Equivalence of CHR States Revisited

Frank Raiser, Hariolf Betz, and Thom Frühwirth

Faculty of Engineering and Computer Sciences, Ulm University, Germany
`firstname.lastname@uni-ulm.de`

Abstract. While it is generally agreed-upon that certain classes of CHR states should be considered equivalent, no standard definition of equivalence has ever been established. Furthermore, the compliance of equivalence with rule application is generally assumed, but has never been proven. We systematically develop an axiomatic notion of state equivalence based on rule applicability and the declarative semantics. We supply the missing proof for its compliance with rule application and provide a proof technique to determine equivalence of given states. The compliance property leads to a simplified formulation of the operational semantics. Furthermore, it justifies a novel view based on equivalence classes of states which provides a powerful proof technique.

1 Introduction

While equivalence of states is apparently an elementary concept in Constraint Handling Rules (CHR), the community has never agreed on a standard definition of that concept up to now. A plethora of definitions of state equivalence has been introduced in various areas of application. For example, the operational equivalence algorithm compares two resulting states of different programs for equivalence [1]. Equivalence is the basis for invariants such as in [2]. Several definitions [3–6] have been introduced in the context of confluence considerations. Finally, from an operational point of view, it is clear that the normalization function of the operational semantics implicitly assumes a notion of state equivalence.

As the various authors had different intentions, the resulting definitions of state equivalence vary considerably. There is a general agreement that from an operational point of view any notion of state equivalence should be compliant with rule applications, i.e. for equivalent states the same rules are applicable and lead to equivalent results. However, this property has never been proven for any of the previously proposed definitions. Another general agreement is that from a declarative point of view the logical reading of equivalent states should also be equivalent.

Our aim is therefore to develop a definition of state equivalence that satisfies both the operational and the declarative view. Instead of defining a notion of state equivalence for a fixed problem setting, we intend a notion for which these generally agreed-upon properties hold. By construction, our definition of state equivalence then is compliant with rule application and the logical reading of states. Thus, it becomes a generic proof technique that can be applied to specific problems with the additional knowledge that the above-mentioned properties are satisfied.

In this paper, we make the following contributions:

- We justify a set of desirable properties for a general notion of state equivalence and present them in the form of example cases in Sect. 2.2.
- We give a concise overview of the existing definitions of state equivalence in Sect. 2.3 and compare their behavior with respect to our example cases in Sect. 2.4. We show that none of the existing definitions satisfies all of the example cases.
- We introduce an axiomatic definition of state equivalence in Sect. 3.1 along with several useful properties following from that definition.
- We present a necessary, sufficient, and decidable criterion for determining equivalence of states in Sect. 3.2.
- In Sect. 3.3 we show that our definition of state equivalence complies with all of the example cases defined in Sect. 2.2.
- In Sect. 4.1, we show that our notion of equivalence leads to a clearer definition of the operational semantics. We prove its equivalence to the traditional definition and – for the first time in the literature – we show that state equivalence is indeed compliant with rule application.
- In Sect. 4.3, we present a view of the CHR transition system that is based on equivalence classes of states rather than individual states.

Finally, we present our conclusions and establish possible further research paths in Sect. 5.

2 Existing Equivalence Definitions

Constraint Handling Rules (CHR) [7–9] is a concurrent committed-choice rule-based programming language, originally developed as a portable language extension for the implementation of user-defined constraint solvers. In the main part of our paper, specific knowledge of CHR is not required. For a discussion of the operational semantics of CHR, refer to Sect. 4.

In this section, we evaluate existing definitions of state equivalence and postulate desirable properties of an equivalence relation over CHR states. To this end, we present several prototypical example cases of equivalent and non-equivalent CHR states in Sect. 2.2. In Sect. 2.3 we concisely introduce the different notions of state equivalence that have been proposed so far, before we investigate how these notions apply to our example states in Sect. 2.4. We first define the syntax of CHR states and rules. The corresponding transition system is given in Sect. 4 where we apply our results from Sect. 3.

2.1 Preliminaries

In CHR, we distinguish two disjoint sets of constraints which we call *built-in constraints* and *CHR constraints.*

Definition 1 (CHR State). *A* CHR state σ *is a tuple* $\langle \mathbb{G}, \mathbb{B}, \mathbb{V} \rangle$*. The* goal $\mathbb{G}$ *is a multiset of CHR constraints. The* built-in constraint store $\mathbb{B}$ *is a conjunction of built-in constraints.* $\mathbb{V}$ *is a set of* global variables*.*

We use $\sigma, \sigma_0, \sigma_1, \ldots$ *to denote states and* Σ *to denote the set of all states.*

We found more elaborate definitions than Def. 1 unsuitable for our demand on the logical reading of equivalent states, because they contain information that is not reflected in the logical reading of the states, such as the propagation history [10]. Depending on the domain of application, our result can be extended to these definitions. Our notion of state clearly separates the three components that each have to be treated differently by state equivalence.

For any CHR state we distinguish three sets of variables according to the following definition.

Definition 2 (Variable Types). *For the variables occurring in a state* $\sigma = \langle \mathbb{G}, \mathbb{B}, \mathbb{V} \rangle$ *we distinguish three different types:*

1. *a variable* $v \in \mathbb{V}$ *is called a* global *variable*
2. *a variable* $v \notin \mathbb{V}$ *is called a* local *variable*
3. *a variable* $v \notin (\mathbb{V} \cup \mathbb{G})$ *is called a* strictly local *variable*

The following definition introduces the logical reading of CHR states.

