State Equivalence and Persistent Constraints

Joint work with Hariolf Betz and Thom Frühwirth

Table of Contents

Motivation

- CHR is becoming harder to analyze
  - operational semantics given by more and more rules
  - implementations of propagation rules break with logic

Motivation

- CHR is becoming harder to analyze
  - operational semantics given by more and more rules
  - implementations of propagation rules break with logic
- Problems in formal foundation:
  - state equivalence: never properly defined, yet always used
  - propagation history: state-of-the-art, yet hard to analyze
  - large transition systems: large, mostly boring, case distinction proofs

Goals

- properly define state equivalence of CHR states
  - avoids reinventing the wheel (happened about a dozen times already)
  - should be intuitive, simple, and easy to apply in formal reasoning
- reduce the transition system
  - CHR in it's most abstract form only needs one rule to describe operational semantics
- find a better way to deal with trivial non-termination
  - should be logically sound and complete

Motivation

- our goal: define state equivalence in CHR that is:
  - sound – declaratively and operationally
  - typical – no need to re-invent the wheel
  - axiomatic – for minimality
  - telling – each axiom tells an intention
  - elegant – to use for positive and negative proofs

Intuitive Understanding

- intuitive understanding when states are equivalent

Example Cases

Example 1:

$$\langle c(X), \top, \emptyset \rangle \equiv \langle c(Y), \top, \emptyset \rangle$$

- renaming of local variables

Intuitive Understanding

- intuitive understanding when states are equivalent

Example Cases

Example 2:

$$\langle c(X), X = 0, \{X\} \rangle \equiv \langle c(0), X = 0, \{X\} \rangle$$

- equality substitutions

Intuitive Understanding

- intuitive understanding when states are equivalent

Example Cases

Example 3:

$$\langle \top, X \geq 0 \land X \leq 0 \land Y = 0, \{X\} \rangle \equiv \langle \top, X = 0, \{X\} \rangle$$

- logically equivalent built-in stores

Intuitive Understanding

- intuitive understanding when states are equivalent

Example Cases

Example 4:

$$\langle c(0), \top, \{X\}\rangle \equiv \langle c(0), \top, \emptyset\rangle$$

- unused global variables
- often excluded, because operational semantics never changes the global variables

Intuitive Understanding

- intuitive understanding when states are equivalent

Example Cases

Example 5:

$$\langle c(X), \top, \{X\} \rangle \not\equiv \langle c(Y), \top, \{Y\} \rangle$$

- renaming of global variables is not equivalent

Available Definitions

- problem: we have differing (not just different!) definitions

	Ex.1	Ex.2	Ex. 3	Ex.4	Ex. 5
	$\equiv$	$\not\equiv$	$\not\equiv$	$\not\equiv$	$\equiv$
	$\equiv$	$\not\equiv$	$\equiv$	$\equiv$	$\not\equiv$
	$\equiv$	$\not\equiv$	$\equiv$	$\not\equiv$	$\not\equiv$
	$\equiv$	$\equiv$	$\equiv$	$\not\equiv$	$\not\equiv$
	$\not\equiv$	$\equiv$	$\equiv$	$\not\equiv$	$\not\equiv$
Desired	$\equiv$	$\equiv$	$\equiv$	$\equiv$	$\not\equiv$

(sources for these definitions are given in the CHR'09 paper)

An Axiomatic Definition

Definition (State Equivalence)

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

1. *(Substitution)* $\langle \mathbb{G}, x \doteq t \wedge \mathbb{B}, \mathbb{V} \rangle \equiv \langle \mathbb{G}\left[x/t\right], x \doteq t \wedge \mathbb{B}, \mathbb{V} \rangle$
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)* $\langle \mathbb{G}, \perp, \mathbb{V} \rangle \equiv \langle \mathbb{G}', \perp, \mathbb{V} \rangle$

An Axiomatic Definition

- only four telling axioms are required
- all desired intuitions (prev. examples) satisfied
- axioms facilitate elegant positive proofs

