Constraint Handling Rules

Essentials
Thom Frühwirth

University of Ulm, Germany

Images are subject to copyright of the respective owners
Citations may be not recent and incomplete for space reasons

Syntax and Declarative Semantics

Declarative Semantics

Simplification rule: \[ H \leftrightarrow C \mid B \quad \forall \bar{x} \ (C \rightarrow (H \leftrightarrow \exists \bar{y} \ B)) \]

Propagation rule: \[ H \Rightarrow C \mid B \quad \forall \bar{x} \ (C \rightarrow (H \rightarrow \exists \bar{y} \ B)) \]

Constraint Theory for Built-Ins

- Head \( H \): non-empty conjunction of CHR constraints
- Guard \( C \): conjunction of built-in constraints
- Body \( B \): conjunction of CHR and built-in constraints (goal)

Soundness and Completeness based on logical equivalence of states in a computation.
Operational Semantics

Apply rules until exhaustion in any order (fixpoint computation). Initial goal (query) \( \rightarrow^* \) result (answer).

**Simplify**

If \((H \equiv C \mid B)\) rule with renamed fresh variables \( \bar{x} \)
and \(CT \models G_{\text{builtin}} \rightarrow \exists \bar{x}(H=H' \land C)\)
then \(H' \land G \leftrightarrow G \land H \equiv H' \land B\)

**Propagate**

If \((H \Rightarrow C \mid B)\) rule with renamed fresh variables \( \bar{x} \)
and \(CT \models G_{\text{builtin}} \rightarrow \exists \bar{x}(H=H' \land C)\)
then \(H' \land G \leftrightarrow H' \land G \land H \equiv H' \land B\)

Refined operational semantics [Duck+, ICLP 2004]: Similar to procedure calls, CHR constraints evaluated depth-first from left to right and rules applied top-down in program text order. **Active vs. Partner constraint.**

Example Partial Order Constraint

\[
\begin{align*}
X \leq X & \iff \text{true} \tag{reflexivity} \\
X \leq Y \land Y \leq X & \iff X = Y \tag{antisymmetry} \\
X \leq Y \land Y \leq Z & \Rightarrow X \leq Z \tag{transitivity}
\end{align*}
\]

\[
\begin{align*}
A \leq B \land B \leq C \land C \leq A & \tag{transitivity} \\
A \leq B \land B \leq C \land C \leq A \land A \leq C & \tag{transitivity} \\
A \leq B \land B \leq C \land A = C & \tag{antisymmetry} \\
A \leq B \land B \leq A \land A = C & \tag{antisymmetry} \\
A = B \land A = C & \tag{built-in solver}
\end{align*}
\]
Properties of CHR programs

Guaranteed properties

- Anytime approximation algorithm
- Online incremental algorithm
- Concurrent/Parallel execution

Analyzable properties

- Termination/Time Complexity (semi-automatic)
- Determinism/Confluence (decidable)
- Program Equivalence (decidable!)

Anytime Algorithm - Approximation

Computation can be interrupted and restarted at any time.
Intermediate results approximate final result.

\[
\frac{A \leq B \land B \leq C \land C \leq A}{A \leq B \land B \leq C \land C \leq A \land A \leq C} \quad \text{(transitivity)}
\]

\[
\frac{A \leq B \land B \leq C \land C \leq A \land A \leq C}{A = B \land A = C} \quad \text{(antisymmetry)}
\]
Online Algorithm - Incremental

The complete input is initially unknown.
The input data arrives incrementally during computation.
No recomputation from scratch necessary.

Monotonicity and Incrementality
If \( G \longrightarrow G' \)
then \( G \land C \longrightarrow G' \land C \)

\[
\begin{align*}
A \leq B \land B \leq C \land C \leq A & \quad \text{(transitivity)} \\
A \leq B \land B \leq C \land A \leq C \land C \leq A & \quad \text{(antisymmetry)} \\
A \leq B \land B \leq C \land A = C & \\
\ldots
\end{align*}
\]

Concurrency - Weak Parallelism

Rules can be applied in parallel to different parts of the problem.