Definition 3 (Logical Reading of CHR States).

Let σ *be a CHR state of the form* $\langle \{g_1, g_2, \ldots g_n\}, \mathbb{B}, \mathbb{V} \rangle$*. Then the logical reading of* σ *is:*

$$\bar{\exists}_{\mathbb{V}} \, (g_1 \wedge g_2 \wedge \ldots \wedge g_n \wedge \mathbb{B})$$

where $\bar{\exists}_{\mathbb{V}}$ *is existential quantification of all free variables except those in* $\mathbb{V}$*.*

A CHR program defines a set of rules, as given in Def. 4, by which the constraints in $\mathbb{G}$ are to be rewritten to a final solved form. The store $\mathbb{B}$ contains built-in constraints that have been posted to an underlying solver. These are assumed to be solved implicitly by the host language $\mathcal{H}$ according to a complete and decidable constraint theory $\mathcal{CT}$. The set $\mathbb{V}$ usually contains the variables that occur in the initial state of a computation and can be thought of as communication channels with the outside world.

Definition 4 (CHR rule).

A CHR rule *is of the following form*

$$H_1 \backslash H_2 \Leftrightarrow G | B_c \uplus B_b$$

where the head H_1, H_2 *consists of two multisets of CHR constraints, the* guard G *is a conjunction of built-in constraints, and the body consists of a multiset* B_c *of CHR constraints and a conjunction* B_b *of built-in constraints.*

2.2 Examples of CHR States

Let us now consider the following examples of equivalent and non-equivalent states to highlight the differences between existing definitions of state equivalence:

$$\langle c(X), \top, \emptyset\rangle \equiv \langle c(Y), \top, \emptyset\rangle \tag{1}$$
$$\langle c(X), X = 0, \{X\}\rangle \equiv \langle c(0), X = 0, \{X\}\rangle \tag{2}$$
$$\langle \top, X \geq 0 \wedge X \leq 0 \wedge Y = 0, \{X\}\rangle \equiv \langle \top, X = 0, \{X\}\rangle \tag{3}$$
$$\langle c(0), \top, \{X\}\rangle \equiv \langle c(0), \top, \emptyset\rangle \tag{4}$$
$$\langle c(X), \top, \{X\}\rangle \not\equiv \langle c(Y), \top, \{Y\}\rangle \tag{5}$$

The equivalences (1)-(3) are motivated by the fact that the same rules are applicable to these states with the same results.

As the states in equivalence (4) have the same logical reading $c(0)$ according to Def. 3 we require them to be equivalent. Note that unused global variables can practically occur, for example when applying rule $c(X) \Leftrightarrow c(0)$ to the state $\langle c(X), \top, \{X\}\rangle$. Concerning non-equivalence (5), note that X, Y are free variables and therefore the logical readings $c(X), c(Y)$ are not equivalent.

2.3 Existing Definitions

Over the last decade, the CHR community proposed various definitions for state equivalence. The following list identifies six distinct categories of equivalence definitions in the literature:

I The definitions based on variable renaming [1,4,11,12] are often as simple as stating that two states are equivalent (or variants) if they can be obtained by variable renaming only. These definitions arose from the notion of variance on terms.

II In [13] a definition is given that is based on renaming of local variables as well as logical equivalence of built-in stores.

III In [5] a similar definition is given for arbitrary binary relations rather than for CHR states only.

IV [14] gives another definition based on the so-called refined operational semantics [15] of CHR.

V [6] – a follow-up to [14] – extends the definition with the usage of a unifier instead of variable renaming.

VI In [16,17] a *normalization function* is defined. While we emphasize that this definition was not targeted towards determining state equivalence, we include it in this work due to its clear structure that is similar to our proposed definition. When we talk about equivalence with respect to normalization we implicitly assume that two states are equivalent iff their normalizations are syntactically equivalent.

2.4 Comparison of Existing Equivalence Definitions

We have applied each of the existing definitions to each of the example cases. The results are presented in Table 1. Each entry shows whether the two corresponding example states are considered equivalent or not, according to the category used in that row. The last row presents the results that we deem desirable. As we can see, none of the previously published definitions of state equivalence respects all of the example cases.

	(1)	(2)	(3)	(4)	(5)
Def. I	$\equiv$	$\not\equiv$	$\not\equiv$	$\not\equiv$	$\equiv$
Def. II	$\equiv$	$\not\equiv$	$\equiv$	$\equiv$	$\not\equiv$
Def. III	$\equiv$	$\not\equiv$	$\equiv$	$\equiv$	$\not\equiv$
Def. IV	$\equiv$	$\not\equiv$	$\equiv$	$\not\equiv$	$\not\equiv$
Def. V	$\equiv$	$\equiv$	$\equiv$	$\not\equiv$	$\not\equiv$
Def. VI	$\not\equiv$	$\equiv$	$\equiv$	$\not\equiv$	$\not\equiv$
Desired	$\equiv$	$\equiv$	$\equiv$	$\equiv$	$\not\equiv$

Table 1. Comparison of different state equivalence definitions

3 An Axiomatic Definition of Equivalence

In this section, we introduce an axiomatic definition of equivalence that satisfies all the desirable properties we identified in the previous section. We present our definition in Sect. 3.1 along with several properties. In Sect. 3.2 we give a necessary, sufficient, and decidable criterion to prove equivalence and non-equivalence between CHR states. In Sect. 3.3, we prove compliance with the example cases from Sect. 2.2.

