Fondements de la Programmation: Constraint Domains and Applications

Giovanni Coray - Vincenzo Pallotta

June 25, 2000

References:

- "Constraint Logic Programming: A survey" - J. Jaffar and M. Maher, The Journal of Logic Programming Vol. 19 & 20, pp. 503-582, May 1994.
- "Programming with constraints: An Introduction" - K. Mariot and P.J. Stuckey, MIT Press, 1998.
- "Theory and practice of Constraint Handling Rules" - Thom Früwirth, The Journal of Logic Programming Vol. 19 & 20, May 1994.

Constraints over $\Re$

We define the parameter $\mathcal{R} = \langle \Sigma, \mathcal{D}, \mathcal{L}, \mathcal{T} \rangle$ where:

- $\Sigma = \{\{0,1\}, \{+.*\}, \{=, <, \leq\}\}$
- $\mathcal{D} = \Re$
- $\mathcal{L}$ is the constraints set generated by primitive constraints (e.g. by combining constraints like: $E_1 = E_2$ or $E_1 < E_2$ or $E_1 \leq E_2$ using logical connectives and where $E_1 E_2$ are expression built using the symbols $0, 1, +, *$. For instance the constraint: $\exists x : (3x + y = 2)$ is an abbreviation of $\exists x : (x + x + x + y = 1 + 1)$.
- If $\mathcal{L}$ is restricted to $\Sigma \setminus \{*\}$ then $(\mathcal{D}, \mathcal{L})$ is the linear arithmetics over reals $\Re_{lin}$. The symplex algorithm can be considered as a constraint solver for linear disequations.
- If $\mathcal{D}$ is further restricted to rational numbers then we have linear arithmetics over rationals Q_{lin}.
- Adding the relation $\neq$we get $\Re^{\neq}_{lin}$ and $Q^{\neq}_{lin}$.
- If we further restrict $\Sigma = \{0, 1, =, +\}$ then we have linear equations over reals $\Re_{lineqn}$. A constraint solver for this domain is the Gauss elimination algorithm.

Event Calculus

Initial state:

- $initially(P)$: a property P holds at time 0

Scheduling:

- $happens(E, T)$: an event E happens at time T

Causal rules:

- $initiates(E, P, T)$: the event E initiates the property P at time T
- $terminates(E, P, T)$: the event E terminates the property P at time T

Inference engine:

- $clipped(T_1, P, T_2)$: the property P is clipped during the interval (T_1, T_2)
- $holds\_at(P, T)$: the property P holds at time T.

Event Calculus: the block world

Initial state:

$initially(on(b_3, table)).$ $initially(on(b_1, b_2)).$ $initially(clear(b_1)).$ $initially(clear(table)).$
$initially(on(b_2, b_3)).$ $initially(on(b_4, table)).$ $initially(clear(b_4)).$

Scheduling:

$happens(move(1, 4), 1.25).$
$happens(move(2, 1), 2.75).$

Causal rules:

$$\begin{aligned} initiates(move(X,Y), on(X,Y), T) \leftarrow\ & holds\_at(clear(X), T), \\ & holds\_at(clear(Y), T), \\ & X \neq Y, X \neq table. \end{aligned}$$

$$\begin{aligned} terminates(move(X,Y), on(X,Z), T) \leftarrow\ & holds\_at(clear(X), T), \\ & holds\_at(clear(Y), T), \\ & holds\_at(on(X,Z), T), \\ & X \neq Y, Y \neq Z. \end{aligned}$$

$$\begin{aligned} initiates(move(X,Y), clear(Z), T) \leftarrow\ & holds\_at(clear(X), T), \\ & holds\_at(clear(Y), T), \\ & holds\_at(on(X,Z), T), \\ & X \neq Y, Y \neq Z. \end{aligned}$$

$$\begin{aligned} terminates(move(X,Y), clear(Y), T) \leftarrow\ & holds\_at(clear(X), T), \\ & holds\_at(clear(Y), T), \\ & X \neq Y, X \neq table. \end{aligned}$$

Event Calculus: inference engine

Inference engine:

- $clipped(T_1, P, T_2)$: the property P is clipped during the interval (T_1, T_2)

$$\begin{aligned} clipped(T_1, P, T_2) \leftarrow T_1 < T, T < T_2 | \\ happens(E, T), \\ terminates(E, P, T). \end{aligned}$$

- $holds\_at(P, T)$: the property P holds at time T.

$$\begin{aligned} holds\_at(P, T) \leftarrow \\ initially(P), \\ not\, clipped(0, P, T). \end{aligned}$$

$$\begin{aligned} holds\_at(P, T) \leftarrow T_1 < T | \\ happens(E, T_1), \\ initiates(E, P, T_1), \\ not\, clipped(T_1, F, T). \end{aligned}$$

Event Calculus: running the query...

Query: $holds\_at(on(X, Y), 2)$. returns the following answer constraints:

$$\begin{array}{l} \{X = b_3, Y = table\} \\ \{X = b_2, Y = b_3\} \\ \{X = b_4, Y = table\} \\ \{X = b_1, Y = b_4\} \end{array}$$

Finite Terms and Rational Trees

$\Sigma_P = \{HU_P, \{\doteq\}\}$ is the set of the Herbrand universe of P plus terms equality.

