Fondements de la Programmation: Constraint Handling Rules

Giovanni Coray - Vincenzo Pallotta

June 25, 2000

References:

- "Constraint Handling Rules" - Thom Früwirth, In *Constraint Programming: basic and trends* (A. Podelski, ed.) LNCS 910, Springer Verlag, March 1995. `http://www.icparc.ic.ac.uk/eclipse/reports/chatillon.ps.Z`
- "ECLiPSe " `http://www.icparc.ic.ac.uk/eclipse/`
- "SICStus PROLOG" `http://www.sics.se/sicstus/`

A constraint theory for partial order relation

Consider the following logical theory:

Reflexivity: $\forall X . X \leq X$

Antisymmetry: $\forall X, Y . ((X \leq Y \wedge Y \leq X) \Leftrightarrow X = Y)$

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

Let's define the minum operator between two elements of a partial order:

$$min(X, Y, Z) \Leftrightarrow ((X \leq Y \wedge X = Z) \vee (Y \leq X \wedge Y = Z))$$

This theory is sound and complete but it doesn't say nothing on how to compute the relation.

User defined constraints

A LP meta interpreter for CLP:

$$\begin{aligned}
demo_{CLP}(C|empty, AC) &\leftarrow C = AC. \\
demo_{CLP}(C|(G_1, G_2).AC) &\leftarrow demo_{CLP}(C|G_1, C_1), \\
& demo_{CLP}(C_1|G_2, AC). \\
demo_{CLP}(C|G, AC) &\leftarrow clause_{CLP}(G, B, C'), \\
& simplify(C, C', C''), \\
& demo_{CLP}(C''|B, AC).
\end{aligned}$$

where AC is the answer constraint.
Clauses like:

$$H \leftarrow C|B$$

are represented by:

$$clause_{CLP}(H, B, C)$$

where C is a list of constraints $[C_1, ..., C_n]$.
The predicate $simplify(C_1, C_2, C_3)$ implements the constraint checker/solver.

Constraint Handling Rules

Example (partial order relation):

$$\begin{aligned}
X \leq Y &\Leftrightarrow X \doteq Y|true. \\
X \leq Y, Y \leq X &\Leftrightarrow true|X \doteq Y. \\
X \leq Y, Y \leq Z &\Rightarrow true|X \leq Z.
\end{aligned}$$

Computation (assume built-in equality $\doteq$):

$$\{A \leq B, C \leq A, B \leq C\}$$

$C \leq A, A \leq B$ propagates $C \leq B$
$C \leq B, B \leq C$ yields $B \doteq C$
$A \leq B, C \leq A$ yields $A \doteq B$ since $B \doteq C$

$$\{B \doteq C, A \doteq B\}$$

CHR syntax

A CHR-program is a finite set of CHR-rules of two possible form:

Propagation

$$E \Rightarrow C|G$$

Simplification

$$E \Leftrightarrow C|G$$

where

- E is a conjunction of CHR-constraints
- the guard C is a conjunction of predefined (or buil-in) constraints
- G is a goal.

CHR states and derivations

- A *state* is a triple $\langle F, E, D\rangle$ where:
  - F is the goal store (i.e. constraint to be solved)
  - E is the CHR-store (i.e. accumulated and simplified user constraints)
  - D is a predefined constraint store (i.e. accumulated and simplified built-in constraints)
- The empty CHR-store E is represented by $false$.
- An *initial state* is a state of the form $\langle F, true, true\rangle$.
- A successful final state is a state which cannot be further reduced and it has the form $\langle true, E, D\rangle$ with $D \neq false$.
- A failed final state is a state of the form $\langle F, E, false\rangle$.
- A *derivation* is a finite sequence of states starting from the initial state $\langle F, true, true\rangle$ obtained by the application of CHR-rules.

Logical meaning of CHR-programs

The declarative interpretation of a CHR program P is given by a conjunction of universally quantified logical formulas (one for each rule), $\mathcal{P}$, and a consistent built-in constraint theory CT which determines the meaning of built-in (and predefined) constraints appearing in the program.

- The constraint theory CT is expected to include an equality constraint and the basic constraint $true$ and $false$.
- Let $\overline{X}$ denote the sequence of global variable occurring in the head atoms $H_1, \dots, H_n$ of a CHR-rule. Let $\overline{Y}$ ($\overline{Z}$) are the other local variables occurring in the guard $G_1, \dots, G_m$ (the body $B_1, \dots, B_k$) of the CHR-rule.
- We assume also that no variables occur in both the guard and the body of a rule.