3.1 Definition of State Equivalence

Our notion of state equivalence is given in the following definition.

Definition 5 (State Equivalence).
Equivalence between CHR states is the smallest equivalence relation $\equiv$ *over CHR states that satisfies the following conditions:*

1. (Equality as Substitution)

$$\langle \mathbb{G}, x \doteq t \wedge \mathbb{B}, \mathbb{V}\rangle \equiv \langle \mathbb{G}\,[x/t]\,, x \doteq t \wedge \mathbb{B}, \mathbb{V}\rangle$$

2. (Transformation of the Constraint Store) *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. (Omission of Non-Occurring Global Variables) *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. (Equivalence of Failed States)

$$\langle \mathbb{G}, \bot, \mathbb{V} \rangle \equiv \langle \mathbb{G}', \bot, \mathbb{V} \rangle$$

The axioms are chosen such that we can guarantee compliance with the operational semantics (cf. Sect. 4.1) as well as the logical readings of CHR states.

Firstly, names of local variables in CHR states are chosen non-deterministically upon execution. Hence, considering these names invariant with respect to state equivalence suggests itself. In combination, axiom 1 and axiom 2 guarantee this desired property (cf. Lemma 1:1).

Axiom 1 and axiom 2 are furthermore invariant with respect to rule applicability and comply with logical equivalence of the logical readings. The same holds for axiom 4: On the logical level, inconsistent logical readings are of course logically equivalent. It is necessary to guarantee compliance with the operational semantics as we justify in Sect. 4.2.

Axiom 3 suggests itself with regard to logical readings, since adding or removing global constraints results in syntactically identical and therefore indistinguishable logical readings. Operationally, unused global variables have no effect, so it stands to reason to consider them redundant.

Lemma 1 states several properties that follow from Def. 5.

Lemma 1 (Properties of State Equivalence). *The equivalence relation over CHR states given in Def. 5 has the following properties:*

1. (Renaming of Local Variables) *Let x, y be variables such that $x, y \notin \mathbb{V}$ and y does not occur in $\mathbb{G}$ or $\mathbb{B}$:*

$$\langle \mathbb{G}, \mathbb{B}, \mathbb{V} \rangle \equiv \langle \mathbb{G}[x/y], \mathbb{B}[x/y], \mathbb{V} \rangle$$

2. (Partial Substitution) *Let $\mathbb{G}[x \wr t]$ be a multiset where* some *occurrences of x are substituted with t:*

$$\langle \mathbb{G}, x \doteq t \wedge \mathbb{B}, \mathbb{V} \rangle \equiv \langle \mathbb{G}[x \wr t], x \doteq t \wedge \mathbb{B}, \mathbb{V} \rangle$$

3. (Logical Equivalence) *If*

$$\langle \mathbb{G}, \mathbb{B}, \mathbb{V} \rangle \equiv \langle \mathbb{G}', \mathbb{B}', \mathbb{V}' \rangle$$

then $\mathcal{CT} \models \exists \bar{y}.\mathbb{G} \wedge \mathbb{B} \leftrightarrow \exists \bar{y}'.\mathbb{G}' \wedge \mathbb{B}'$, where $\bar{y}, \bar{y}'$ are the local variables of $\langle \mathbb{G}, \mathbb{B}, \mathbb{V} \rangle, \langle \mathbb{G}', \mathbb{B}', \mathbb{V}' \rangle$, respectively.

Proof.

Property 1: *By transformation of the constraint store, we have that* $\langle \mathbb{G}, \mathbb{B}, \mathbb{V}\rangle$ *is equivalent to* $\langle \mathbb{G}, x \doteq y \wedge \mathbb{B}, \mathbb{V}\rangle$*. We apply equality as substitution and get* $\langle \mathbb{G}[x/y], x \doteq y \wedge \mathbb{B}, \mathbb{V}\rangle$ *which by transformation is equivalent to* $\langle \mathbb{G}[x/y], \mathbb{B}[x/y], \mathbb{V}\rangle$*.*

Property 2: *By substitution, we have that both* $\langle \mathbb{G}, \mathbb{B}, \mathbb{V}\rangle$ *and* $\langle \mathbb{G}[x \wr t], x \doteq t \wedge \mathbb{B}, \mathbb{V}\rangle$ *are equivalent to* $\langle \mathbb{G}[x/t], x \doteq t \wedge \mathbb{B}, \mathbb{V}\rangle$*. The equivalence relation is implicitly symmetric and transitive.*

Property 3: *All conditions given in Def. 5 correspond to valid logical equivalences:*

Definition 5:1 *preserves logical equivalence since*

$$\mathbb{G} \wedge x \doteq t \leftrightarrow \mathbb{G}[x/t] \wedge x \doteq t$$