If \( A \longrightarrow B \)
and \( C \longrightarrow D \)
then \( A \land C \longrightarrow B \land D \)
Concurrency - Strong Parallelism

**Interleaving semantics**: Parallel computation step can be simulated by a sequence of sequential computation steps.

Rules can be applied in parallel to **overlapping parts** of a goal, if overlap is not removed.

\[
\begin{align*}
\text{If} \quad & A \land E \quad \implies \quad B \land E \\
\text{and} \quad & C \land E \quad \implies \quad D \land E \\
\text{then} \quad & A \land C \land E \quad \implies \quad B \land D \land E
\end{align*}
\]

\[
\begin{align*}
A < B & \land \quad B < C \quad \land \quad C < A \\
\downarrow & \quad \downarrow \\
A < B \land A < C & \land \quad B < C \quad \land \quad C < A \land B < A \\
\downarrow & \\
\ldots \quad A = C \quad \ldots
\end{align*}
\]

**The CHR Machine**
Sublanguage of CHR.
Can be mapped to Turing machines and vice versa.
CHR is Turing-complete.
Can be mapped to RAM machines and vice versa.
Every algorithm can be implemented in CHR with best known time and space complexity.
[Sneyers, Schrijvers, Demoen, CHR’05]
CHR Program Analysis

Prove that...

Termination
Every computation starting from any goal ends. [LNAI 1865, 2000]

Complexity
Worst-case time complexity follows from structure of rules. [KR’02]

Consistency and Correctness
Logical reading of the rules is consistent and follows from a specification. [Constraints Journal 2000]

Decidable Confluence
The answer of a query is always the same, no matter which of the applicable rules are applied. [CP’96, CP’97, Constraints Journal 2000]

Completion
Non-confluent programs made confluent by adding rules. [CP’98]

Decidable Operational Equivalence
Two programs have the same results for any given query. [CP’99]

Minimal States

For each rule, there is a minimal, most general state to which it is applicable.

Rule: \[ H \iff C \mid B \quad \text{or} \quad H \Rightarrow C \mid B \]

Minimal State: \[ H \land C \]

Every other state to which the rule is applicable contains the minimal state (cf. Monotonicity/Incrementality).
Confluence

Given a goal, every computation leads to the same result no matter what rules are applied.
A decidable, sufficient and necessary condition for confluence of terminating CHR programs through joinability of critical pairs.

\[ X \leq X \iff \text{true} \quad \text{(reflexivity)} \]
\[ X \leq Y \land Y \leq X \iff X = Y \quad \text{(antisymmetry)} \]

Start from overlapping minimal states

Completion

Derive rules from a non-joinable critical pair for transition from one of the critical states into the other one.

\[ X \leq Y \land Y \leq X \iff X = Y \quad \text{(antisymmetry)} \]
\[ X \leq Y \land Y < X \iff \text{false} \quad \text{(inconsistency)} \]

\[ A \leq B \land B \leq A \land B < A \]
\[ \text{antisymmetry} \quad \text{inconsistency} \]
\[ A = B \land B < A \]
\[ B \leq A \land \text{false} \]
\[ A = B \land A < A \]
\[ \text{false} \]
\[ X < X \iff \text{false} \quad \text{(irreflexivity)} \]
Operational Equivalence

Given a goal and two programs, computations in both programs leads to the same result.
A decidable, sufficient and necessary condition for operational equivalence of terminating CHR programs through joinability of minimal states.

\[ P_1 \quad \text{min}(X, Y, Z) \iff X \leq Y \lor Z = X. \]
\[ \text{min}(X, Y, Z) \iff X > Y \lor Z = Y. \]

\[ P_2 \quad \text{min}(X, Y, Z) \iff X < Y \lor Z = X. \]
\[ \text{min}(X, Y, Z) \iff X \geq Y \lor Z = Y. \]

\[
\begin{align*}
\text{min}(X, Y, Z) &\land X \leq Y \\
\downarrow & \quad P_1 \\
Z = X \land X \leq Y
\end{align*}
\]

\[
\begin{align*}
\text{min}(X, Y, Z) &\land X \leq Y \\
\downarrow & \quad P_2 \\
Z = X \land X \leq Y
\end{align*}
\]
Finally...

Google “Constraint Handling Rules” for the CHR website

Transcribed as CHR, means
to speed, to propagate, to be famous