Equivalence in CHR
Tools for Proofs

Frank Raiser | August 2010 | CHR Summer School, Belgium

Table of Contents

Equivalence of CHR States – Motivation

Important question:
Given two states, are they equivalent?

Equivalence of CHR States – Motivation

Important question:
Given two states, are they equivalent?

Why is this question important?

- CHR is non-deterministic: when applying different rules to a state, we would like to know if resulting states are equivalent $\leadsto$ confluence
- Input same state into different programs, we would like to check if the resulting states are equivalent
  - $\leadsto$ Program equivalence
  - Common in proofs involving source-to-source transformations
- ...

State Equivalence Examples

Definition (State)

A *state* is a tuple of the form $\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$ with $\mathbb{G}$ a multiset of CHR constraints, $\mathbb{B}$ a conjunction of built-ins, and $\mathbb{V}$ the set of global variables.

State Equivalence Examples

Definition (State)

A *state* is a tuple of the form $\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$ with $\mathbb{G}$ a multiset of CHR constraints, $\mathbb{B}$ a conjunction of built-ins, and $\mathbb{V}$ the set of global variables.

Example

$$
\begin{array}{lcl}
\langle c(X); \top; \{X\} \rangle & \equiv^? & \langle c(X); \top; \{X\} \rangle \\
\langle c(X); \top; \{X\} \rangle & \equiv^? & \langle c(Y); \top; \{Y\} \rangle \\
\langle c(X); \top; \emptyset \rangle & \equiv^? & \langle c(Y); \top; \emptyset \rangle \\
\langle c(X); X = 0; \{X\} \rangle & \equiv^? & \langle c(0); X = 0; \{X\} \rangle \\
\langle \emptyset; X \geq 0 \wedge X \leq 0; \{X\} \rangle & \equiv^? & \langle \emptyset; X = 0; \{X\} \rangle \\
\langle \emptyset; X = 1 \wedge X = 2; \{X\} \rangle & \equiv^? & \langle \emptyset; Y = 1 \wedge Y = 2; \{Y\} \rangle
\end{array}
$$

State Equivalence Examples

Definition (State)

A *state* is a tuple of the form $\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$ with $\mathbb{G}$ a multiset of CHR constraints, $\mathbb{B}$ a conjunction of built-ins, and $\mathbb{V}$ the set of global variables.

Example

$$
\begin{array}{lcl}
\langle c(X); \top; \{X\} \rangle & \equiv & \langle c(X); \top; \{X\} \rangle \\
\langle c(X); \top; \{X\} \rangle & \equiv^? & \langle c(Y); \top; \{Y\} \rangle \\
\langle c(X); \top; \emptyset \rangle & \equiv^? & \langle c(Y); \top; \emptyset \rangle \\
\langle c(X); X = 0; \{X\} \rangle & \equiv^? & \langle c(0); X = 0; \{X\} \rangle \\
\langle \emptyset; X \geq 0 \land X \leq 0; \{X\} \rangle & \equiv^? & \langle \emptyset; X = 0; \{X\} \rangle \\
\langle \emptyset; X = 1 \land X = 2; \{X\} \rangle & \equiv^? & \langle \emptyset; Y = 1 \land Y = 2; \{Y\} \rangle
\end{array}
$$

State Equivalence Examples

Definition (State)

A *state* is a tuple of the form $\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$ with $\mathbb{G}$ a multiset of CHR constraints, $\mathbb{B}$ a conjunction of built-ins, and $\mathbb{V}$ the set of global variables.

