Advanced Solver Technology

Helmut Seidl

TUM + DTU

2006

Part 1

Beyond Finiteness:

Strongly Recognizable Relations

Outline of Part 1:

- The Spi Calculus
- Normalizable Horn Clauses
- Efficient Subclasses
- Instantiation

1. The Spi Calculus

The Spi Calculus — Abadi, Gordon 1999

$\rightarrow$ allows formalization of (certain) cryptographic protocols;

$\rightarrow$ generalizes π by

- structured terms and
- explicit encryption / decryption.

Example:

$$\mathbf{send}(c_1, \{\mathsf{pair}(x_1, x_2)\}_k);$$
$$\mathbf{recv}(c_2, y);$$
$$\mathbf{case}\ y\ \mathbf{of}\ \{x\}_k; \mathbf{1}$$

Control-flow Analysis:

- Approximate possible values of variables;
- Approximate possible values sent through channels !

$\Longrightarrow$

Who knows what?

Natural Formulation of the Analysis:

Horn Clauses

terms	program parts values
predicates	*reach/1* *occurs/1* *hasValue/2* *hasMessage/2*

Decryption:

$$
\begin{aligned}
reach(\mathsf{prog}) \quad & \Leftarrow \\
hasValue(X, Y) \quad & \Leftarrow \quad reach(\mathsf{decrypt}(E, X, K, T)) \wedge \\
& \qquad hasValue(K, V) \wedge \\
& \qquad hasValue(E, \mathsf{enc}(Y, V))
\end{aligned}
$$

$$\ldots$$

$$\begin{array}{lcl}
occurs(E) & \Leftarrow & reach(\mathsf{decrypt}(E, _, _, _)) \\
reach(T) & \Leftarrow & reach(\mathsf{decrypt}(E, _, K, T)) \wedge \\
 & & hasValue(K, V) \wedge \\
 & & hasValue(E, \mathsf{enc}(_, V))
\end{array}$$

2. Normalizable Horn Clauses

Desired Properties:

- Relational formulation;
- Possibly infinite least model;
- Effective construction.

Comparison:

Set Constraints: Heintze, Jaffar 1990

$$\begin{aligned} B &\supseteq a(A, A) \\ C &\supseteq B \cap \pi_{b,2}(A) \end{aligned}$$

Comparison:

Set Constraints: Heintze, Jaffar 1990

$$\begin{array}{rcl} B & \supseteq & a(A, A) \\ C & \supseteq & B \cap \pi_{b,2}(A) \end{array}$$

can be simulated by:

$$\begin{array}{rcl} \mathsf{aux}_B(a(X_1, X_2)) & \Leftarrow & \mathsf{aux}_A(X_1) \wedge \mathsf{aux}_A(X_2) \\ \mathsf{aux}_C(X) & \Leftarrow & \mathsf{aux}_B(X) \wedge \mathsf{aux}_A(b(\_, X)) \\ & & \text{:-)} \end{array}$$

Uniform Horn Clauses:

Frühwirth, Shapiro, Vardi, Yardeni 1991

$$\begin{array}{lcl} p(a(X_1,\ldots,X_k)) & \Leftarrow & p_1(X_1)\wedge\ldots\wedge p_k(X_k) \\ p(X) & \Leftarrow & q_1(t_1)\wedge\ldots\wedge q_m(t_m) \\ p(c) & \Leftarrow & q_1(t_1)\wedge\ldots\wedge q_m(t_m) \end{array}$$

Clauses which are not uniform:

$$
\begin{array}{lcll}
p(a(X,Y)) & \Leftarrow & q(b(X,Y)) & \text{(relabeling)} \\
p(X,Y) & \Leftarrow & e(X,Z) \wedge p(Z,Y) & \text{(composition)} \\
p(X,Y,Z) & \Leftarrow & q(Y,X,Z) & \text{(permutation)} \\
p(X,Z) & \Leftarrow & q(X,Y,Z) & \text{(general projection)}
\end{array}
$$

The Class $\mathcal{H}1$:

Chr. Weidenbach 1999

Nielson, Nielson, Seidl 2002

$$\begin{array}{lcl} p(a(X_1, \ldots, X_k)) & \Leftarrow & any \\ p(X_1, \ldots, X_k) & \Leftarrow & any \end{array}$$