Definition 5:2: *As* $\mathcal{CT} \models \exists \bar{s}.\mathbb{B} \leftrightarrow \exists \bar{s}'.\mathbb{B}'$ *and the variables in* $\bar{s}, \bar{s}'$ *do not occur in* $\mathbb{G}, \mathbb{G}'$*, we have*

$$\mathcal{CT} \models \exists \bar{y}.\mathbb{B} \wedge \mathbb{G} \leftrightarrow \exists \bar{y}'.\mathbb{B}' \wedge \mathbb{G}$$

Definition 5:3: *For a variable* X *that does not occur in* $\mathbb{B}$ *or* $\mathbb{G}$ *we obviously have*

$$\mathcal{CT} \models \exists X.\exists \bar{y}.\mathbb{B} \wedge \mathbb{G} \leftrightarrow \exists \bar{y}.\mathbb{B} \wedge \mathbb{G}$$

Definition 5:4 *preserves logical equivalence due to the* ex falso quodlibet *property.*

As logical equivalence is reflexive, transitive, and symmetric, Prop. 3 holds.

□

3.2 A Sufficient and Decidable Criterion for State Equivalence

Logical equivalence between $\exists \bar{y}.\mathbb{G} \wedge \mathbb{B}$ and $\exists \bar{y}'.\mathbb{G}' \wedge \mathbb{B}'$ is a necessary but not a sufficient condition for state equivalence between $\langle \mathbb{G}, \mathbb{B}, \mathbb{V}\rangle$ and $\langle \mathbb{G}', \mathbb{B}', \mathbb{V}'\rangle$ (cf. Lemma 1:3). This is due to the fact that unlike logical equivalence, state equivalence preserves the multiplicities of logically equivalent user-defined constraints. A similar condition which is also sufficient can be formulated in *linear logic* [18].

Theorem 1 gives a necessary and sufficient criterion for deciding state equivalence. Note that due to its preconditions it technically decides a smaller relation than $\equiv$, because it only applies to the case that local variables are renamed apart and the set of global variables is unchanged.

However, this restriction is not problematic for deciding equivalence in general. By Lemma 1 we are free to rename local variables apart and by Def. 5:3 we can adjust the sets of global variables to match. Therefore, Thm. 1 gives us a necessary and sufficient criterion for equivalence of arbitrary states: first we transform the states into equivalent states that satisfy the preconditions, then we apply the theorem. The transformation is straightforward and equivalence-preserving, hence, the result we get from the theorem applies to the original states by transitivity of $\equiv$. Finally, decidability of our criterion is a direct consequence of decidability of $\mathcal{CT}$.

Theorem 1 (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.*

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

Proof. Let $\mathcal{C}$ be a binary predicate on CHR states such that $\mathcal{C}(\langle \mathbb{G}, \mathbb{B}, \mathbb{V}\rangle, \langle \mathbb{G}', \mathbb{B}', \mathbb{V}\rangle)$ holds iff

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

$\Rightarrow$:

We show that each of the three implicit conditions – reflexivity, symmetry and transitivity – as well as all the four explicit conditions of Def. 5 are sound w.r.t. criterion $\mathcal{C}$.

Reflexivity: *Reflexivity is given as the following judgment is obviously true:*

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

Symmetry: *Symmetry of $\mathcal{C}$ is obvious.*

Transitivity: *Assume three states $\sigma = \langle \mathbb{G}, \mathbb{B}, \mathbb{V}\rangle, \sigma' = \langle \mathbb{G}', \mathbb{B}', \mathbb{V}\rangle, \sigma'' = \langle \mathbb{G}'', \mathbb{B}'', \mathbb{V}\rangle$ with distinct local variables $\bar{y}, \bar{y}', \bar{y}''$ such that $\mathcal{C}(\sigma, \sigma')$ and $\mathcal{C}(\sigma', \sigma'')$. By definition, we have:*

$$\begin{array}{ll}
\mathcal{CT} \models \forall(\mathbb{B} \rightarrow \exists\bar{y}'.((\mathbb{G} = \mathbb{G}') \wedge \mathbb{B}')) & (i) \\
\mathcal{CT} \models \forall(\mathbb{B}' \rightarrow \exists\bar{y}.((\mathbb{G} = \mathbb{G}') \wedge \mathbb{B})) & (ii) \\
\mathcal{CT} \models \forall(\mathbb{B}' \rightarrow \exists\bar{y}''.((\mathbb{G}' = \mathbb{G}'') \wedge \mathbb{B}'')) & (iii) \\
\mathcal{CT} \models \forall(\mathbb{B}'' \rightarrow \exists\bar{y}'.((\mathbb{G}' = \mathbb{G}'') \wedge \mathbb{B}')) & (iv)
\end{array}$$

From (i) and (iii) follows:

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

From (ii) and (iv) follows:

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

Consequently, $\mathcal{C}(\sigma, \sigma'')$.