Example

$$
\begin{array}{lcl}
\langle c(X); \top; \{X\} \rangle & \equiv & \langle c(X); \top; \{X\} \rangle \\
\langle c(X); \top; \{X\} \rangle & \not\equiv & \langle c(Y); \top; \{Y\} \rangle \\
\langle c(X); \top; \emptyset \rangle & \equiv^? & \langle c(Y); \top; \emptyset \rangle \\
\langle c(X); X = 0; \{X\} \rangle & \equiv^? & \langle c(0); X = 0; \{X\} \rangle \\
\langle \emptyset; X \geq 0 \wedge X \leq 0; \{X\} \rangle & \equiv^? & \langle \emptyset; X = 0; \{X\} \rangle \\
\langle \emptyset; X = 1 \wedge X = 2; \{X\} \rangle & \equiv^? & \langle \emptyset; Y = 1 \wedge Y = 2; \{Y\} \rangle
\end{array}
$$

State Equivalence Examples

Definition (State)

A *state* is a tuple of the form $\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$ with $\mathbb{G}$ a multiset of CHR constraints, $\mathbb{B}$ a conjunction of built-ins, and $\mathbb{V}$ the set of global variables.

Example

$$
\begin{array}{lcl}
\langle c(X); \top; \{X\} \rangle & \equiv & \langle c(X); \top; \{X\} \rangle \\
\langle c(X); \top; \{X\} \rangle & \not\equiv & \langle c(Y); \top; \{Y\} \rangle \\
\langle c(X); \top; \emptyset \rangle & \equiv & \langle c(Y); \top; \emptyset \rangle \\
\langle c(X); X = 0; \{X\} \rangle & \equiv^? & \langle c(0); X = 0; \{X\} \rangle \\
\langle \emptyset; X \geq 0 \wedge X \leq 0; \{X\} \rangle & \equiv^? & \langle \emptyset; X = 0; \{X\} \rangle \\
\langle \emptyset; X = 1 \wedge X = 2; \{X\} \rangle & \equiv^? & \langle \emptyset; Y = 1 \wedge Y = 2; \{Y\} \rangle
\end{array}
$$

State Equivalence Examples

Definition (State)

A *state* is a tuple of the form $\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$ with $\mathbb{G}$ a multiset of CHR constraints, $\mathbb{B}$ a conjunction of built-ins, and $\mathbb{V}$ the set of global variables.

Example

$$
\begin{array}{rcl}
\langle c(X); \top; \{X\} \rangle & \equiv & \langle c(X); \top; \{X\} \rangle \\
\langle c(X); \top; \{X\} \rangle & \not\equiv & \langle c(Y); \top; \{Y\} \rangle \\
\langle c(X); \top; \emptyset \rangle & \equiv & \langle c(Y); \top; \emptyset \rangle \\
\langle c(X); X = 0; \{X\} \rangle & \equiv & \langle c(0); X = 0; \{X\} \rangle \\
\langle \emptyset; X \geq 0 \wedge X \leq 0; \{X\} \rangle & \equiv^? & \langle \emptyset; X = 0; \{X\} \rangle \\
\langle \emptyset; X = 1 \wedge X = 2; \{X\} \rangle & \equiv^? & \langle \emptyset; Y = 1 \wedge Y = 2; \{Y\} \rangle
\end{array}
$$

State Equivalence Examples

Definition (State)

A *state* is a tuple of the form $\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$ with $\mathbb{G}$ a multiset of CHR constraints, $\mathbb{B}$ a conjunction of built-ins, and $\mathbb{V}$ the set of global variables.

Example

$$
\begin{array}{lcl}
\langle c(X); \top; \{X\} \rangle & \equiv & \langle c(X); \top; \{X\} \rangle \\
\langle c(X); \top; \{X\} \rangle & \not\equiv & \langle c(Y); \top; \{Y\} \rangle \\
\langle c(X); \top; \emptyset \rangle & \equiv & \langle c(Y); \top; \emptyset \rangle \\
\langle c(X); X = 0; \{X\} \rangle & \equiv & \langle c(0); X = 0; \{X\} \rangle \\
\langle \emptyset; X \geq 0 \wedge X \leq 0; \{X\} \rangle & \equiv & \langle \emptyset; X = 0; \{X\} \rangle \\
\langle \emptyset; X = 1 \wedge X = 2; \{X\} \rangle & \equiv^{?} & \langle \emptyset; Y = 1 \wedge Y = 2; \{Y\} \rangle
\end{array}
$$

State Equivalence Examples

Definition (State)

A *state* is a tuple of the form $\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$ with $\mathbb{G}$ a multiset of CHR constraints, $\mathbb{B}$ a conjunction of built-ins, and $\mathbb{V}$ the set of global variables.