Examples:

$$
\begin{array}{lcll}
p(a(X,Y)) & \Leftarrow & q(b(X,Y)) & \text{(relabeling)} \\
p(X,Y) & \Leftarrow & e(X,Z) \wedge p(Z,Y) & \text{(composition)} \\
p(X,Y,Z) & \Leftarrow & q(Y,X,Z) & \text{(permutation)} \\
p(X,Z) & \Leftarrow & q(X,Y,Z) & \text{(general projection)}
\end{array}
$$

... all are $\mathcal{H}$*1* :-)

Theorem:

Assume c is $\mathcal{H}$*1* with least model $\mathcal{H}$. Then

(1) $\mathcal{H}(p)$ is strongly recognizable;

(2) $\mathcal{H}$ can be computed in deterministic exponential time.

Strongly Recognizable Relation $=$

finite union of Cartesian products
of recognizable tree languages

Closure under:

- Boolean operations;
- Cartesian product;
- transitive closure.

Idea: Normalization

Goal: Automata Form:

$$\begin{array}{rcl} p(a(Z_1, \ldots, Z_k)) & \Leftarrow & p_1(X_1) \wedge \ldots \wedge p_n(X_n) \\ p(Z_1, \ldots, Z_k) & \Leftarrow & p_1(X_1) \wedge \ldots \wedge p_n(X_n) \\ & & \text{where all } X_i \text{ are among } Z_j \end{array}$$

Idea: Normalization

Goal: Automata Form:

$$\begin{array}{rcl} p(a(Z_1,\ldots,Z_k)) & \Leftarrow & p_1(X_1) \wedge \ldots \wedge p_n(X_n) \\ p(Z_1,\ldots,Z_k) & \Leftarrow & p_1(X_1) \wedge \ldots \wedge p_n(X_n) \\ & & \text{where all } X_i \text{ are among } Z_j \end{array}$$

Technique:

$\Longrightarrow$ Add simpler implied rules until saturation :-)

$\Longrightarrow$ All non-automata clauses then are redundant :-)

$\Longrightarrow$ All non-automata clauses can be removed :-)

Adding Simpler Clauses (0):

Expanding univeral predicates:

For:

$$p(X) \quad \Leftarrow$$

add all clauses:

$$p(a(Z_1, \ldots, Z_k)) \quad \Leftarrow \quad p(Z_1) \wedge \ldots \wedge p(Z_k)$$

for every constructor a :-)

Adding Simpler Clauses (1):

Resolving Complex Queries:

$$
\begin{array}{rcl}
p(X,Y) & \Leftarrow & q(a(\ b(X)\ ,\ Z\ ))\ \wedge h(Z) \\
q(a(X_1,X_2)) & \Leftarrow & q_1(X_1) \wedge q_2(X_2)
\end{array}
$$

Adding Simpler Clauses (1):

Resolving Complex Queries:

$$\begin{array}{rcl} p(X,Y) & \Leftarrow & \boxed{q(a(\ b(X)\ ,\ Z\ ))} \wedge h(Z) \\ \boxed{q(a(X_1,X_2))} & \Leftarrow & q_1(X_1) \wedge q_2(X_2) \end{array}$$

Adding Simpler Clauses (1):

Resolving Complex Queries:

$$
\begin{array}{rcl}
p(X,Y) & \Leftarrow & \boxed{q(a(\ b(X)\ ,\ Z\ ))} \wedge h(Z) \\
\boxed{q(a(X_1, X_2))} & \Leftarrow & q_1(X_1) \wedge q_2(X_2)
\end{array}
$$

resolves to:

$$p(X,Y) \;\Leftarrow\; q_1(\ b(X)\ ) \wedge q_2(\ Z\ ) \wedge h(Z)$$

Adding Simpler Clauses (1):

Resolving Complex Queries:

$$
\begin{array}{rcl}
p(X,Y) & \Leftarrow & \boxed{q(a(\boxed{b(X)},\boxed{Z}))} \wedge h(Z) \\
\boxed{q(a(X_1,X_2))} & \Leftarrow & q_1(X_1) \wedge q_2(X_2)
\end{array}
$$

resolves to:

$$
p(X,Y) \quad \Leftarrow \quad q_1(\boxed{b(X)}) \wedge q_2(\boxed{Z}) \wedge h(Z)
$$

Adding Simpler Clauses (1):

Resolving Complex Queries:

$$\begin{array}{rcl} p(X,Y) & \Leftarrow & \boxed{q(a(\boxed{b(X)}, \boxed{Z}))} \wedge h(Z) \\ \boxed{q(a(X_1, X_2))} & \Leftarrow & q_1(X_1) \wedge q_2(X_2) \end{array}$$

resolves to:

$$p(X,Y) \Leftarrow q_1(\boxed{b(X)}) \wedge q_2(\boxed{Z}) \wedge h(Z)$$

- Unification does not introduce new bindings in the first clause.
- The variables in the automata clause are instantiated to sub-terms ...