Equality as Substitution: *Assume two states $\sigma = \langle \mathbb{G}, x \doteq t \wedge \mathbb{B}, \mathbb{V}\rangle, \sigma' = \langle \mathbb{G}[x/t], x \doteq t \wedge \mathbb{B}, \mathbb{V}\rangle$ with local variables $\bar{y}, \bar{y}'$. As $\mathcal{CT} \models \forall(x \doteq t \rightarrow (\mathbb{G} = \mathbb{G}[x/t]))$, we have $\mathcal{C}(\sigma, \sigma')$.*

Transformation of the Constraint Store: *Assume two states $\sigma = \langle \mathbb{G}, \mathbb{B}, \mathbb{V}\rangle, \sigma' = \langle \mathbb{G}, \mathbb{B}', \mathbb{V}\rangle$ with local variables $\bar{y}, \bar{y}'$ and strictly local variables $\bar{s}, \bar{s}'$ such that $CT \models \exists\bar{s}.\mathbb{B} \leftrightarrow \exists\bar{s}'.\mathbb{B}'$. This implies the following judgment:*

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

Hence, $\mathcal{C}(\sigma, \sigma')$.

Omission of Non-Occurring Global Variables: *Does not apply since σ, σ' share the set $\mathbb{V}$ of global variables.*

Equivalence of Failed States: *For any two failed states, we have states of the form $\langle \mathbb{G}, \bot, \mathbb{V}\rangle, \langle \mathbb{G}', \bot, \mathbb{V}\rangle$. The following judgment proves $\mathcal{C}(\langle \mathbb{G}, \bot, \mathbb{V}\rangle, \langle \mathbb{G}', \bot, \mathbb{V}\rangle)$:*

$$\mathcal{CT} \models \forall(\bot \rightarrow \exists \bar{y}'.((\mathbb{G} = \mathbb{G}') \wedge \bot)) \wedge \forall(\bot \rightarrow \exists \bar{y}.((\mathbb{G} = \mathbb{G}') \wedge \bot))$$

$\Leftarrow$*:*

We consider two CHR states $\sigma = \langle \mathbb{G}, \mathbb{B}, \mathbb{V}\rangle, \sigma' = \langle \mathbb{G}', \mathbb{B}', \mathbb{V}\rangle$ with local variables $\bar{y}$ and $\bar{y}'$. We assume that

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

If there does not exist a pairwise matching $\mathbb{G} = \mathbb{G}'$, we have $\mathbb{B} = \mathbb{B}' = \bot$, which proves that $\sigma \equiv \sigma'$ by Def. 5:4. In the following, we assume that a pairwise matching $\mathbb{G} = \mathbb{G}'$ does exist.

It follows from $\forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{G} = \mathbb{G}') \wedge \mathbb{B}')$ by Def. 5:2 that:

$$\sigma \equiv \langle \mathbb{G}, \mathbb{G} = \mathbb{G}' \wedge \mathbb{B} \wedge \mathbb{B}', \mathbb{V}\rangle$$

By Def. 5:1 we have:

$$\sigma \equiv \langle \mathbb{G}', \mathbb{G} = \mathbb{G}' \wedge \mathbb{B} \wedge \mathbb{B}', \mathbb{V}\rangle$$

From $\forall(\mathbb{B}' \rightarrow \exists \bar{y}.((\mathbb{G} = \mathbb{G}') \wedge \mathbb{B}))$ we get by Def. 5:2 that:

$$\sigma \equiv \langle \mathbb{G}', \mathbb{B}', \mathbb{V}\rangle = \sigma'$$

□

3.3 Example Cases Revisited

We claimed earlier that our definition of state equivalence replicates the desired equivalences and non-equivalences given in the examples in Sect. 2.2. Now we revisit these examples and prove their compliance with our definition.

Example (1)**:** $\langle c(X), \top, \emptyset\rangle \equiv \langle c(Y), \top, \emptyset\rangle$

Proof. Renaming of local variables (cf. Lemma 1:1) □

Example (2)**:** $\langle c(X), X = 0, \{X\}\rangle \equiv \langle c(0), X = 0, \{X\}\rangle$

Proof. Follows directly from Def. 5:1. □

Example (3)**:** $\langle \top, X \geq 0 \wedge X \leq 0 \wedge Y = 0, \{X\}\rangle \equiv \langle \top, X = 0, \{X\}\rangle$

Proof. Follows from Def. 5:2, as:

$$\mathcal{CT} \models \exists Y.(X \geq 0 \wedge X \leq 0 \wedge Y = 0) \leftrightarrow (X = 0)$$

□

Example (4)**:** $\langle c(0), \top, \{X\}\rangle \equiv \langle c(0), \top, \emptyset\rangle$

Proof. Follows directly from Def. 5:3. □

Example (5)**:** $\langle c(X), \top, \{X\}\rangle \not\equiv \langle c(Y), \top, \{Y\}\rangle$

Proof. Follows from Thm. 1, as $\mathcal{CT} \not\models \forall(\top \rightarrow c(X) = c(Y))$. □

4 Impact on the Operational Semantics

In this section, we discuss the impact of our definition of state equivalence on the operational semantics of CHR. Section 4.1 applies our notion of state equivalence to the traditional operational semantics. The resulting formulation is clearer and more lucid than the traditional one. More importantly, it enables us to prove that state equivalence is indeed compliant with rule applications. This important property – while generally assumed – has never been proven before. This in turn gives rise to a definition of the operational semantics based directly on equivalence classes of states which we present in Sect. 4.3.

