Operational Equivalence of CHR Programs And Constraints

Slim Abdennadher and Thom Frühwirth

Computer Science Department

University of Munich

Oettingenstr. 67, 80538 Munich, Germany

Motivation

- Correctness of program transformation
- The use of modules or libraries with similar functionality
- Combination of constraint solvers

Example: Are the two CHR rules defining `max`

```
max(X,Y,Z) ⇔  X<Y |  Z=Y.
max(X,Y,Z) ⇔  X≥Y |  Z=X.
```

operationally equivalent to these two rules?

```
max(X,Y,Z) ⇔  X≤Y |  Z=Y.
max(X,Y,Z) ⇔  X>Y |  Z=X.
```

CHR: Syntax and Declarative Semantics

Upper case letters stand for conjunctions of CHR (user-defined) or built-in (predefined) constraints.

Simplification rule: $H \Leftrightarrow C \mid B \qquad \forall \bar{x}\ (C \rightarrow (H \leftrightarrow \exists \bar{y}\ B))$

Propagation rule: $H \Rightarrow C \mid B \qquad \forall \bar{x}\ (C \rightarrow (H \rightarrow \exists \bar{y}\ B))$

($\bar{x}$: variables occurring in H or C; $\bar{y}$: variables occurring only in B)

Declarative semantics of a CHR program:

- declarative reading of the rules and
- constraint theory CT for the built-in constraints.

CHR: Operational Semantics

Solve

If $CT \models \forall^* \; (G \leftrightarrow G')$

and G' is “simpler” than G

then $\dfrac{G}{G'}$

Simplify

If $(H \Leftrightarrow C \mid B)$ is a fresh variant of a rule with variables $\bar{x}$

and G_{bi} are the built-in constraints in G

and $CT \models G_{bi} \rightarrow \exists \bar{x}(H{=}H' \wedge C)$

then $\dfrac{H' \wedge G}{H{=}H' \wedge B \wedge G}$

Propagate

If $(H \Leftrightarrow C \mid B)$ is a fresh variant of a rule with variables $\bar{x}$

and G_{bi} are the built-in constraints in G

and $CT \models G_{bi} \rightarrow \exists \bar{x}(H{=}H' \wedge C)$

then $\dfrac{H' \wedge G}{H{=}H' \wedge B \wedge H' \wedge G}$

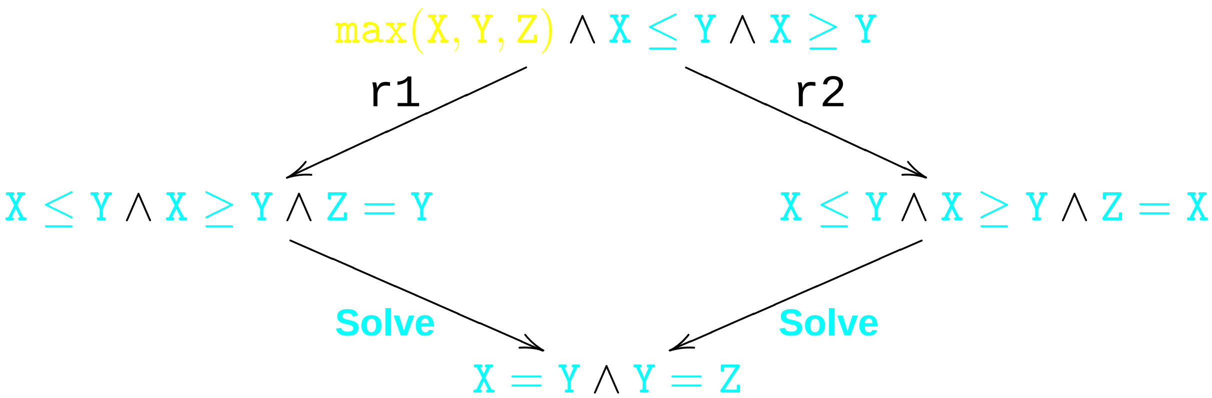
Confluence

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 (Abdennadher, CP97).

Example

```
r1 @  max(X,Y,Z) ⇔  X≤Y |  Z=Y.
r2 @  max(X,Y,Z) ⇔  X≥Y |  Z=X.
```



Compatibility of Programs

Definition: Let P_1 and P_2 be two confluent and terminating CHR programs and let the union of the two programs, $P_1 \cup P_2$, be terminating. P_1 and P_2 are *compatible* if $P_1 \cup P_2$ is confluent.

Example

$P1$:
```
max(X,Y,Z) ⇔ X<Y | Z=Y.
max(X,Y,Z) ⇔ X≥Y | Z=X.
```

$P2$:
```
max(X,Y,Z) ⇔ X≤Y | Z=Y.
max(X,Y,Z) ⇔ X>Y | Z=X.
```

Critical ancestor states from one rule in P_1 and one rule in P_2:

- max(X,Y,Z) $\wedge$ X<Y $\wedge$ X≤Y
- max(X,Y,Z) $\wedge$ X≥Y $\wedge$ X≤Y
- max(X,Y,Z) $\wedge$ X≥Y $\wedge$ X>Y

Compatibility vs. Operational Equivalence

Example

$P1$: `max(X,Y,Z) ⇔ X<Y | Z=Y.`
`max(X,Y,Z) ⇔ X≥Y | Z=X.`

$P2$: `max(X,Y,Z) ⇔ X≤Y | Z=Y.`
`max(X,Y,Z) ⇔ X>Y | Z=X.`

$P1$ and $P2$ are not operationally equivalent:

$$\mathtt{max}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \mathtt{X} \geq \mathtt{Y} \xrightarrow{P1} \mathtt{Z} = \mathtt{X} \wedge \mathtt{X} \geq \mathtt{Y}$$

$$\mathtt{max}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \mathtt{X} \geq \mathtt{Y} \xrightarrow{P2}$$

Operational Equivalence of Programs

Let P_1 and P_2 be CHR programs.