Example

$$
\begin{array}{rcl}
\langle c(X); \top; \{X\} \rangle & \equiv & \langle c(X); \top; \{X\} \rangle \\
\langle c(X); \top; \{X\} \rangle & \not\equiv & \langle c(Y); \top; \{Y\} \rangle \\
\langle c(X); \top; \emptyset \rangle & \equiv & \langle c(Y); \top; \emptyset \rangle \\
\langle c(X); X = 0; \{X\} \rangle & \equiv & \langle c(0); X = 0; \{X\} \rangle \\
\langle \emptyset; X \geq 0 \wedge X \leq 0; \{X\} \rangle & \equiv & \langle \emptyset; X = 0; \{X\} \rangle \\
\langle \emptyset; X = 1 \wedge X = 2; \{X\} \rangle & \equiv & \langle \emptyset; Y = 1 \wedge Y = 2; \{Y\} \rangle
\end{array}
$$

An Axiomatic Definition

or: what does it mean to be the “same”?

Definition (State Equivalence)

Equivalence between CHR states is the smallest equivalence relation $\equiv$ over CHR states satisfying:

1. *(Substitution)* $\langle \mathbb{G}; X \doteq t \land \mathbb{B}; \mathbb{V} \rangle \equiv \langle \mathbb{G}[X/t]; X \doteq t \land \mathbb{B}; \mathbb{V} \rangle$
2. *(Built-ins Equivalence)* If $\mathcal{CT} \models \exists \bar{s}.\mathbb{B} \leftrightarrow \exists \bar{s}'.\mathbb{B}'$ where $\bar{s}, \bar{s}'$ are the strictly local variables of $\mathbb{B}, \mathbb{B}'$, respectively, then $\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle \equiv \langle \mathbb{G}; \mathbb{B}'; \mathbb{V} \rangle$
3. *(Non-Occurring Globals)* If X is a variable that does not occur in $\mathbb{G}$ or $\mathbb{B}$ then $\langle \mathbb{G}; \mathbb{B}; \{X\} \cup \mathbb{V} \rangle \equiv \langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$
4. *(Failed States)* $\langle \mathbb{G}; \bot; \mathbb{V} \rangle \equiv \langle \mathbb{G}'; \bot; \mathbb{V} \rangle$

An Axiomatic Definition – Example

Example (Equivalence Proof)

$$\langle c(1), d(X); X = 2; \{X\}\rangle \equiv \langle c(Y), d(2); Y = 1 \wedge X = 2; \{X\}\rangle$$

An Axiomatic Definition – Example

Example (Equivalence Proof)

$$\langle c(1), d(X); X = 2; \{X\}\rangle \equiv \langle c(Y), d(2); Y = 1 \wedge X = 2; \{X\}\rangle$$

$$\begin{array}{ll} & \langle c(1), d(X); X = 2; \{X\}\rangle \\ \equiv^{\mathcal{CT}} & \langle c(1), d(X); Y = 1 \wedge X = 2; \{X\}\rangle \\ \equiv^{\text{Sub}} & \langle c(Y), d(X); Y = 1 \wedge X = 2; \{X\}\rangle \\ \equiv^{\text{Sub}} & \langle c(Y), d(2); Y = 1 \wedge X = 2; \{X\}\rangle \end{array}$$

Decision Criterion

or: how to tell if two states differ?