4.1 A Simplified Formulation of the Operational Semantics

In this section, we present a formulation of the operational semantics based on state equivalence. Our definition is not only based on the traditional definition, but is also provably equivalent. Consider the following definition for the traditional operational semantics, adjusted from [9]:

Definition 6 (Traditional Operational Semantics). *For a CHR program $\mathcal{P}$, the state transition system $(\Sigma, \mapsto)$ is defined as follows, where $(r \ @ \ H_1 \setminus H_2 \Leftrightarrow G \mid B_c \uplus B_b)$ is a copy of a rule in $\mathcal{P}$ containing only fresh variables.*

$$\frac{\begin{array}{c}(r \ @ \ H_1 \setminus H_2 \Leftrightarrow G \mid B_c \uplus B_b) \text{ with fresh variables } \bar{y} \\ \mathcal{CT} \models \forall(\mathbb{B} \rightarrow \exists \bar{y}.(H_1 = H_1' \wedge H_2 = H_2' \wedge G))\end{array}}{\langle H_1' \uplus H_2' \uplus \mathbb{G}, \mathbb{B}, \mathbb{V}\rangle \mapsto^r \langle H_1' \uplus B_c \uplus \mathbb{G}, H_1 = H_1' \wedge H_2 = H_2' \wedge G \wedge B_b \wedge \mathbb{B}, \mathbb{V}\rangle}$$

Integrating the notion of state equivalence permits removing the matching that has traditionally been hidden in the complex formula $H_1 = H_1' \wedge H_2 = H_2'$. Furthermore, imposing a guard condition on $\mathcal{CT}$ becomes dispensable, leading to the following simplified operational semantics:

Definition 7 (Operational Semantics). *For a CHR program $\mathcal{P}$ we define the state transition system $(\Sigma, \rightarrowtail)$ as follows, where $(r \ @ \ H_1 \setminus H_2 \Leftrightarrow G \mid B_c \uplus B_b)$ is a copy of a rule in $\mathcal{P}$ containing only fresh variables.*

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

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

If the rule r is clear from the context or any rule is sufficient we simply write $\sigma \rightarrowtail \tau$. As usual $\rightarrowtail^*$ is the reflexive-transitive closure of $\rightarrowtail$.

Theorem 2 proves the equivalence of both definitions. An important aspect of this equivalence, and therefore the second rule of Def. 7, is that state equivalence is compliant with rule applications. It seems that the CHR community intuitively agrees this property holds for state equivalence. It is noteworthy that, to the best of our knowledge, the following is effectively the first published proof for this important property.

Theorem 2 (Equivalence of the Definitions). *For a CHR state σ we have*

1. *If $\sigma \rightarrowtail^r \tau$ then there exists a state $\tau' \equiv \tau$ with $\sigma \mapsto^r \tau'$*
2. *If $\sigma \mapsto^r \tau'$ then there exists a state $\tau \equiv \tau'$ with $\sigma \rightarrowtail^r \tau$*

Proof.

1: *Let r @ $H_1 \setminus H_2 \Leftrightarrow G \mid B_c \uplus B_b$, $\sigma = \langle H_1 \uplus H_2 \uplus \mathbb{G}, G \wedge \mathbb{B}, \mathbb{V}\rangle \rightarrowtail^r \tau = \langle H_1 \uplus B_c \uplus \mathbb{G}, G \wedge B_b \wedge \mathbb{B}, \mathbb{V}\rangle$, and $\sigma' \equiv \sigma$. Let $\bar{y}, \bar{y}'$ be the local variables of σ, σ' respectively.*
Since r uses only fresh variables, we can assume w.l.o.g. that the local variables of σ are renamed apart from the local variables of σ' and σ' is of the form $\langle H_1' \uplus H_2' \uplus \mathbb{G}', \mathbb{B}', \mathbb{V}\rangle$.
As $\sigma \equiv \sigma'$ we get by Thm. 1 that

$$\mathcal{CT} \models \forall(\mathbb{B}' \rightarrow \exists\bar{y}.((H_1' \uplus H_2' \uplus \mathbb{G}' = H_1 \uplus H_2 \uplus \mathbb{G}) \wedge G \wedge \mathbb{B}))$$

and consequently:

$$\mathcal{CT} \models \forall(\mathbb{B}' \rightarrow \exists\bar{y}.(H_1' = H_1 \wedge H_2' = H_2 \wedge G))$$

Therefore, we can apply the traditional operational semantics and get:

$$\sigma' \mapsto^r \langle H_1' \uplus B_c \uplus \mathbb{G}', H_1 = H_1' \wedge H_2 = H_2' \wedge G \wedge B_b \wedge \mathbb{B}', \mathbb{V}\rangle = \tau'$$

By the above and Def. 5:2 we get:

$$\tau' \equiv \langle H_1' \uplus B_c \uplus \mathbb{G}', H_1 = H_1' \wedge H_2 = H_2' \wedge \mathbb{G} = \mathbb{G}' \wedge G \wedge B_b \wedge \mathbb{B}' \wedge \mathbb{B}, \mathbb{V}\rangle$$

By Def. 5:1 and Def. 5:2 we then get:

$$\tau' \equiv \langle H_1 \uplus B_c \uplus \mathbb{G}, G \wedge B_b \wedge \mathbb{B}, \mathbb{V}\rangle = \tau$$