A state S is P_1, P_2*-joinable*, iff there are two computations $S \mapsto^*_{P_1} T$ and $S \mapsto^*_{P_2} T$, where T is a final state.

P_1 and P_2 are *operationally equivalent* iff all states are P_1, P_2-joinable.

Decidable, Sufficient and Necessary Condition

terminating and confluent

Theorem

critical

The *set of critical states of* P_1 *and* P_2:

$\{H \wedge C \mid (H \odot C \mid B) \in P_1 \cup P_2,$ where $\odot \in \{\Leftrightarrow, \Rightarrow\}\}$

Motivation: Equivalence of Constraints

P_1:

```
p(a) ⇔ s.
p(b) ⇔ r.
s ∧ q ⇔ true.
```

P_2:

```
p(a) ⇔ s.
p(b) ⇔ r.
```

p depends on s and r.

P_1 and P_2 are not operationally equivalent but operationally p-equivalent.

Operational Equivalence of Constraints

A c-*state* is a state where all CHR constraints have the same CHR symbol c. Let c defined in two CHR programs P_1 and P_2.

P_1 and P_2 are operationally c-equivalent if all c-states are P_1, P_2-joinable.

Sufficient Condition

terminating and confluent

Theorem

critical

The set of c-*critical states*:

$\{H \wedge C \mid (H \odot C \mid B) \in P_1 \cup P_2,$ where $\odot \in \{ \Leftrightarrow , \Rightarrow \}$ and H contains only c-dependent CHR symbols$\}$

Example

P_1:

```
p(a) ⇔ s.
p(b) ⇔ r.
s ∧ q ⇔ true.
```

P_2:

```
p(a) ⇔ s.
p(b) ⇔ r.
```

p depends on s and r.

The set of p-*critical states*: $\{\mathtt{p(a)},\ \mathtt{p(b)}\}$

Relationships

- Operational equivalence $\Longrightarrow$ Compatibility
- Compatibility $\not\Longrightarrow$ Operational equivalence (e.g. `max`)
- Operational equivalence of two CHR programs $\Longrightarrow$ operational c-equivalence of all common constraints c
- Operational c-equivalence of all common constraints of two CHR programs $\not\Longrightarrow$ Operational equivalence

Counterexample

P_1:

p $\Leftrightarrow$ s.
s $\wedge$ q $\Leftrightarrow$ true.

P_2:

p $\Leftrightarrow$ s.
s $\wedge$ q $\Leftrightarrow$ false.

Common CHR symbols p, s and q.

- s and p are the p-dependent CHR constraint symbols.
- s is the only s-dependent symbol.
- q is the only q-dependent symbol.

p is the only p-critical state. It is P_1, P_2-joinable.

But P_1 and P_2 are not operationally equivalent:

- s $\wedge$ q $\mapsto_{P_1}$ true
- s $\wedge$ q $\mapsto_{P_2}$ false

Conclusions

Given terminating and confluent CHR programs.

- A decidable, sufficient and necessary syntactic condition for operational equivalence of CHR programs
- A sufficient syntactic condition for operational equivalence of CHR constraints

Future Work

- Relationship between operational equivalence and logical equivalence
- Combination of solvers by program transformation using confluence, completion and operational equivalence

Example Operational Equivalence

$\texttt{sum([],Sum)} \Leftrightarrow \texttt{Sum=0.}$

$\texttt{sum([X|Xs],Sum)} \Leftrightarrow \texttt{sum(Xs,Sum1)} \wedge \texttt{Sum = Sum1 + X.}$

versus

$\texttt{sum([],Sum)} \Leftrightarrow \texttt{Sum = 0.}$

$\texttt{sum([X|Xs],Sum)} \Leftrightarrow \texttt{sum1(X,Xs,Sum).}$

$\texttt{sum1(X,[],Sum)} \Leftrightarrow \texttt{Sum = X.}$

$\texttt{sum1(X,Xs,Sum)} \Leftrightarrow \texttt{sum(Xs,Sum1)} \wedge \texttt{Sum = Sum1 + X.}$

sum([],Sum) and sum([X|Xs],Sum) are the sum-critical states. They are joinable.

Example for sufficient, but not necessary condition

```
p(X) ⇔  X>0 | q(X).
q(X) ⇔  X<0 | true.
```

versus

```
p(X) ⇔  X>0 | q(X).
q(X) ⇔  X<0 | false.
```

P_1 and P_2 are operationally p-equivalent, but the p-critical state $\mathtt{q}(\mathtt{X}) \wedge \mathtt{X} < 0$ is not P_1, P_2-joinable.

Example

```
max(X,Y,Z) ⇔ X<Y | Z=Y.
max(X,Y,Z) ⇔ X≥Y | Z=X.
range(X,Min,Max) ⇔ max(X,Min,X) ∧ max(X,Max,Max).
```

versus

```
range(X,Min,Max) ⇔ Max<Min | false.
range(X,Min,Max) ⇔ Min≤Max | Min≤X ∧ X≤Max.
```

P_1 and P_2 are not operationally `range`-equivalent, e.g. `range(5,6,Max)`.