Theorem (Criterion for $\equiv$)

Let $\sigma = \langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle, \sigma' = \langle \mathbb{G}'; \mathbb{B}'; \mathbb{V} \rangle$ be CHR states with local variables $\bar{y}, \bar{y}'$ that have been renamed apart.

$$\sigma \equiv \sigma'$$

if and only if

$$\mathcal{CT} \models \begin{array}{c} \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{G} = \mathbb{G}') \wedge \mathbb{B}')) \\ \wedge \\ \forall(\mathbb{B}' \rightarrow \exists \bar{y}.((\mathbb{G} = \mathbb{G}') \wedge \mathbb{B})) \end{array}$$

- Simplifies negative proofs and allows automatic proof

Decision Criterion – Example

Example (Non-Equivalence Proof)

$$\langle c(X); X = 1; \{X\} \rangle \not\equiv \langle c(2); \top; \{X\} \rangle$$

Decision Criterion – Example

Example (Non-Equivalence Proof)

$$\langle c(X); X = 1; \{X\}\rangle \not\equiv \langle c(2); \top; \{X\}\rangle$$

- No local variables
- $\forall X.(X = 1 \rightarrow ((c(X) = c(2)) \wedge \top)$
- Simplified: $\forall X.X = 1 \rightarrow X = 2$
- Clearly: $\mathcal{CT} \not\models \forall X.X = 1 \rightarrow X = 2$

Summary: State Equivalence

Take Home Messages

- Axiomatic Definition of State Equivalence
- Decidable Criterion available
- Implementation available for automation

Operational Semantics – Motivation

Within a proof one may have to show that a rule application leads from one state to another. This should be quick and easy, right?

Operational Semantics – Motivation

Within a proof one may have to show that a rule application leads from one state to another. This should be quick and easy, right?

Example (Derivation Proof)

$$\mathrm{gcd}(N) \backslash \mathrm{gcd}(M) \Leftrightarrow M \geq N \wedge N > 0 \mid \mathrm{gcd}(L), L = M\%N$$

Given the above rule, prove that it allows rewriting gcd(6) and gcd(3) into gcd(3) and gcd(0).

Operational Semantics – Motivation

A formal proof is complicated and lengthy

Using the theoretical operational semantics ω_t:

$$
\begin{array}{ll}
 & \langle \mathrm{gcd}(6), \mathrm{gcd}(3); \emptyset; \top; \emptyset\rangle_0^{\emptyset} \\
\rightarrow_{\text{Intro}} & \langle \mathrm{gcd}(3); \mathrm{gcd}(6)\#0; \top; \emptyset\rangle_1^{\emptyset} \\
\rightarrow_{\text{Intro}} & \langle \emptyset; \mathrm{gcd}(6)\#0, \mathrm{gcd}(3)\#1; \top; \emptyset\rangle_2^{\emptyset} \\
\rightarrow_{\text{gcd}} & \langle \mathrm{gcd}(L), L = M\%N; \mathrm{gcd}(3)\#1; \mathrm{gcd}(6) = \mathrm{gcd}(M) \wedge \mathrm{gcd}(3) = \mathrm{gcd}(N) \wedge M \geq N \wedge N > 0; \emptyset\rangle_2^{\emptyset} \\
\rightarrow_{\text{Intro}} & \langle L = M\%N; \mathrm{gcd}(3)\#1, \mathrm{gcd}(L)\#2; \mathrm{gcd}(6) = \mathrm{gcd}(M) \wedge \mathrm{gcd}(3) = \mathrm{gcd}(N) \wedge M \geq N \wedge N > 0; \emptyset\rangle_3^{\emptyset} \\
\rightarrow_{\text{Solve}} & \langle \emptyset; \mathrm{gcd}(3)\#1, \mathrm{gcd}(L)\#2; L = 0; \emptyset\rangle_3^{\emptyset}
\end{array}
$$

this includes proving that:

$$
\begin{array}{l}
\mathcal{CT} \models \exists N, M.(\mathrm{gcd}(6) = \mathrm{gcd}(M) \wedge \mathrm{gcd}(3) = \mathrm{gcd}(N) \wedge M \geq N \wedge N > 0) \\
\mathcal{CT} \models \forall((\mathrm{gcd}(6) = \mathrm{gcd}(M) \wedge \mathrm{gcd}(3) = \mathrm{gcd}(N) \wedge M \geq N \wedge N > 0 \wedge L = M\#N) \leftrightarrow L = 0)
\end{array}
$$