$\Longrightarrow$ Every new clause has smaller terms :-)

Adding Simpler Clauses (2):

Resolving with a One-variable Clause:

$$
\begin{array}{rcl}
p(X) & \Leftarrow & q(X) \ \wedge h(X) \\
q(\ a(X_1, X_2)\ ) & \Leftarrow & q_1(X_1) \wedge q_2(X_2)
\end{array}
$$

Adding Simpler Clauses (2):

Resolving with a One-variable Clause:

$$
\begin{aligned}
p(X) \quad &\Leftarrow \quad \boxed{q(X)} \wedge h(X) \\
\boxed{q(\ a(X_1, X_2)\ )} \quad &\Leftarrow \quad q_1(X_1) \wedge q_2(X_2)
\end{aligned}
$$

Adding Simpler Clauses (2):

Resolving with a One-variable Clause:

$$
\begin{aligned}
p(X) \quad &\Leftarrow \quad \boxed{q(X)} \wedge h(X) \\
\boxed{q(\ a(X_1, X_2)\ )} \quad &\Leftarrow \quad q_1(X_1) \wedge q_2(X_2)
\end{aligned}
$$

resolves to:

$$
p(\ a(X_1, X_2)\ ) \quad \Leftarrow \quad q_1(X_1) \wedge q_2(X_2) \wedge h(\ a(X_1, X_2)\ )
$$

Adding Simpler Clauses (2):

Resolving with a One-variable Clause:

$$\begin{array}{rcl} p(X) & \Leftarrow & \boxed{q(X)} \wedge h(X) \\ \boxed{q(\boxed{a(X_1, X_2)})} & \Leftarrow & q_1(X_1) \wedge q_2(X_2) \end{array}$$

resolves to:

$$p(\boxed{a(X_1, X_2)}) \quad \Leftarrow \quad q_1(X_1) \wedge q_2(X_2) \wedge h(\boxed{a(X_1, X_2)})$$

Adding Simpler Clauses (2):

Resolving with a One-variable Clause:

$$\begin{array}{rcl} p(X) & \Leftarrow & \boxed{q(X)} \wedge h(X) \\ \boxed{q(\boxed{a(X_1, X_2)})} & \Leftarrow & q_1(X_1) \wedge q_2(X_2) \end{array}$$

resolves to:

$$p(\boxed{a(X_1, X_2)}) \quad \Leftarrow \quad q_1(X_1) \wedge q_2(X_2) \wedge h(\boxed{a(X_1, X_2)})$$

- The single variable is instantiated with the constructor term of the automata clause.
- New terms in the body are subsequently removed :-)

Adding Simpler Clauses (3):

Splitting:

$$p(X) \quad \Leftarrow \quad q(X) \wedge \ h_1(Z) \wedge h_2(Z)$$

Adding Simpler Clauses (3):

Splitting:

$$p(X) \quad \Leftarrow \quad q(X) \wedge \boxed{h_1(Z) \wedge h_2(Z)}$$

Adding Simpler Clauses (3):

Splitting:

$$p(X) \quad \Leftarrow \quad q(X) \land \boxed{h_1(Z) \land h_2(Z)}$$

is split into:

$$\begin{aligned} p(X) \quad &\Leftarrow \quad q(X) \land \mathsf{flag}() \\ \mathsf{flag}() \quad &\Leftarrow \quad \boxed{h_1(Z) \land h_2(Z)} \end{aligned}$$

Adding Simpler Clauses (3):

Splitting:

$$p(X) \quad \Leftarrow \quad q(X) \wedge \boxed{h_1(Z) \wedge h_2(Z)}$$

is split into:

$$\begin{aligned} p(X) \quad &\Leftarrow \quad q(X) \wedge \mathsf{flag}() \\ \mathsf{flag}() \quad &\Leftarrow \quad \boxed{h_1(Z) \wedge h_2(Z)} \end{aligned}$$

If $\boxed{h_1(Z) \wedge h_2(Z)}$ is satisfiable, i.e., the intersection of languages accepted by the corresponding automata states is non-empty, we add the clause:

$$p(X) \quad \Leftarrow \quad q(X)$$

Discussion:

- Normalization results in a set of automata clauses :-)
- The automata representation allows to decide satisfiability of queries such as:

$$\Leftarrow\ p(X, a(X, b(Y, c, Y))) \wedge q(c(X, X))$$

  ... and also provide candidate substitutions :-)
- $\mathcal{H}1$-clauses can be used to approximate general Horn clauses ...

