Z,V), p(X,V,W))).
clause((p(X,V,W) :- p(X,Y,U), p(Y,Z,V), p(U,Z,W))).
```

Theorem proving in groups

To show $i(i(x)) = x$ by refutation,
we show that the formula $\neg\forall x\ p(i(i(X)), e, X)$ is unsatisfiable
By Skolemization we get the goal clause $\neg p(i(i(a)), e, a)$

```
| ?- solve(p(i(i(a)),e,a)).
depth 2
yes
| ?- solve(p(a,e,a)).
depth 4
yes
| ?- solve(p(a,i(a),e)).
depth 3
yes
```

Theorem proving in groups (cont.)

To show that any non empty subset of a group, stable by division, is a subgroup we add two clauses

```
clause(s(a)).
clause((s(Z) :- s(X), s(Y), p(X,i(Y),Z))).
```

and prove that s contains e and $i(a)$.

```
| ?- solve(s(e)).
depth 4
yes
| ?- solve(s(i(a))).
depth 5
yes
```

Higher-order theorem proving in CLP(λ)

Church's simply typed λ-calculus

$t \ ::= \ v \mid t_1 \rightarrow t_2$

$e : t \ ::= \ x : t \mid (\lambda x : t_1.e : t_2) : t_1 \rightarrow t_2 \mid (e_1 : t_1 \rightarrow t_2(e_2 : t_1)) : t_2$

Theory of functionality

$\lambda x.e_1 =_\alpha \lambda y.e_1[y/x]$ if $y \notin V(e_1)$,

$(\lambda x.e_1)e_2 \ \rightarrow_\beta \ e_1[e_2/x]$

$=_\alpha . \rightarrow_\beta$ is terminating and confluent

$$e_1 \ =_{\alpha,\beta} \ e_2 \ \textit{iff} \ \downarrow_\beta e_1 \ =_\alpha \downarrow_\beta e_2.$$

Equality is decidable, but not unification...

Theorem proving in CLP(λ)

Theorem (Cantor's Theorem)

$\mathbb{N}^{\mathbb{N}}$ is not countable.

Proof.

By two steps of CSLD resolution!
Let us suppose $\exists h : \mathbb{N} \rightarrow (\mathbb{N} \rightarrow \mathbb{N})\ \forall f : \mathbb{N} \rightarrow \mathbb{N}\ \exists n : \mathbb{N}\ h(n) = f$
After Skolemisation we get $\forall F\ h(n(F)) = F$, i.e. $\forall F\ \neg h(n(F)) \neq F$.
Let us consider the following program

$$G \neq H \leftarrow G(N) \neq H(N).$$
$$N \neq s(N).$$

We have $h(n\ F) \neq F \ \longrightarrow^{\sigma_1}\ (h(n\ F))(I) \neq F(I) \ \longrightarrow^{\sigma_2}\ \square$
where the unifier $\sigma_2 = \{G = h\ I\ I,\ I = n(F),\ F = \lambda i.s(h\ i\ i),\ H = F\}$
is Cantor's diagonal argument! $\square$

Negation as Failure

A derivation CSLD is fair if every atom which appears in a goal of the derivation is selected after a finite number of resolution steps.
A fair CSLD tree for a goal G is a CSLD derivation tree for G in which all derivations are fair.
A goal G is finitely failed if G has a fair CSLD derivation tree to G, which is finite and which contains no success.

```
p :- p.

| ?- member(a,[b,c,d]).
no

| ?- p, member(a,[b,c,d]).
...
```

Logical semantics of finite failure?

Horn clauses entail no negative information: the Herbrand's base $\mathcal{B}_{\mathcal{X}}$ is a model.

On the other hand, the complement of the least $\mathcal{X}$-model $M_P^{\mathcal{X}}$ is not recursively enumerable.

Indeed let us suppose the opposite. We could define in Prolog the predicates:

- `success(P,B)` which succeeds iff $M_P \models \exists B$, i.e. if the goal B has a successful SLD derivation with the program P
- `fail(P,B)` which succeeds iff $M_P \models \neg\exists B$

Undecidability of $M_P^{\mathcal{X}}$