$\mathcal{D}$ is the domain of finite trees with the Herbrand pre-interpretation:

- each node of the tree is labelled by a constant or a function symbol, the number of children of each node si the arity of the label function, and the children are ordered. Each function symbol in Σ is interpreted as a tree constructor where each $f \in \Sigma$ of arity n maps n trees to a tree whose root is labeled by f and whose subtrees are the arguments of the mapping.

$\mathcal{L}$ is the first order language generated by primitive constraints:

- equation between terms and them (e.g. $X \doteq g(Y)$, $\exists Z.X \doteq f(Z, Z) \wedge Y \doteq g(Z)$).

$\mathcal{FT} = (\mathcal{D}, \mathcal{L})$ is the Herbrand constraint domain, as used in PROLOG.

- Finite trees are computed as solution by unification with occur check.
- If we consider $\mathcal{D}$ as the set of rational trees (i.e. finite and infinite trees with finitely many subtrees), and the same interpetration of terms, then we have the $\mathcal{RT}$ constraint domain. PROLOG unification (without occur check) is complete w.r.t. this constraint domain.

Constraints over Feature Trees

$\Sigma = \{S, F, \{=\}\}$ where:

- S is the set of *sorts* (unary predicates)
- F is the set of *features* (binary predicates)

$\mathcal{D}$ is the set of *feature trees* that is finite and infinite trees where for each tree:

- each node is labelled by a sort
- each edge is labelled by a feature
- no node has two outbound edges with the same label.

$\mathcal{D}$ interpret:

- each sort s as the set of feature trees whose root is labeled by s, and
- each feature f as partial functions such that $f(t_1) = t_2$ iff t_2 is the sub-tree of t_1 that is reached by the edge labelled by f. If t_1 has no edge labelled by f then $f(t_1)$ is undefined.

The constraint domain of feature trees is denoted by $\mathcal{FEAT}$. Typical constrains are:

$wine(X) \wedge \exists Y.region(X, Y) \wedge valais(Y) \wedge \exists Y.color(X, Y) \wedge white(Y).$

which can be represented by a more readable sugared syntax as:

$$X : wine[region \Rightarrow valais, color \Rightarrow white]$$

Boolean constrains

- $\Sigma = \{\{0,1\}, \{\neg, \wedge, \vee, \oplus, \Rightarrow\}, \{=\}\}$
- $\mathcal{D} = \{true, false\}$ with the usual interpretation of logical connectives ($\oplus$is the esclusive or, etc.).
- $\mathcal{L}$ is the language generated by the primitive constraints, then $\mathcal{BOOL} = (\mathcal{D}, \mathcal{L})$ is the (two-valued) boolean constraint domain.
- If we consider $\Sigma' = \Sigma \cup \{a_1, \ldots, a_i, \ldots\}$ wher a_i are constant, we can extend the language and the domain to the one of free boolean algebra generated by $\{a_1, \ldots, a_i, \ldots\}$ obtaining $\mathcal{BOOL}_\infty = (\mathcal{D}', \mathcal{L}')$ as the boolean constraint domain over with infinitely many symbolic values.
- Example:

$$\begin{array}{lll} add(I_1, I_2, I_3, O_1, O_2) & \leftarrow & X_1 = I_1 \oplus I_1, \\ & & A_1 = I_1 \wedge I_2, \\ & & O_1 = X_1 \oplus I_3, \\ & & A_2 = I_3 \wedge X_1, \\ & & O_2 = A_1 \vee A_2. \end{array}$$

Finite Domains

- $\Sigma = \{\{0,1\}, \{+\}, \{\in [m,n]\}_{m \leq n}, =, \neq, \leq\}\}$
- $\mathcal{D} = Z$ is the domain of integer numbers with its usual interpretation of $+$.
- $\mathcal{L}$ is the set of firs order constraints generated by Σ with the restriction that each variable is constrained by an interval ($X \in [m,n]$).
- $\mathcal{FD}$ is the constraint finite domain. Example:

  $X \in [1,5] \wedge Y \in [0,7] \wedge X \neq 3 \wedge X + 2Y \leq 5 \wedge X + Y \leq 5$
- $\mathcal{FD}$ can be axiomatized using the *Presburger Arithmetic* (i.e. a linear fragment of Peano arithmetics) which is complete and decidable even if NP-hard.

A constraint theory for $\mathcal{RT}$

Reflexivity: $\forall X.(X \doteq X)$

Symmetry: $\forall X, Y.(X \doteq Y \Leftrightarrow Y \doteq X)$

Transitivity:
$\forall X, Y, Z.((X \doteq Y \wedge Y \doteq Z) \Rightarrow X \doteq Z)$

Substitution (Compatibility):

$$\forall((X_1 \doteq Y_1 \wedge \ldots \wedge X_n \doteq Y_n) \Rightarrow f(X_1, \ldots, X_n) \doteq f(Y_1, \ldots, Y_n))$$

Decomposition:

$$\forall(f(X_1, \ldots, X_n) \doteq f(Y_1, \ldots, Y_n) \Rightarrow (X_1 \doteq Y_1 \wedge \ldots \wedge X_n \doteq Y_n))$$

Clash:

$$\forall(f(X_1, \ldots, X_n) \doteq g(Y_1, \ldots, Y_m) \Rightarrow false)$$

if $f \neq g$ or $m \neq n$.

Note: No Acyclicity axiom, i.e. no occurs check