2: *Let $\sigma = \langle H_1' \uplus H_2' \uplus \mathbb{G}, \mathbb{B}, \mathbb{V}\rangle \mapsto^r \tau' = \langle H_1' \uplus B_c \uplus \mathbb{G}, H_1 = H_1' \wedge H_2 = H_2' \wedge G \wedge B_b \wedge \mathbb{B}, \mathbb{V}\rangle$.*
As $\mathcal{CT} \models \forall(\mathbb{B} \rightarrow \exists\bar{y}.(H_1 = H_1' \wedge H_2 = H_2' \wedge G))$ we apply Def. 5:2 to σ:

$$\sigma \equiv \langle H_1' \uplus H_2' \uplus \mathbb{G}, H_1 = H_1' \wedge H_2 = H_2' \wedge G \wedge \mathbb{B}, \mathbb{V}\rangle$$

Using Def. 5:1 we get by substitution:

$$\sigma \equiv \langle H_1 \uplus H_2 \uplus \mathbb{G}, H_1 = H_1' \wedge H_2 = H_2' \wedge G \wedge \mathbb{B}, \mathbb{V}\rangle$$

We can apply rule r according to Def. 7 now resulting in

$$\sigma \rightarrowtail^r \langle H_1 \uplus B_c \uplus \mathbb{G}, G \wedge B_b \wedge (H_1 = H_1' \wedge H_2 = H_2' \wedge \mathbb{B}), \mathbb{V}\rangle = \tau$$

By Def. 5:1 we get $\tau \equiv \tau'$. □

4.2 Termination on Failure

Both formulations of the operational semantics given above allow arbitrary rule applications on failed states. In the case of the traditional operational semantics, the applicability condition is of the form $\mathcal{CT} \models \forall(\mathbb{B} \rightarrow \ldots)$. This condition is trivially satisfied for any inconsistent built-in store due to the principle *ex contradictio quodlibet* (ECQ). In our formulation of the operational semantics arbitrary rule applications are due to the explicit equivalence of failed states established in Def. 5:4. For example, under a program $P = \{a \Leftrightarrow b\}$ both $\langle c, \bot, \emptyset\rangle \mapsto^* \langle b, \bot, \emptyset\rangle$ and $\langle c, \bot, \emptyset\rangle \rightarrowtail^* \langle b, \bot, \emptyset\rangle$ are correct.

For theoretical considerations such as correctness analyses, this property is intentional: Firstly, it corresponds to the ECQ property which holds in most logical formalisms. Secondly, allowing derivations from failed states preserves monotonicity with respect to strengthening the constraint store such that e.g. $\langle \mathbb{G}, \mathbb{B}, \mathbb{V}\rangle \mapsto^* \langle \mathbb{G}', \mathbb{B}', \mathbb{V}'\rangle$ implies $\langle \mathbb{G}, \mathbb{B} \wedge \bar{\mathbb{B}}, \mathbb{V}\rangle \mapsto^* \langle \mathbb{G}', \mathbb{B}' \wedge \bar{\mathbb{B}}, \mathbb{V}'\rangle$, regardless of whether $\mathbb{B} \wedge \bar{\mathbb{B}}$ is consistent.

In a practical implementation, however, this property would lead to trivial non-termination of any failed computation. Existing implementations consequently do not allow computation from failed states. Formally, this property can be captured by introducing $\mathcal{CT} \models \exists\mathbb{B}$ as an applicability condition. Alternatively, we can solve this issue by explicitly disallowing atomic derivation steps between equivalent states. Unlike the first solution, such a formulation also avoids trivial non-termination originating from rules of the form $H \Leftrightarrow G \mid H$.

4.3 Founding the Operational Semantics on Equivalence Classes

Having finally shown the compliance of state equivalence with rule application, we revisit the operational semantics once more: We know that a rule application is possible for all states equivalent to a state containing the head and guard of that rule and that we are free to choose any element of the set of equivalent result states. Therefore, the actual syntactical representation of a state is of no importance, leading to a different view on the transition system: instead of considering all syntactical representations as different states, we propose the following reformulation of the operational semantics based on equivalence classes over states.

Definition 8. *For a CHR program $\mathcal{P}$ we define the state transition system $(\Sigma/\equiv\ ,\ \rightarrowtail)$ as follows. The application of a rule r is based on a copy of it that contains only fresh variables.*

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

This concise definition of the operational semantics of CHR requires only one rule and combines all of our results on state equivalence and its behavior with regard to rule applications. Furthermore, it leads to simplifications in other research areas where state equivalence is relevant. For example, confluence analysis in CHR is made complicated by different syntactical representations. Figure 1 depicts that term rewriting systems (TRS) require all computations to reach the exact same final term. However, the corresponding CHR results demand equivalence of final states. Under the operational semantics as given in Def. 8, the original *diamond property* of confluence depicted in Fig. 1 found in the TRS literature applies directly to CHR.

$$\begin{tikzcd}
 & \sigma \arrow[dl, "*"'] \arrow[dr, "*"] & \\
\sigma_1 \arrow[dr, "*"'] & TRS & \sigma_2 \arrow[dl, "*"] \\
 & \sigma' &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \sigma \arrow[dl, "*"'] \arrow[dr, "*"] & \\
\sigma_1 \arrow[dr, "*"'] & CHR & \sigma_2 \arrow[dl, "*"] \\
 & \sigma_1' \equiv \sigma_2' &
\end{tikzcd}$$

Fig. 1. Diamond property of confluence in TRS and CHR

5 Conclusion

As the main result, we have proven the compliance of state equivalence with rule application. To the best of our knowledge, this is the first published proof of this important property for any notion of equivalence.