Equivalence-based Operational Semantics

or: how to make things simple

Definition (Equivalence-based Operational Semantics)

$$\frac{r \ @ \ H_1 \backslash H_2 \Leftrightarrow G \mid B_c, B_b}{\langle H_1 \uplus H_2 \uplus \mathbb{G}; G \wedge \mathbb{B}; \mathbb{V} \rangle \rightarrowtail^r \langle H_1 \uplus B_c \uplus \mathbb{G}; G \wedge B_b \wedge \mathbb{B}; \mathbb{V} \rangle}$$

$$\frac{\sigma' \equiv \sigma \qquad \sigma \rightarrowtail^r \tau \qquad \tau \equiv \tau'}{\sigma' \rightarrowtail^r \tau'}$$

- Supports simplification, propagation, and simpagation rules (via $H_1 = \emptyset$ and $H_2 = \emptyset$)

Equivalence-based Operational Semantics

Advantages

- Every inference rule corresponds to a CHR rule application
- No additional conditions need to be proven
- Equivalent states are exchangeable anytime during derivation
  - Built-in store can be simplified anytime
  - In proofs we are free to select the most suitable state from all equivalent states for each derivation step
- Compatible with abstract operational semantics of CHR

Derivation Proof

Example (gcd Derivation Revisited)

$$\mathrm{gcd}(N) \setminus \mathrm{gcd}(M) \Leftrightarrow M \geq N \wedge N > 0 \mid \mathrm{gcd}(L), L = M\%N$$

$$\begin{array}{ll} & \langle \mathrm{gcd}(6), \mathrm{gcd}(3); \top; \emptyset \rangle \\ \equiv & \langle \mathrm{gcd}(M), \mathrm{gcd}(N); M \geq N \wedge N > 0 \wedge M = 6 \wedge N = 3; \emptyset \rangle \\ \rightarrowtail & \langle \mathrm{gcd}(L), \mathrm{gcd}(N); M \geq N \wedge N > 0 \wedge M = 6 \wedge N = 3 \wedge L = M\%N; \emptyset \rangle \\ \equiv & \langle \mathrm{gcd}(0), \mathrm{gcd}(3); \top; \emptyset \rangle \end{array}$$

More Abstract Formulation

or: how one rule captures the essence of CHR

Operational Semantics based on Equivalence Classes

$$\frac{r \; @ \; H_1 \backslash H_2 \Leftrightarrow G \mid B_c, B_b}{[\langle H_1 \uplus H_2 \uplus \mathbb{G}; G \wedge \mathbb{B}; \mathbb{V}\rangle] \rightarrowtail^r [\langle H_1 \uplus B_c \uplus \mathbb{G}; G \wedge B_b \wedge \mathbb{B}; \mathbb{V}\rangle]}$$

Operational Semantics based on Equivalence Classes

Advantages

- In program analysis, we have no more explicit state equivalence test
  - Instead, check that results are exactly the same (equivalence class)
- In a proof, if the current state is applicable to $r \; @ \; H_1 \backslash H_2 \Leftrightarrow G \mid B_c, B_b$, you know the state is

$$[\langle H_1 \uplus H_2 \uplus \mathbb{G}; G \wedge \mathbb{B}; \mathbb{V} \rangle]$$

  for some $\mathbb{G}, \mathbb{B}$, and $\mathbb{V}$.
- Equivalent to the less abstract formulation (= all advantages from before)

Summary: Equivalence-based Operational Semantics

Take Home Messages

- Every inference rule corresponds to a CHR rule application
- You can “w.l.o.g.” consider the most suitable state representation *at any point*

Merging and Splitting – Motivation

- Monotonicity is a big strength of CHR
  - Given any derivation $\sigma \rightarrowtail^* \tau$, the same rules are applicable if you "add" additional constraints to σ.
  - The added constraints then occur unchanged in the resulting state.
- Can we formalize this?
- If so, we can "subtract" (by duality) unnecessary constraints to make states simpler