Simplification: a simplification rule $H_1, \dots, H_n \Leftrightarrow G_1, \dots, G_m | B_1, \dots, B_k$ is a logical equivalence if the guard is satisfied:

$$\forall \overline{X}(\exists \overline{Y}(G_1 \wedge \dots \wedge G_m) \Rightarrow ((H_1 \wedge \dots \wedge H_n) \Leftrightarrow \exists \overline{Z}(B_1 \wedge \dots \wedge B_k)).$$

Propagation: a propagation rule $H_1, \dots, H_n \Rightarrow G_1, \dots, G_m | B_1, \dots, B_k$ is a logical equivalence if the guard is satisfied:

$$\forall \overline{X}(\exists \overline{Y}(G_1 \wedge \dots \wedge G_m) \Rightarrow ((H_1 \wedge \dots \wedge H_n) \Rightarrow \exists \overline{Z}(B_1 \wedge \dots \wedge B_k)).$$

Example: The CHR-rule for reflexivity

$$X \leq Y \Leftrightarrow X \doteq Y | true$$

has the logical reading:

$$\forall X, Y (X = Y) \Rightarrow ((X \leq Y) \Leftrightarrow true).$$

Operational semantics

The aim of the computation is to incrementally reduce arbitrary states to states that contain no more goals.
Let P be a CHR-program defining some CHR constraints and CT be a constraint theory for the built-in constraints. The transition relation $\longmapsto$ for CHR is then defined as follows:

Solve

$$\langle C \wedge F, E, D\rangle \longmapsto \left\langle F, E, D'\right\rangle$$

if C is a built-in constraint and $CT \models (C \wedge D) \Leftrightarrow D'$.

Introduce

$$\langle H \wedge F, E, D\rangle \longmapsto \langle F, H \wedge E, D\rangle$$

if H is a CHR-constraint.

Simplify

$$\left\langle F, H' \wedge E, D\right\rangle \longmapsto \left\langle B \wedge F, E, (H = H') \wedge D\right\rangle$$

if $(H \Leftrightarrow G|B) \in P$ and $CT \models D \Rightarrow \exists \overline{X}(H = H' \wedge G)$
where $\overline{X}$ are the variables occuring in the mult-head H.

Propagate

$$\left\langle F, H' \wedge E, D\right\rangle \longmapsto \left\langle B \wedge F, H' \wedge E, (H = H') \wedge D\right\rangle$$

if $(H \Leftrightarrow G|B) \in P$ and $CT \models D \Rightarrow \exists \overline{X}(H = H' \wedge G)$
where $\overline{X}$ are the variables occuring in the mult-head H.

Note: $(H_1 \wedge \cdots \wedge H_n) = (H_1' \wedge \cdots \wedge H_n')$ means $H_1 = H_1' \wedge \cdots \wedge H_n = H_n'$.

Fair rule application

- A CHR rule is applicable to CHR constraints H' whenever these constraints match the head atoms H of a CHR-rule by means of syntactic equalities implied by the built-in constraint store D.
- It is required that rules are applied fairly (i.e. that every rule that is applicable, eventually will).
- Fairness is respected and trivial non-termination is avoided by applying a propagation rule at most once to the same constraints.
- Example revisited:

$$
\begin{array}{ll}
 & \langle A \leq B \wedge C \leq A \wedge B \leq C, true, true \rangle \\
\longmapsto^{3}_{Introduce} & \langle true, A \leq B \wedge C \leq A \wedge B \leq C, true \rangle \\
\longmapsto_{Propagate}\ \textbf{Trans.} & \langle C \leq B, A \leq B \wedge C \leq A \wedge B \leq C, true \rangle \\
\longmapsto_{Introduce} & \langle true, A \leq B \wedge C \leq A \wedge B \leq C \wedge C \leq B, true \rangle \\
\longmapsto_{Simplify}\ \textbf{Antis.} & \left\langle \begin{array}{l} X = Y, A \leq B \wedge C \leq A, \\ (X \leq Y \wedge Y \leq X) = (B \leq C \wedge C \leq B) \end{array} \right\rangle \\
\longmapsto_{Solve} & \langle true, A \leq B \wedge C \leq A, B = C \rangle \\
\longmapsto_{Simplify}\ \textbf{Antis.} & \left\langle \begin{array}{l} X = Y, true, B = C \wedge \\ (X \leq Y \wedge Y \leq X) = (A \leq B \wedge C \leq A) \end{array} \right\rangle \\
\longmapsto_{Solve} & \langle true, true, A = B \wedge B = C \rangle
\end{array}
$$

Note: that in the last simplification the rule can be applied since

$$CT \models (B = C) \Rightarrow \exists X, Y((X \leq Y \wedge Y \leq X) = (A \leq B \wedge C \leq A))$$

Soundness and completeness

Let P be a CHR program and G be a goal.