```
loop:- loop.
contr(P):- success(P,P), loop.
contr(P):- fail(P,P).
```

If `contr(contr)` has a success,
then `success(contr,contr)` succeeds,
and `fail(contr,contr)` doesn't succeed,
hence `contr(contr)` doesn't succeed: contradiction.

If `contr(contr)` doesn't succeed,
then `fail(contr,contr)` succeeds,
hence `contr(contr)` succeeds: contradiction.

Therefore programs `success` and `fail` cannot exist.

Clark's completion

The Clark's completion of P is the set P^* of formulas of the form
$\forall X\ p(X) \leftrightarrow (\exists Y_1 c_1 \wedge A_1^1 \wedge \ldots \wedge A_{n_1}^1) \vee \ldots \vee (\exists Y_k c_k \wedge A_1^k \wedge \ldots \wedge A_{n_k}^k)$
where the $p(X) \leftarrow c_i | A_1^i, \ldots, A_{n_i}^i$ are the rules in P and Y_i's the local variables,
$\forall X \neg p(X)$ if p is not defined in P.

Example

CLP($\mathcal{H}$) program `p(s(X)):- p(X).`
Clark's completion $P^* = \{\forall x\ p(x) \leftrightarrow \exists y\ x = s(y) \wedge p(y)\}$.
The goal `p(0)` finitely fails, we have $P^*, CET \models \neg p(0)$.
The goal `p(X)` doesn't finitely fail,
we have $P^*, CET \not\models \neg\exists X\ p(X)$ although $P^* \models_{\mathcal{H}} \neg\exists X\ p(X)$

Supported $\mathcal{X}$-models

Proposition

i) I is a supported $\mathcal{X}$-model of P iff ii) I is a $\mathcal{X}$-model of P^* iff iii) I is a fixed point of $T_P^{\mathcal{X}}$.

Proof.

I is a $\mathcal{X}$-model of P^*
iff I is a $\mathcal{X}$-model of $\forall X\ p(X) \leftarrow \phi_1 \vee \ldots \vee \phi_k$ for every formula $\forall X\ p(X) \leftrightarrow \phi_1 \vee \ldots \vee \phi_k$ in P^*,
iff I is a post-fixed point of $T_P^{\mathcal{X}}$, i.e. $.T_P^{\mathcal{X}}(I) \subseteq I$.
I is a supported $\mathcal{X}$-interpretation of P,
iff I is a $\mathcal{X}$-model of $\forall X\ p(X) \rightarrow \phi_1 \vee \ldots \vee \phi_k$ for every formula $\forall X\ p(X) \leftrightarrow \phi_1 \vee \ldots \vee \phi_k$ in P^*,
iff I is a pre-fixed point of $T_P^{\mathcal{X}}$, i.e. $I \subseteq T_P^{\mathcal{X}}(I)$.
Thus *i)* I is a supported $\mathcal{X}$-model of P iff *ii)* I is a $\mathcal{X}$-model of P^* iff *iii)* I is a fixed point of $T_P^{\mathcal{X}}$. $\square$

Models of the Clark's completion

Theorem

i) P^* has the same least $\mathcal{X}$-model than P, $M_P^{\mathcal{X}} = M_{P^*}^{\mathcal{X}}$
ii) $P \models_{\mathcal{X}} c \supset A$ iff $P^* \models_{\mathcal{X}} c \supset A$, for all c and A,
iii) $P, \mathcal{T} \models c \supset A$ iff $P^*, \mathcal{T} \models c \supset A$.

Proof.

i) is an immediate corollary of full abstraction and least $\mathcal{X}$-model theorems.
For iii) we clearly have $(P, \mathcal{T} \models c \supset A) \Rightarrow (P^*, \mathcal{T} \models c \supset A)$. We show the contrapositive of the opposite, $(P, \mathcal{T} \not\models c \supset A) \Rightarrow (P^*, \mathcal{T} \not\models c \supset A)$.
Let I be a model of P and $\mathcal{T}$, based on a structure $\mathcal{X}$, let ρ be a valuation such that $I \models \neg A\rho$ and $\mathcal{X} \models c\rho$.
We have $M_P^{\mathcal{X}} \models \neg A\rho$, thus $M_{P^*}^{\mathcal{X}} \models \neg A\rho$, and as $\mathcal{T} \models c\rho$, we conclude that $P^*, \mathcal{T} \not\models c \supset A$.
The proof of ii) is identical, the structure $\mathcal{X}$ being fixed. □

Soundness of Negation as Finite Failure

Theorem