Merge Operator

or: how to extend a state

Definition (Merge Operator $\diamond$)

Let $\sigma_1 = \langle \mathbb{G}_1; \mathbb{B}_1; \mathbb{V}_1 \rangle$ and $\sigma_2 = \langle \mathbb{G}_2; \mathbb{B}_2; \mathbb{V}_2 \rangle$ such that local variables of one state are disjunct from all variables in the other state.

$$\sigma_1 \diamond_{\mathbb{V}} \sigma_2 ::= \langle \mathbb{G}_1 \uplus \mathbb{G}_2; \mathbb{B}_1 \wedge \mathbb{B}_2; (\mathbb{V}_1 \cup \mathbb{V}_2) \setminus \mathbb{V} \rangle$$

$$[\sigma_1] \diamond_{\mathbb{V}} [\sigma_2] ::= [\sigma_1 \diamond_{\mathbb{V}} \sigma_2].$$

For $\mathbb{V} = \emptyset$, we write $\sigma_1 \diamond \sigma_2$ and $[\sigma_1] \diamond [\sigma_2]$, respectively.

Merge Operator

Example

- Equality holds in both directions: merge or split

$$[\langle c(X); \top; \{X\}\rangle] \diamond_{\{X\}} [\langle \emptyset; X = 1; \{X\}\rangle] = [\langle c(X); X = 1; \emptyset\rangle]$$

- Pay attention to global variables

$$[\langle c(X); \top; \emptyset\rangle] \diamond [\langle \emptyset; X = 1; \emptyset\rangle] = [\langle c(X); Y = 1; \emptyset\rangle]$$

- For $\diamond_{\mathbb{V}}$, the $\mathbb{V}$ variables act as a temporary bridge between the two states.

Merge Operator

Example (gcd)

$$\mathrm{gcd}(N) \backslash \mathrm{gcd}(M) \Leftrightarrow M \geq N \wedge N > 0 \mid \mathrm{gcd}(L), L = M\%N$$

State splitting: remove everything not required for rule application

$$\begin{aligned}
& [\langle \mathrm{gcd}(6), \mathrm{gcd}(3); \top; \emptyset \rangle] \\
\equiv\ & [\langle \mathrm{gcd}(M), \mathrm{gcd}(N); M \geq N \wedge N > 0 \wedge M = 6 \wedge N = 3; \emptyset \rangle] \\
=\ & [\langle \mathrm{gcd}(M), \mathrm{gcd}(N); M \geq N \wedge N > 0; \{N, M\} \rangle] \\
& \diamond_{\{N,M\}} [\langle \emptyset; M = 6 \wedge N = 3; \{N, M\} \rangle]
\end{aligned}$$

Monotonicity and State Splitting

or: how to switch between larger and smaller derivations

Lemma (Monotonicity)

If $[\sigma] \rightarrowtail [\tau]$ *then* $[\sigma] \diamond_{\mathbb{V}} [\sigma'] \rightarrowtail [\tau] \diamond_{\mathbb{V}} [\sigma']$ *for all* $\mathbb{V}$.

- For any given derivation, you can extend start and result state
- For any derivation, you can subtract from the start state and consider the remaining derivation

Monotonicity and State Splitting

or: how to switch between larger and smaller derivations

Lemma (State Splitting with $\diamond_{\mathbb{V}}$)

Let the state $[\sigma]$ be applicable to a rule $r = (H_1 \backslash H_2 \Leftrightarrow G \mid B_c, B_b)$ with $\mathbb{V}$ being the variables occurring in H_1 and H_2. Then

$$\exists[\delta].[\sigma] = [\langle H_1 \uplus H_2; G; \mathbb{V}\rangle] \diamond_{\mathbb{V}} [\delta].$$