Example:

$$p(a(\ b(X)\ , Y, X)) \quad \Leftarrow \quad \textit{blabla}$$

Example:

$$p(a(\boxed{b(X)}, Y, X)) \quad \Leftarrow \quad \textit{blabla}$$

Example:

$$p(a(\boxed{b(X)}, Y, X)) \quad \Leftarrow \quad \textit{blabla}$$

can be approximated with:

$$\begin{aligned} \mathsf{aux}(\boxed{b(X)}) \quad &\Leftarrow \quad \textit{blabla} \\ p(a(Z, Y, X)) \quad &\Leftarrow \quad \mathsf{aux}(Z) \wedge \textit{blabla} \end{aligned}$$

Example:

$$p(a(\boxed{b(X)}, Y, X)) \quad \Leftarrow \quad \textit{blabla}$$

can be approximated with:

$$\begin{array}{rcr} \mathsf{aux}(\boxed{b(X)}) & \Leftarrow & \textit{blabla} \\ p(a(Z, Y, X)) & \Leftarrow & \mathsf{aux}(Z) \wedge \textit{blabla} \end{array}$$

The least model for the approximative set of clauses provides super-sets of tuples for every given predicate ...

$\Longrightarrow$

If a query is now unsatisfiable, then it was originally unsatisfiable :-)

If it is satisfiable, we know nothing :-(

3. Polynomial Subclasses

The Problem with $\mathcal{H}1$:

Construction of least model is DEXPTIME-hard.

Frühwirth et al. 1990, Seidl 1994

The Restriction $\mathcal{H}2$:

Head variables occur in pre-conditions at most once :-)

Examples:

$$
\begin{array}{llll}
p(a(X,Y)) & \Leftarrow & q(b(X,Y)) & \text{(relabeling)} \\
p(X,Y) & \Leftarrow & e(X,Z) \wedge p(Z,Y) & \text{(composition)} \\
p(X,Y,Z) & \Leftarrow & q(Y,X,Z) & \text{(permutation)} \\
p(X,Z) & \Leftarrow & q(X,Y,Z) & \text{(general projection)}
\end{array}
$$

... all are $\mathcal{H}2$:-))

Theorem:

Let c an $\mathcal{H}2$-clause where k is the max. number of variable occurrences in implications.

Then the least model of c can be computed in time

$$|c|^{\mathcal{O}(k)}$$

Corollary:

The transitive closure of a strongly recognizable binary tree relation can be computed in polynomial time.

Proof:

$$
\begin{array}{rcl}
trans(X,Y) & \Leftarrow & edge(X,Y) \\
trans(X,Z) & \Leftarrow & edge(X,Y) \wedge trans(Y,Z)
\end{array}
$$

$\square$

The Problem with $\mathcal{H}2$:

- The exponent can be large :-(
- Not all implications of Spi analysis satisfy $\mathcal{H}2$:-((

Example: Expression Evaluation

$$occurs(E_1) \quad \Leftarrow \quad occurs(\mathsf{pair}(E_1, E_2))$$

$$occurs(E_2) \quad \Leftarrow \quad occurs(\mathsf{pair}(E_1, E_2))$$

$$\begin{aligned} & hasValue(\mathsf{pair}(E_1, E_2), \mathsf{pair}(V_1, V_2)) \\ & \quad \Leftarrow \quad occurs(\mathsf{pair}(E_1, E_2)) \wedge \\ & \qquad\qquad hasValue(E_1, V_1) \wedge \\ & \qquad\qquad hasValue(E_2, V_2) \end{aligned}$$

