Constraint Programming: Theory and Practice

Topics:

Rational Tree Solver (RT)

Prof. Dr. Slim Abdennadher and Prof. Dr. Carmen Gervet

13.12.2011

Rational Trees (RTs)

Possibly **infinite tree** which has a finite set of subtrees.

- No *occur-check* in early Prolog implementations for efficiency reasons, unification could go into an infinite loop
- In Prolog II, algorithm for properly handling the resulting infinite terms was introduced
- **Rational trees**: finite and infinite terms that are representable by finite trees

Example:

- Infinite tree $f(f(f(\ldots)))$ only contains itself
- finite representation as directed (possibly cyclic) graph
- representation as equality constraint, e.g., $X{=}f(X)$

Constraint System RT

- Based on constraint system E (finite trees)
- Constraint theory CET without acyclicity axiom (occur-check)

Signature

- Infinitely many function symbols.
- Constraint symbols.
  - Nullary symbols *true*, *false*
  - Binary symbol for syntactic equality $=$

Domain

Herbrand universe with Herbrand interpretation

Constraint System RT (2)

Constraint theory

Reflexivity	$\forall(\mathit{true} \rightarrow x{=}x)$
Symmetry	$\forall(x{=}y \rightarrow y{=}x)$
Transitivity	$\forall(x{=}y \wedge y{=}z \rightarrow x{=}z)$
Compatibility	$\forall(x_1{=}y_1 \wedge \ldots \wedge x_n{=}y_n \rightarrow f(x_1,\ldots,x_n){=}f(y_1,\ldots,y_n))$
Decomposition	$\forall(f(x_1,\ldots,x_n){=}f(y_1,\ldots,y_n) \rightarrow x_1{=}y_1 \wedge \ldots \wedge x_n{=}y_n)$
Contradiction (Clash)	$\forall(f(x_1,\ldots,x_n){=}g(y_1,\ldots,y_m) \rightarrow \mathit{false})$ if $f \neq g$ or $n \neq m$

Allowed atomic constraints

$$C ::= \mathit{true} \quad | \quad \mathit{false} \quad | \quad s{=}t$$

(s, t: terms over Σ)

RT – Solved Normal Form

Conjunction of allowed constraints is **solved (in solved normal form)**:

- *false* or
- $X_1 = t_1 \wedge \ldots \wedge X_n = t_n \ (n \geq 0)$
  - $X_1, \ldots, X_n$ pairwise distinct
  - X_i different to t_j if $i \leq j$

Examples:

- Not in solved normal form:
  $f(X, b) = f(a, Y)$, or $X = t \wedge X = s$, or $X = Y \wedge Y = X$
- In solved normal form: $X = Z \wedge Y = Z \wedge Z = t$
- Logically equivalent but syntactically different solved forms:
  $X = Y$ and $Y = X$
  $X = f(X)$ and $X = f(f(X))$

RT – Variable Elimination Constraint Solver

- `s@<t`: built-in total order on terms
  - `s`, `t` variables: given order
  - `s` variable, `t` function term
  - `s` is ordered before `t` function terms: size of `s` is less than size of `t` (*size*: number of occurrences of function symbols)
- `var(X)`, `nonvar(X)`
- `same_functor(T1,T2)`: tests if `T1` and `T2` have the same function symbol and the same arity
- `args2list(T1,L1)`: `L1` is the list of arguments of the term `T1`

RT – Variable Elimination Constraint Solver (2)

```
reflexivity   @ X eq X <=> var(X) | true.

orientation   @ T eq X <=> var(X),X@<T | X eq T.

decomposition @ T1 eq T2 <=> nonvar(T1),nonvar(T2) |
                   same_functor(T1,T2),
                   args2list(T1,L1),args2list(T2,L2),
                   same_args(L1,L2).

confrontation @ X eq T1, X eq T2 <=> var(X),X@<T1,T1@=<T2 |
                   X eq T1, T1 eq T2.

same_args([],[])              <=> true.
same_args([T1|L1],[T2|L2]) <=> T1 eq T2, same_args(L1,L2).
```

RT Example – Equating two Terms

`h(Y,f(a),g(X,a)) eq h(f(U),Y,g(h(Y),U)))`

$\mapsto_{\texttt{decomposition}}\mapsto^*$ `Y eq f(U), f(a) eq Y, g(X,a) eq g(h(Y),U)`

$\mapsto_{\texttt{orientation}}$ `Y eq f(U), Y eq f(a), g(X,a) eq g(h(Y),U)`

$\mapsto_{\texttt{decomposition}}\mapsto^*$ `Y eq f(U), Y eq f(a), X eq h(Y), a eq U`

$\mapsto_{\texttt{orientation}}$ `Y eq f(U), Y eq f(a), X eq h(Y), U eq a`

$\mapsto_{\texttt{confrontation}}$ `Y eq f(U), f(U) eq f(a), X eq h(Y), U eq a`

$\mapsto_{\texttt{decomposition}}\mapsto^*$ `Y eq f(U), U eq a, X eq h(Y), U eq a`

$\mapsto_{\texttt{confrontation}}$ `Y eq f(U), U eq a, X eq h(Y), a eq a`

$\mapsto_{\texttt{decomposition}}\mapsto^*$ `Y eq f(U), U eq a, X eq h(Y)`

RT Example – Equating two Terms (2)

$$
\begin{array}{ll}
 & \underline{\texttt{X eq f(X)}},\ \underline{\texttt{X eq f(f(X))}} \\
\longmapsto_{\texttt{confrontation}} & \texttt{X eq f(X)},\ \underline{\texttt{f(X) eq f(f(X))}} \\
\longmapsto_{\texttt{decomposition}}\longmapsto^{*} & \underline{\texttt{X eq f(X)}},\ \underline{\texttt{X eq f(X)}} \\
\longmapsto_{\texttt{confrontation}} & \texttt{X eq f(X)},\ \underline{\texttt{f(X) eq f(X)}} \\
\longmapsto_{\texttt{decomposition}}\longmapsto^{*} & \texttt{X eq f(X)},\ \underline{\texttt{X eq X}} \\
\longmapsto_{\texttt{reflexivity}} & \texttt{X eq f(X)}
\end{array}
$$