Concurrent Constraint Logic Programming

Constraint Handling Rules

Complete definition of P⁺⁺ ∪ CT

1. Completion P⁺⁺
   For each p/n in P add to P⁺⁺ the formula
   
   \[
   p(t₁) \leftarrow \text{Guard}_₁ \land \exists Y_₁(t₁ \equiv Y_₁ \land G₁) \lor \ldots \lor \exists Y_n(t_m \equiv X \land G_n)
   \]

2. CT including Clark’s Equality Theory (CET)
   
   \[
   \begin{align*}
   \text{Reflexivity (} Τ \rightarrow X \equiv X) \\
   \text{Symmetry (} X \equiv Y \rightarrow Y \equiv X) \\
   \text{Transitivity (} X \equiv Y \land Y \equiv Z \rightarrow X \equiv Z) \\
   \text{Compatibility (} X_₁ \equiv Y_₁ \land \ldots \land X_n \equiv Y_n \land \text{Guard}_m \rightarrow f(X₁, \ldots, X_n) \equiv f(Y₁, \ldots, Y_n)) \\
   \text{Decomposition (} f(X₁, \ldots, X_n) \equiv f(Y₁, \ldots, Y_n)) \\
   \text{Contradiction (Clash) (} f(X₁, \ldots, X_n) \equiv g(Y₁, \ldots, Y_m) \rightarrow ⊥) \text{ if } f \neq g \text{ or } n \neq m \\
   \text{Acyclicity (} X \equiv t \rightarrow ⊥) \text{ if } t \text{ is function term and } X \text{ appears in } t
   \end{align*}
   \]

CHR-Rule Logical Reading

Simplify
\[ E \equiv C \lor G \rightarrow \exists X(C \rightarrow (E \equiv ∃Y(G))) \]

Propagate
\[ E \equiv C \lor G \rightarrow \exists X(C \rightarrow (E \rightarrow ∃Y(G))) \]

Constraint Logic Programming

Atom: A, B ::= p(t₁, ⋯, tₙ), n ≥ 0
Constraint: C, D ::= e(t₁, ⋯, tₙ) | C ∧ D, n ≥ 0
Goal: G, H ::= ⊤ | ⊥ | A | C ∨ G ∨ H

Declarative Semantics

Soundness
If a goal G has successful derivation with answer constraint C, then P⁺⁺ ∪ CT |= ∃(C → G).

Completeness
If P⁺⁺ ∪ CT |= ∃(C → G) and C is satisfiable in CT, then there are successful derivations for G with answer constraints C₁, ⋯, Cₙ s.t. CT |= ∃(C → (C₁ ∨ ⋯ ∨ Cₙ)).

Soundness and Completeness of Failed Derivations
Let P be a CL program and G a goal. P⁺⁺ ∪ CT |= ∃G if and only if each fair derivation starting with (G, true) fails finitely.

Concurrent Constraint Logic Programming

Atom: A, B ::= p(t₁, ⋯, tₙ), n ≥ 0
Constraint: C, D ::= e(t₁, ⋯, tₙ) | C ∧ D, n ≥ 0
Goal: G, H ::= ⊤ | ⊥ | A | C ∨ G ∨ H

Operational Semantics

Unfold
If (B ← H) is a fresh variant of a clause in P and CT |= (B≜A ∧ C)
then (A ∧ G, C) → (H ∧ A, (B≜A) ∧ C)

Failure
If there is no clause (B ← H) in P
with CT |= (∃((B≜A) ∧ C)
then (A ∧ G, C) → (⊥, false)

Solve
If CT |= ∃(C ∧ D₁) ↔ D₂
then (C ∧ G, D₁) → (G, D₂)

Constraint Handling Rules

CHR Constraint: E, F ::= e(t₁, ⋯, tₙ) | E ∧ F, n ≥ 0
Built-in Constraint: C, D ::= e(t₁, ⋯, tₙ) | C ∧ D, n ≥ 0
Goal: G, H ::= ⊤ | ⊥ | A | C | G ∨ H

CHC Rule: R ::= E ⇒ C | G | E ⇒ C | G
CHR Program: P ::= R₁, ⋯, Rₙ