This property has significant impact on the formulation of the operational semantics of CHR: It allows for a considerably more compact and lucid definition of the operational semantics than the ones known in the literature. Furthermore, it justifies an operational semantics of CHR based on equivalence classes of states rather than individual states.

We have presented the first axiomatic definition of state equivalence in the literature. It is more intuitive than existing definitions and provides elegant proof techniques such as applied in our proof of Lemma 1. However, it is not always trivial to prove state equivalence – and more so: non-equivalence – of arbitrary CHR states using the axiomatic definition. Therefore, we have presented a necessary, sufficient, and decidable criterion for both equivalence and non-equivalence of states. Example case (5) in Sect. 3.3 shows an application of this criterion.

We have evaluated the previously published definitions of state equivalence for a set of example cases and shown that none of them satisfies all of the examples. Contrarily, our definition of state equivalence satisfies all of the example cases.

We expect that the notion of state extensions [6] leads to a further enhanced formulation of the operational semantics. This concept would allow to disregard parts of the state that are unaffected by rule application. This would lead to an even more compact definition of the state transition system.

As we have shown, the operational semantics based on equivalence classes allows a simplified formulation of confluence results for CHR. We furthermore plan to reinvestigate results from observable confluence, operational equivalence, and their combination under this context. Combined with the expressive possibilities of state extensions this should lead to more concise formulations and proofs of all those results.

References

1. Abdennadher, S., Frühwirth, T.: Operational equivalence of CHR programs and constraints. In Jaffar, J., ed.: Principles and Practice of Constraint Programming, CP 1999. Volume 1713 of Lecture Notes in Computer Science., Springer-Verlag (1999) 43–57
2. Raiser, F., Frühwirth, T.: Strong joinability analysis for graph transformation systems in CHR. In: 5th International Workshop on Computing with Terms and Graphs, TERMGRAPH'09. (2009)
3. Abdennadher, S., Frühwirth, T.W., Meuss, H.: On confluence of constraint handling rules. In: Principles and Practice of Constraint Programming, 2nd International Conference. (1996) 1–15
4. Frühwirth, T., Pierro, A.D., Wiklicky, H.: Probabilistic constraint handling rules. In: Electronic Notes in Theoretical Computer Science. (2002) 1–16
5. Haemmerlé, R., Fages, F.: Abstract critical pairs and confluence of arbitrary binary relations. In: RTA '07: Proc. 18th Intl. Conf. Term Rewriting and Applications. Volume 4533 of Lecture Notes in Computer Science., Paris, France, Springer-Verlag (June 2007)
6. Duck, G.J., Stuckey, P.J., Sulzmann, M.: Observable confluence for constraint handling rules. In Dahl, V., Niemelä, I., eds.: Logic Programming, 23rd International Conference, ICLP 2007. Volume 4670 of Lecture Notes in Computer Science., Porto, Portugal, Springer-Verlag (September 2007) 224–239
7. Frühwirth, T.: Theory and practice of constraint handling rules. Journal of Logic Programming, Special Issue on Constraint Logic Programming **37**(1-3) (October 1998) 95–138
8. Frühwirth, T., Abdennadher, S.: Essentials of Constraint Programming. Springer-Verlag (2003)
9. Frühwirth, T.: Constraint Handling Rules. Cambridge University Press (2009) draft.
10. Sneyers, J., Van Weert, P., Schrijvers, T., De Koninck, L.: As time goes by: Constraint Handling Rules – A survey of CHR research between 1998 and 2007. Submitted to *Journal of Theory and Practice of Logic Programming* (2008)
11. Meister, M.: Efficient Declarative Programming. PhD thesis, Ulm University, Ulm, Germany (2007)
12. Duck, G.J.: Compilation of Constraint Handling Rules. PhD thesis, University of Melbourne, Australia (December 2005)
13. Raiser, F., Tacchella, P.: On Confluence of Non-terminating CHR Programs. In Djelloul, K., Duck, G.J., Sulzmann, M., eds.: Constraint Handling Rules, 4th Workshop, CHR 2007, Porto, Portugal (September 2007) 63–76
14. Duck, G.J., Stuckey, P.J., Sulzmann, M.: Observable confluence for constraint handling rules. In: Constraint Handling Rules, Third Workshop, CHR 2006. (2006)

15. Duck, G.J., Stuckey, P.J., de la Banda, M.J.G., Holzbaur, C.: The refined operational semantics of constraint handling rules. In: Logic Programming, 20th International Conference, ICLP 2004. (2004) 90–104
16. Abdennadher, S., Frühwirth, T., Meuss, H.: Confluence and semantics of constraint simplification rules. Constraints **4**(2) (1999) 133–165
17. Abdennadher, S.: Rule-based Constraint Programming: Theory and Practice. Habilitationsschrift, Institute of Computer Science, LMU, Munich, Germany (July 2001)
18. Betz, H., Frühwirth, T.: A linear-logic semantics for Constraint Handling Rules. In: Principles and Practice of Constraint Programming, 11th International Conference, CP 2005. Volume 3709 of Lecture Notes in Computer Science., Sitges, Spain, Springer-Verlag (October 2005) 137–151