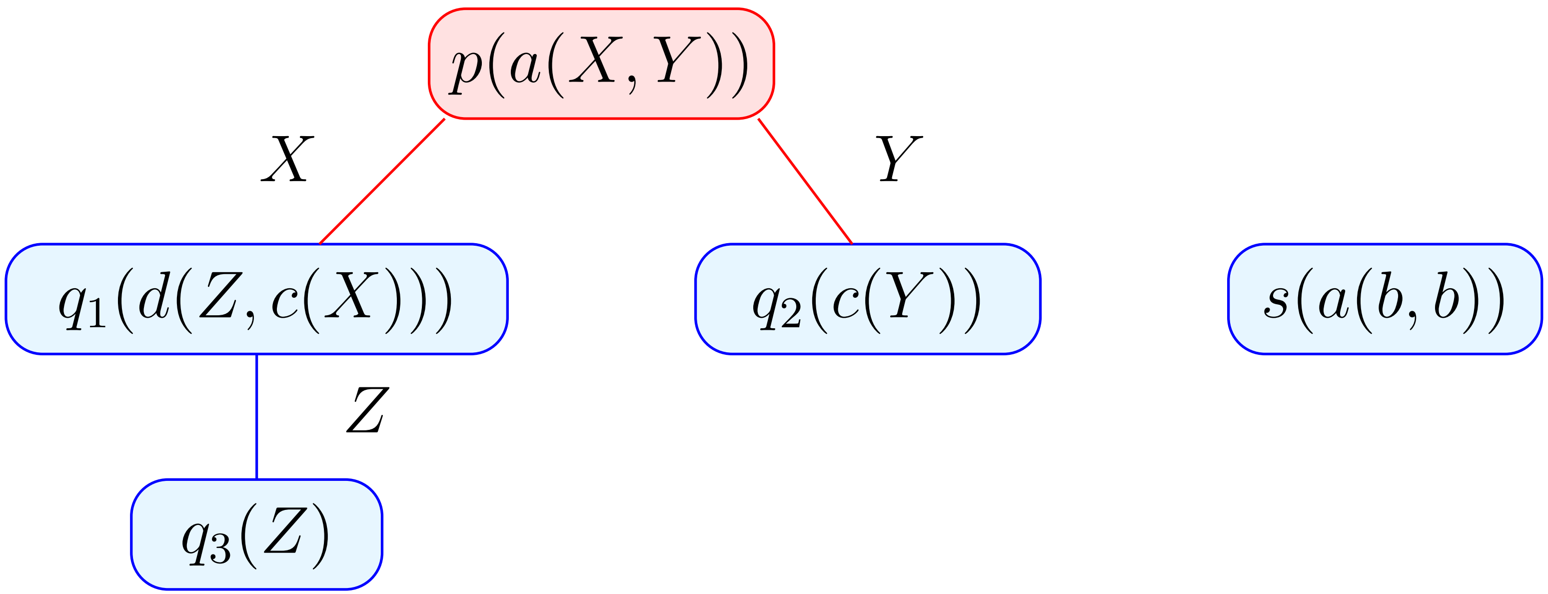
... is not even $\mathcal{H}1$:-(

Forgetting about Expressivity: $\mathcal{H}3$

All variable dependences are acyclic.

Example:

$$
\begin{aligned}
p(a(X,Y)) \quad &\Leftarrow \quad q_1(d(Z,c(X))) \wedge \\
& \quad q_2(c(Y)) \wedge \\
& \quad q_3(Z) \wedge \\
& \quad s(a(b,b))
\end{aligned}
$$



node	:	literal
edge	:	common variable

Idea:

- Apply normalization as in the general case :-)
- Introduce auxiliary predicates to break very complex right-hand sides into simple ones before-hand ...

A Simple Clause:

$$\mathsf{get}_1(X_1) \Leftarrow q_1(d(Z, X_1)) \wedge q_3(Z)$$

Resolving Simple Clauses:

$$
\begin{array}{rcl}
\mathsf{get}_1(X_1) & \Leftarrow & q_1(d(Z, X_1)) \ \wedge q_3(Z) \\
q_1(d(Z, X_1)) & \Leftarrow & h_1(Z) \wedge h_2(X_1)
\end{array}
$$

Resolving Simple Clauses:

$$\begin{aligned}
\mathsf{get}_1(X_1) &\Leftarrow \boxed{q_1(d(Z, X_1))} \wedge q_3(Z) \\
\boxed{q_1(d(Z, X_1))} &\Leftarrow h_1(Z) \wedge h_2(X_1)
\end{aligned}$$