- Eliminates everything from current state that is not required for rule application
- Facilitates macro-step proofs
  - A macro-step is a terminating derivation starting from a rule state like $[\langle H_1 \uplus H_2; G; \mathbb{V}\rangle]$
  - Every finite derivation has a finite number of macro-steps (induction proofs)

State Splitting – Example

Example (gcd State Splitting (cont.))

$$
\begin{array}{ll}
 & [\langle \mathrm{gcd}(6), \mathrm{gcd}(3); \top; \emptyset\rangle] \\
= & [\langle \mathrm{gcd}(M), \mathrm{gcd}(N); M \geq N \wedge N > 0; \{N, M\}\rangle] \\
 & \diamond_{\{N,M\}} [\langle \emptyset; M = 6 \wedge N = 3; \{N, M\}\rangle] \\
\rightarrowtail & [\langle \mathrm{gcd}(N), \mathrm{gcd}(L); M \geq N \wedge N > 0 \wedge L = M\%N; \{N, M\}\rangle] \\
 & \diamond_{\{N,M\}} [\langle \emptyset; M = 6 \wedge N = 3; \{N, M\}\rangle] \\
= & [\langle \mathrm{gcd}(N), \mathrm{gcd}(L); M \geq N \wedge N > 0 \wedge L = M\%N \wedge M = 6 \wedge N = 3; \emptyset\rangle] \\
= & [\langle \mathrm{gcd}(3), \mathrm{gcd}(0); \top; \emptyset\rangle]
\end{array}
$$

State Splitting in Semantics

Definition (Operational Semantics with State Splitting)

Apply:

$$\frac{r \;@\; H_1 \backslash H_2 \Leftrightarrow G \mid B_c, B_b \qquad \mathbb{V} = \mathrm{vars}(H_1, H_2)}{[\langle H_1 \uplus H_2; G; \mathbb{V}\rangle] \rightarrowtail^r [\langle H_1 \uplus B_c; G \wedge B_b; \mathbb{V}\rangle]}$$

Extend:

$$\frac{[\sigma] \rightarrowtail^r [\tau]}{[\sigma] \diamond_{\mathbb{V}} [\delta] \rightarrowtail^r [\tau] \diamond_{\mathbb{V}} [\delta]}$$

- **Apply**: minimal description of requirements and consequences of rule application
- **Extend**: arbitrary extensions possible (for any $\mathbb{V}$)

Algebraic Properties of $\diamond$

or: how to make further use of $\diamond$

Lemma

$(\Sigma/\equiv, \diamond)$ *is a commutative monoid (for* $\mathbb{V} = \emptyset$*).*

Commutative monoid:

- Totality
- Associativity
- Commutativity
- Identity element

- commutative monoid implies algebraic preordering
  - $[\sigma] \lhd [\tau]$ if $\exists[\delta].[\tau] = [\sigma] \diamond [\delta]$
  - in fact, $\lhd$ is a partial order (antisymmetric)

Summary: Merging and Splitting

Take Home Messages

- Merge operator $\diamond$ formalizes monotonicity
- State splitting extracts state components not required for rule application

Overall Summary: Presented Tools

Take Home Messages

- State equivalence
  - Axiomatic definition, decidable criterion, implementation available
- Operational Semantics
  - Equivalence-based op.sem.
  - Rewriting of equivalence classes
- Merge Operator
  - Formalizes monotonicity

Available Literature

- Frank Raiser, Hariolf Betz, Thom Frühwirth, *Equivalence of CHR States Revisited*, CHR 2009
  - axiomatic state equivalence, decidable criterion, new formulations of operational semantics
- Hariolf Betz, Frank Raiser, Thom Frühwirth, *A Complete and Terminating Execution Model for Constraint Handling Rules*, ICLP 2010
  - extension for propagation rules based on persistent constraints
  - full version available as technical report 1/2010 at Ulm University
- Frank Raiser, *Graph Transformation Systems in Constraint Handling Rules: Improved Methods for Program Analysis*, PhD thesis, Ulm University
  - available soon (hopefully)
  - covers everything in this talk