If G is finitely failed then $P^*, \mathcal{T} \models \neg G$.

Proof.

By induction on the height h of the tree in finite failure for $G = c|A, \alpha$ where A is the selected atom at the root of the tree.
In the base case $h = 1$, the constrained atom $c|A$ has no CSLD transition, we can deduce that $P^*, \mathcal{T} \models \neg(c \wedge A)$ hence that $P^*, \mathcal{T} \models \neg G$.
For the induction step, let us suppose $h > 1$. Let $G_1, ..., G_n$ be the sons of the root and $Y_1, ..., Y_n$ be the respective sets of introduced variables.
We have $P^*, \mathcal{T} \models G \leftrightarrow \exists Y_1\ G_1 \vee ... \vee \exists_n\ G_n$. By induction hypothesis, $P^*, \mathcal{T} \models \neg G_i$ for every $1 \leq i \leq n$, therefore $P^*, \mathcal{T} \models \neg G$. □

Completeness of Negation as Failure

Theorem ([JL87])

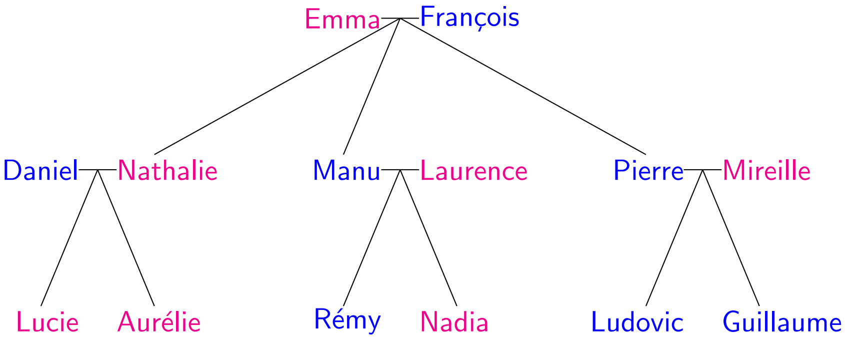
If $P^*, \mathcal{T} \models \neg G$ then G is finitely failed.

We show that if G is not finitely failed then $P^*, \mathcal{T}, \exists(G)$ is satisfiable. If G has a success then by the soundness of CSLD resolution, $P^*, \mathcal{T} \models \exists G$. Else G has a fair infinite derivation $G = c_0|G_0 \longrightarrow c_1|G1 \longrightarrow ...$

For every $i \geq 0$, c_i is $\mathcal{T}$-satisfiable, thus by the compactness theorem, $c_\omega = \bigcup_{i\geq 0} c_i$ is $\mathcal{T}$-satisfiable. Let $\mathcal{X}$ be a model of $\mathcal{T}$ s.t. $\mathcal{X} \models \exists(c_\omega)$. Let $I_0 = \{A\rho \mid A \in G_i \text{ for some } i \geq 0 \text{ and } \mathcal{X} \models c_\omega\rho\}$. As the derivation is fair, every atom A in I_0 is selected, thus $c_\omega|A \longrightarrow c_\omega|A_1, ..., A_n$ with $[c_\omega|A] \cup ... \cup [c_\omega|A_n] \subseteq I_0$. We deduce that $I_0 \subseteq T_P^{\mathcal{X}}(I_0)$. By Knaster-Tarski's theorem, the iterated application up to ordinal ω of the operator $T_P^{\mathcal{X}}$ from I_0 leads to a fixed point I s.t. $I_0 \subseteq I$, thus $[c_\omega|G_0] \in I$. Hence $P^*, \exists(G)$ is $\mathcal{X}$-satisfiable, and $P^*, \mathcal{T}, \exists(G)$ is satisfiable.

Interlude

Short Practical Prolog Tutorial



Bibliography I

Patrick Cousot and Radhia Cousot.

Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints.

In *POPL'77: Proceedings of the 6th ACM Symposium on Principles of Programming Languages*, pages 238–252, New York, 1977. ACM Press.
Los Angeles.

Giorgio Delzanno and Andreas Podelski.

Model checking in clp.

In Rance Cleaveland, editor, *TACAS'99: Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, volume 1579 of *Lecture Notes in Computer Science*, pages 223–239. Springer-Verlag, 1999.

Maurizio Gabbrielli and Giorgio Levi.

Modeling answer constraints in constraint logic programs.

In K. Furukawa, editor, *Proceedings of ICLP'91, International Conference on Logic Programming*, pages 238–252, Cambridge, 1991. MIT Press.

Joxan Jaffar and Jean-Louis Lassez.

Constraint logic programming.

In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages, Munich, Germany*, pages 111–119. ACM, January 1987.

Bibliography II

Michael J. Maher.
Logic semantics for a class of committed-choice programs.
In *Proceedings of ICLP'87, International Conference on Logic Programming*, 1987.

Udaya A. Shankar.
An introduction to assertional reasoning for concurrent systems.
ACM Computing Surveys, 25(3):225–262, 1993.