Properties of Axiomatic Definition

- properties directly derivable from the above axioms

Properties

- renaming of local variables
- partial substitutions
- logical equivalence

- interesting: renaming of local variables *not* needed as axiom
- remaining problems: negative and automatic proofs

Decision Criterion

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.

$$\sigma \equiv \sigma' \text{ iff } \mathcal{CT} \models \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{G} = \mathbb{G}') \wedge \mathbb{B}')) \wedge \forall(\mathbb{B}' \rightarrow \exists \bar{y}.((\mathbb{G} = \mathbb{G}') \wedge \mathbb{B}))$$

- simplifies negative proofs and allows automatic proof
- implementation available (cf. talk tomorrow by Johannes Langbein)

Table of Contents

Impact on Operational Semantics

Claim

Given two equivalent states, we can apply the same rules to them and get equivalent results.

- claim is intuitively clear
- should hold for every sensible definition
- however: no proof existed so far

Impact on Operational Semantics

Theorem

Let $\sigma \equiv \sigma'$ *and* $\sigma \rightarrowtail^r \tau$*, then* $\sigma' \rightarrowtail^r \tau' \equiv \tau$*.*

- using our results the proof is now available
- $\rightsquigarrow$ with this knowledge we can take another look at the operational semantics
- Note: proof assumes no propagation rules and abstract semantics (not refined)

New Formulation

Definition (Operational Semantics)

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

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

- no matchings, no syntactic equalities, no substitutions
- only state equivalence, i.e. only a few axioms
- most elegant formulation we have seen so far

New Formulation

- a new point of view on CHR: rewriting equivalence classes of states

Definition

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

New Formulation

- a new point of view on CHR: rewriting equivalence classes of states

Definition

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

One rule to rule them all, One rule to match them,
One rule to rewrite them all and in the $\equiv$-class collect them

Example Computation

Example (Computation using State Equivalence)

$$\mathrm{sum}(X), \mathrm{sum}(Y) \Leftrightarrow \mathrm{sum}(Z), Z = X + Y$$

$$\sigma = \quad \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1, \{A\} \rangle$$

Example Computation

Example (Computation using State Equivalence)

$$\mathrm{sum}(X), \mathrm{sum}(Y) \Leftrightarrow \mathrm{sum}(Z), Z = X + Y$$

$$\begin{array}{rl} \sigma = & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1, \{A\} \rangle \\ \overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \end{array}$$

Example Computation

Example (Computation using State Equivalence)

$$\mathrm{sum}(X), \mathrm{sum}(Y) \Leftrightarrow \mathrm{sum}(Z), Z = X + Y$$

$$\begin{array}{ll} \sigma = & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1, \{A\} \rangle \\ \overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \\ \overset{subst}{\equiv} & \langle \mathrm{sum}(X), \mathrm{sum}(Y), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \end{array}$$

Example Computation

Example (Computation using State Equivalence)

$$\mathrm{sum}(X), \mathrm{sum}(Y) \Leftrightarrow \mathrm{sum}(Z), Z = X + Y$$

$$\begin{aligned}
\sigma = \quad & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1, \{A\} \rangle \\
\overset{\mathcal{CT}}{\equiv} \quad & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \\
\overset{subst}{\equiv} \quad & \langle \mathrm{sum}(X), \mathrm{sum}(Y), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \\
\rightarrowtail \quad & \langle \mathrm{sum}(Z), A = 1 \wedge A = X \wedge Y = 2 \wedge Z = X + Y, \{A\} \rangle
\end{aligned}$$

Example Computation

Example (Computation using State Equivalence)

$$\mathrm{sum}(X), \mathrm{sum}(Y) \Leftrightarrow \mathrm{sum}(Z), Z = X + Y$$

$$\begin{array}{rl}
\sigma = & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1, \{A\} \rangle \\
\overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \\
\overset{\mathit{subst}}{\equiv} & \langle \mathrm{sum}(X), \mathrm{sum}(Y), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \\
\rightarrowtail & \langle \mathrm{sum}(Z), A = 1 \wedge A = X \wedge Y = 2 \wedge Z = X + Y, \{A\} \rangle \\
\overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(Z), A = 1 \wedge Z = 3, \{A\} \rangle
\end{array}$$