Resolving Simple Clauses:

$$
\begin{aligned}
\mathsf{get}_1(X_1) &\Leftarrow \boxed{q_1(d(Z, X_1))} \wedge q_3(Z) \\
\boxed{q_1(d(Z, X_1))} &\Leftarrow h_1(Z) \wedge h_2(X_1)
\end{aligned}
$$

gives:

$$
\mathsf{get}_1(X_1) \Leftarrow h_2(X_1) \wedge\ h_2(Z) \wedge q_3(Z)
$$

Resolving Simple Clauses:

$$\begin{array}{lcl} \mathsf{get}_1(X_1) & \Leftarrow & \boxed{q_1(d(Z, X_1))} \wedge q_3(Z) \\ \boxed{q_1(d(Z, X_1))} & \Leftarrow & h_1(Z) \wedge h_2(X_1) \end{array}$$

gives:

$$\mathsf{get}_1(X_1) \quad \Leftarrow \quad h_2(X_1) \wedge \boxed{h_2(Z) \wedge q_3(Z)}$$

$\Longrightarrow$ Resolving simple clauses introduces only binary intersections :-)

$\Longrightarrow$ After splitting these intersections, very simple propagation rules remain :-))

Therefore, we translate:

$$p(a(X,Y)) \quad \Leftarrow \quad q_1(d(Z,\ c(X)\ )) \ \wedge \ q_2(c(Y)) \ \wedge \ q_3(Z)$$

into:

$$p(a(X,Y)) \quad \Leftarrow \quad \mathsf{aux}_X(X) \wedge \mathsf{aux}_Y(Y)$$

Therefore, we translate:

$$p(a(X,Y)) \quad \Leftarrow \quad q_1(d(Z,\ c(X)\ ))\ \wedge \boxed{q_2(c(Y))} \wedge\ q_3(Z)$$

into:

$$p(a(X,Y)) \quad \Leftarrow \quad \mathsf{aux}_X(X) \wedge \mathsf{aux}_Y(Y)$$

Therefore, we translate:

$$p(a(X, Y)) \quad \Leftarrow \quad q_1(d(Z,\ c(X)\ ))\ \wedge \boxed{q_2(c(Y))} \wedge\ q_3(Z)$$

into:

$$p(a(X, Y)) \quad \Leftarrow \quad \mathsf{aux}_X(X) \wedge \mathsf{aux}_Y(Y)$$

$$\mathsf{aux}_Y(Y) \qquad \Leftarrow \quad \boxed{q_2(c(Y))}$$

Therefore, we translate:

$$p(a(X,Y)) \quad \Leftarrow \quad q_1(d(Z, \boxed{c(X)})) \wedge \boxed{q_2(c(Y))} \wedge q_3(Z)$$

into:

$$p(a(X,Y)) \quad \Leftarrow \quad \mathsf{aux}_X(X) \wedge \mathsf{aux}_Y(Y)$$

$$\mathsf{aux}_Y(Y) \quad \Leftarrow \quad \boxed{q_2(c(Y))}$$

Therefore, we translate:

$$p(a(X,Y)) \quad \Leftarrow \quad q_1(d(Z, \boxed{c(X)})) \ \wedge \boxed{q_2(c(Y))} \wedge \ q_3(Z)$$

into:

$$\begin{array}{lcl} p(a(X,Y)) & \Leftarrow & \mathsf{aux}_X(X) \wedge \mathsf{aux}_Y(Y) \\ \mathsf{aux}_X(X) & \Leftarrow & \boxed{\mathsf{get}_1(c(X))} \\ \\ \mathsf{aux}_Y(Y) & \Leftarrow & \boxed{q_2(c(Y))} \end{array}$$

Therefore, we translate:

$$p(a(X,Y)) \quad \Leftarrow \quad \boxed{q_1(d(Z, \boxed{c(X)}))} \wedge \boxed{q_2(c(Y))} \wedge \boxed{q_3(Z)}$$

into:

$$p(a(X,Y)) \quad \Leftarrow \quad \mathsf{aux}_X(X) \wedge \mathsf{aux}_Y(Y)$$

$$\mathsf{aux}_X(X) \quad \Leftarrow \quad \boxed{\mathsf{get}_1(c(X))}$$

$$\mathsf{aux}_Y(Y) \quad \Leftarrow \quad \boxed{q_2(c(Y))}$$

Therefore, we translate:

$$p(a(X,Y)) \quad \Leftarrow \quad \boxed{q_1(d(Z, \boxed{c(X)}))} \wedge \boxed{q_2(c(Y))} \wedge \boxed{q_3(Z)}$$

into:

$$\begin{aligned}
p(a(X,Y)) &\quad \Leftarrow \quad \mathsf{aux}_X(X) \wedge \mathsf{aux}_Y(Y) \\
\mathsf{aux}_X(X) &\quad \Leftarrow \quad \boxed{\mathsf{get}_1(c(X))} \\
\boxed{\mathsf{get}_1(X_1)} &\quad \Leftarrow \quad \boxed{q_1(d(Z, X_1)) \wedge q_3(Z)} \\
\mathsf{aux}_Y(Y) &\quad \Leftarrow \quad \boxed{q_2(c(Y))}
\end{aligned}$$

Note:

In expressiveness, $\mathcal{H}3$ clauses correspond to set constraints without intersections – enhanced with:

$$X \quad \supseteq \quad \pi_{a,i}(a(B_1, \ldots, B_{i-1}, \top, B_{i+1}, \ldots, B_k) \cap Y)$$

$//$ conditional projection

Note:

In expressiveness, $\mathcal{H}3$ clauses correspond to set constraints without intersections – enhanced with:

$$X \quad \supseteq \quad \pi_{a,i}(a(B_1, \ldots, B_{i-1}, \top, B_{i+1}, \ldots, B_k) \cap Y)$$

// conditional projection

Theorem:

The least model of an $\mathcal{H}3$-clause c can be computed in time:

$$\mathcal{O}(|c|^3)$$

Note:

In expressiveness, $\mathcal{H}3$ clauses correspond to set constraints without intersections – enhanced with:

$$X \quad \supseteq \quad \pi_{a,i}(a(B_1, \ldots, B_{i-1}, \top, B_{i+1}, \ldots, B_k) \cap Y)$$

$//$ conditional projection

Theorem:

The least model of an $\mathcal{H}3$-clause c can be computed in time:

$$\mathcal{O}(|c|^3)$$

... so what?

4. Instantiation

Expression Evaluation, revisited:

$$
\begin{array}{rcl}
\mathit{occurs}(E_1) & \Leftarrow & \mathit{occurs}(\mathsf{pair}(E_1, E_2)) \\
\mathit{occurs}(E_2) & \Leftarrow & \mathit{occurs}(\mathsf{pair}(E_1, E_2)) \\
\multicolumn{3}{l}{\mathit{hasValue}(\mathsf{pair}(E_1, E_2) \wedge \mathsf{pair}(V_1, V_2))} \\
 & \Leftarrow & \mathit{occurs}(\mathsf{pair}(E_1, E_2)) \wedge \\
 & & \mathit{hasValue}(E_1, V_1) \wedge \\
 & & \mathit{hasValue}(E_2, V_2)
\end{array}
$$

Observation:

occurs/1 holds only for ground sub-terms of the input :-)

Instantiation of all Possible Values:

$$
\begin{aligned}
occurs(e_1) &\Leftarrow occurs(\mathsf{pair}(e_1, e_2)) \\
occurs(e_2) &\Leftarrow occurs(\mathsf{pair}(e_1, e_2)) \\
\rlap{hasValue(\mathsf{pair}(e_1, e_2), \mathsf{pair}(V_1, V_2))} \\
&\Leftarrow occurs(\mathsf{pair}(e_1, e_2)) \wedge \\
&\qquad hasValue(e_1, V_1) \wedge \\
&\qquad hasValue(e_2, V_2)
\end{aligned}
$$

... now is $\mathcal{H}3$:-))

In General for Spi:

- Both *reach/1* and *occurs/1* hold for ground sub-terms of prog only :-)
- Why not instantiating the corresponding variables in all ways ?
- Removal of these variables results in:

$\Rightarrow$ a clause of size O(|prog|)

$\Rightarrow$ ... which is $\mathcal{H}3$.

... we conclude:

Theorem:

Control-flow analysis for Spi is cubic.

$\square$

Discussion:

- Mixing bounded components in predicates with unbounded ones, is convenient for practical applications :-)
- Manually performing instantiation is boring and error-prone ...
- Analyses are required for:

  (1) determining the bounded components of predicates;

  (2) determining the finitely many values for these components :-)

- The latter can be done with Datalog ;-)