Lemma. *The transition for CHR preserve the logical meaning. If C is the logical meaning of a state appearing in a derivation of G, then*

$$\mathcal{P}, CT \models \forall(C \Leftrightarrow G)$$

where $\forall F$ denotes the universal closure of a formula F.

Proof. By structural induction over the derivation steps.

□

Theorem. *Soundness*

If G has a derivation with answer C then

$$\mathcal{P}, CT \models \forall(C \Leftrightarrow G)$$

Proof. Immediately from the above lemma.

□

Theorem. *Completeness*

Let C be a conjunction of constraints. If G has at least one finite derivation and $\mathcal{P}, CT \models \forall(C \Leftrightarrow G)$ then G has a computation with answer C' such that

$$\mathcal{P}, CT \models \forall(C \Leftrightarrow C')$$

Proof. Immediately from soundness theorem.

□

Corollary. *Negation as finite failure*

If G has a finitely failed derivation, then $\mathcal{P}, CT \models \neg\exists G$.

Confluent CHR programs

By the previous lemma we know that in a CHR program, the result of a computation from a given goal will always have the same meaning. However it is not guaranteed that the result is syntactically the same.

Example 1 consider the CHR-program P:

$$
\begin{array}{lcl}
flip(Coin) & \Leftrightarrow & true | Coin \doteq face1. \\
flip(Coin) & \Leftrightarrow & true | Coin \doteq face2.
\end{array}
$$

The source state $\langle true, flip(Coin), true \rangle$ yields the following two non *joinable* target states:

$\langle true, true, takes(Coin, face1) \rangle$ **vs** $\langle true, true, takes(Coin, face2) \rangle$

Example 2 consider the CHR-program P:

$$
\begin{array}{lcl}
min(X, Y, Z) & \Leftrightarrow & X \leq Y | X \doteq Z. \\
min(X, Y, Z) & \Leftrightarrow & Y \leq X | Y \doteq Z.
\end{array}
$$

The source state $\langle true, min(X, Y, Z), X \leq Y, Y \leq X \rangle$ yields to the following critical pair of target states:

$\langle X \doteq Z, true, X \leq Y, Y \leq X \rangle$ **vs** $\langle Y \doteq Z, true, X \leq Y, Y \leq X \rangle$

both possibly yielding to the common state

$$\langle true, true, X \doteq Y \wedge Y \doteq Z. \rangle$$

Integration between CLP and CHR

With the transition rules given so far we can implement our constraint theory only using built-in and previously defined constraints. We want to allow ordinary CLP clauses in our CHR programs to be used when no simplification and propagation is possible anymore.

Labelling In constraint logic program, constraint handling is interleaved by making choices. Labelling is controlled way to make choices, that is choosing values for the constrained variables by means of a labelling predicate called at the end of the program.

Built-in labeling The CHR run-time system provides built-in labeling for user-defined constraints. The programmers can define their own labeling predicates using clauses whose heads are user-defined constraints.

Label_with The *label_with* declaration restricts the use of the clauses for built-in labeling. One can have several *label_with* declarations for the same constraint.

CLP+CHR programs

Definition. *A CLP+CHR program is a finite set of*

- CLP clauses for user-defined predicates and CHR constraints
- CHR rules for CHR constraints
- labelling declarations for CHR constraints of the form:

$$label\_with\, H\; if\; G_1, \ldots G_n$$

where H is a CHR constraint.

Unfolds

$$\langle H' \wedge F, E, D\rangle \longmapsto \langle B \wedge F, E, (H = H') \wedge D\rangle$$

if $(H \leftarrow B) \in P$ and H is not a CHR constraint.

Propagate

$$\langle F, H' \wedge E, D\rangle \longmapsto \langle B \wedge F, H' \wedge E, (H = H') \wedge D\rangle$$

if $(H \leftarrow B) \in P$ and $(label\_with\, H''\; if\; G) \in P$ and $CT \models D \Rightarrow \exists\overline{X}(H' = H'' \wedge G)$ where $\overline{X}$ are the variables occuring in the mult-head H.