Example Computation

Example (Computation using State Equivalence)

$$\mathrm{sum}(X), \mathrm{sum}(Y) \Leftrightarrow \mathrm{sum}(Z), Z = X + Y$$

$$\begin{array}{rl}
\sigma = & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1, \{A\} \rangle \\
\overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \\
\overset{subst}{\equiv} & \langle \mathrm{sum}(X), \mathrm{sum}(Y), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \\
\rightarrowtail & \langle \mathrm{sum}(Z), A = 1 \wedge A = X \wedge Y = 2 \wedge Z = X + Y, \{A\} \rangle \\
\overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(Z), A = 1 \wedge Z = 3, \{A\} \rangle \\
\overset{subst}{\equiv} & \langle \mathrm{sum}(3), A = 1 \wedge Z = 3, \{A\} \rangle
\end{array}$$

Example Computation

Example (Computation using State Equivalence)

$$\mathrm{sum}(X), \mathrm{sum}(Y) \Leftrightarrow \mathrm{sum}(Z), Z = X + Y$$

$$\begin{array}{ll}
\sigma = & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1, \{A\} \rangle \\
\overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \\
\overset{subst}{\equiv} & \langle \mathrm{sum}(X), \mathrm{sum}(Y), A = 1 \wedge A = X \wedge Y = 2, \{A\} \rangle \\
\rightarrowtail & \langle \mathrm{sum}(Z), A = 1 \wedge A = X \wedge Y = 2 \wedge Z = X + Y, \{A\} \rangle \\
\overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(Z), A = 1 \wedge Z = 3, \{A\} \rangle \\
\overset{subst}{\equiv} & \langle \mathrm{sum}(3), A = 1 \wedge Z = 3, \{A\} \rangle \\
\overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(3), A = 1, \{A\} \rangle = \tau
\end{array}$$

Example Computation

Example (Computation using State Equivalence)

$$\mathrm{sum}(X), \mathrm{sum}(Y) \Leftrightarrow \mathrm{sum}(Z), Z = X + Y$$

$$\begin{array}{rl}
\sigma = & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1, \{A\}\rangle \\
\overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(A), \mathrm{sum}(2), A = 1 \wedge A = X \wedge Y = 2, \{A\}\rangle \\
\overset{\mathit{subst}}{\equiv} & \langle \mathrm{sum}(X), \mathrm{sum}(Y), A = 1 \wedge A = X \wedge Y = 2, \{A\}\rangle \\
\rightarrowtail & \langle \mathrm{sum}(Z), A = 1 \wedge A = X \wedge Y = 2 \wedge Z = X + Y, \{A\}\rangle \\
\overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(Z), A = 1 \wedge Z = 3, \{A\}\rangle \\
\overset{\mathit{subst}}{\equiv} & \langle \mathrm{sum}(3), A = 1 \wedge Z = 3, \{A\}\rangle \\
\overset{\mathcal{CT}}{\equiv} & \langle \mathrm{sum}(3), A = 1, \{A\}\rangle = \tau
\end{array}$$

Or simpler: $[\sigma] \rightarrowtail [\tau]$

Table of Contents

Motivation

- problems with token-store approach to trivial non-termination
  - declarativity – state definitions often contain non-declarative elements (eg. token store, constraint identifiers,...)
  - concurrency – complicated by token store
- existing abstract formulations are declarative and concurrent, however suffer from trivial non-termination of propagation rules
  - including the definition we just presented

Our Approach

- Approach #1: introduce persistent constraints
  - persistent constraints loosely correspond to banged resources in linear logic
  - they allow a finite representation of the result of any number of propagation rule firings
- Approach #2: make transition relation irreflexive
  - together with #1 solves trivial non-termination

CHR States with Persistent Constraints

- split CHR constraint store into two stores to distinguish linear and persistent constraints

Definition ($\omega_!$-State)

A $\omega_!$*-state* is a tuple of the form $\langle \mathbb{L}, \mathbb{P}, \mathbb{B}, \mathbb{V} \rangle$, where $\mathbb{L}$ and $\mathbb{P}$ are multisets of CHR constraints called the *linear (CHR) store* and *persistent (CHR) store*, respectively. $\mathbb{B}$ is a conjunction of built-in constraints and $\mathbb{V}$ is a set of variables.

- all components occur in a state's (linear) logical reading

Equivalence of $\omega_!$-States

Definition (Equivalence of $\omega_!$-States)

Equivalence between $\omega_!$-states is the smallest equivalence relation $\equiv$ over $\omega_!$-states that satisfies the following conditions:

1. equality modulo substitution
2. equivalence transformation of the built-in store
3. omission of non-occurring global variables
4. equivalence of failed states
5. contraction – $\langle \mathbb{L}, P \uplus P \uplus \mathbb{P}, \mathbb{B}, \mathbb{V} \rangle \equiv \langle \mathbb{L}, P \uplus \mathbb{P}, \mathbb{B}, \mathbb{V} \rangle$

New Operational Semantics

Definition ($\omega_!$-Transitions)

$$\frac{r \mathbin{@} (H_1^l \uplus H_1^p) \backslash (H_2^l \uplus H_2^p) \Leftrightarrow G \mid B_c, B_b \quad H_2^l \neq \emptyset \quad \sigma \neq \tau}{\begin{array}{c}\sigma = [\langle H_1^l \uplus H_2^l \uplus \mathbb{L}, H_1^p \uplus H_2^p \uplus \mathbb{P}, G \wedge \mathbb{B}, \mathbb{V}\rangle] \\ \rightarrowtail_{\omega_!} [\langle H_1^l \uplus B_c \uplus \mathbb{L}, H_1^p \uplus H_2^p \uplus \mathbb{P}, G \wedge \mathbb{B} \wedge B_b, \mathbb{V}\rangle] = \tau\end{array}}$$

$$\frac{r \mathbin{@} (H_1^l \uplus H_1^p) \backslash H_2^p \Leftrightarrow G \mid B_c, B_b \quad \sigma \neq \tau}{\begin{array}{c}\sigma = [\langle H_1^l \uplus \mathbb{L}, H_1^p \uplus H_2^p \uplus \mathbb{P}, G \wedge \mathbb{B}, \mathbb{V}\rangle] \\ \rightarrowtail_{\omega_!} [\langle H_1^l \uplus \mathbb{L}, H_1^p \uplus H_2^p \uplus B_c \uplus \mathbb{P}, G \wedge \mathbb{B} \wedge B_b, \mathbb{V}\rangle] = \tau\end{array}}$$

Termination Behavior

Example (Transitive Hull)

$$e(X,Y), e(Y,Z) \Longrightarrow e(X,Z)$$

- transitive hull program terminates under $\omega_!$ for all possible inputs
- under all token-store based operational semantics non-termination occurs if there is a cycle contained in the input

Termination Behavior

Example (Non-Termination)

$$a \Longrightarrow b$$

$$b, c(X) \Leftrightarrow c(X+1)$$

- terminates under ω_t
- non-termination under $\omega_!$

Comparison with Existing Operational Semantics

Comparison Table

	ω_{va}	ω_t	ω_r	ω_{set}	$\omega_!$
Termination:	-	+	+	+	+
Effective concurrency:	+	-	-	+	+
Declarative states:	+	-	-	-	+
Number of transition rules:	1	3	7	9	2
Multiset semantics:	+	+	+	-	+
Reduced non-determinism:	-	-	+	+	-

Conclusion

Contributions

- axiomatic definition of state equivalence
  - decidable necessary and sufficient criterion
- operational semantics on equivalence classes
  - facilitates program analysis
  - reduces complexity and length of proofs
- introduction of persistent constraints
  - adjusted state equivalence and operational semantics
  - new way to deal with trivial non-termination
  - differing behavior w.r.t. termination

Future Work

Future Work

- currently only works for range-restricted CHR programs
- further investigate termination behavior
- investigate extensions based on ω_t for $\omega_!$ (e.g. rule priorities)
- implement $\omega_!$ (preferrably in a concurrent setting)
- convince others to regard CHR as rewriting of equivalence classes

Thank You.