facultad de informática

universidad politécnica de madrid

On the Confluence of CHR Analytical Semantics

Rémy Haemmerlé
Universidad Politécnica de Madrid &
IMDEA Software Institute, Spain

TR Number CLIP2/2014.0

On the Confluence of CHR Analytical Semantics

Technical Report Number: CLIP2/2014.0
July, 2014

Abstract

In this technical note we study the confluence of the analytical semantics for Constraint Handling Rules (CHR). In particular we demonstrate that confluence of CHR programs under the classical multiset semantics implies confluence under this set-based semantics.

Contents

In this technical note we present some results on the confluence of the analytical semantics for Constraint Handling Rules (CHR) [3]. In paricular, the note contains the two following contributions: In Section 3.1 we present different notions of confluence and show they are equivalent. In Section 3.2 we demonstrate te confluence of CHR programs under the classical multiset semantics implies confluence under this set-based semantics.

1 The General Framework

In this section, we introduce the syntax, the declarative semantics, and the analytical operational semantics for CHR [2]. For this purpose, we will assume a language of *(built-in) constraints* containing the equality $=$, $\bot$, and $\top$ over some theory $\mathcal{T}$. We define *(user-defined) atoms* using a different set of predicate symbols.

In the following, variables will be denoted by lower case letters from the end of the alphabet, such as x, y, z, ..., while atoms and atomic constraints will be indicated by lowercase letters from the beginning of the alphabet, such as a, b, c, d, ... Sets of constraints and atoms will be denoted by blackboard capital letters, such as $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$, $\mathbb{D}$. By a slight abuse of notation, we will confuse multisets and conjunctions of constraints and atoms, forget braces around such multisets, and use commas for their unions.

1.1 States

A *branch* is a tuple $(\mathbb{A}|\mathbb{C})$, where $\mathbb{A}$ (the *user store*) is a finite multiset of user-defined atoms and $\mathbb{C}$ (the *built-in store*) is a finite conjunction of atomic built-in constraints. Sets of branches can be denoted by Greek letters, Δ or Γ. By a slight abuse of notation, we will confuse multiset of branches, forget braces around such multisets, and use semicolons for their unions. A *state* is a tuple of $\langle\Gamma\rangle_{\vec{x}}$ where Γ is a multiset of branches and $\vec{x}$ (the *global variables*) a finite set of variables. Unsurprisingly, the *local variables* of a state are those variables of the state that are not global. States are always —implicitly— considered modulo alpha-conversion of their local variables.

When no confusion can occur, we will syntactically merge user atoms and built-in constraints within a branch. For the sake of conciseness, we will sometimes denote disjunctions of branches within a state as a finite family. For instance, a state of the form

$$\langle(\mathbb{A}_1|\mathbb{C}_1)\,;\ldots;(\mathbb{A}_n|\mathbb{C}_n)\rangle_{\vec{x}}$$

may be denoted by $\left\langle(\mathbb{A}_i|\mathbb{C}_i)_{i\in I}\right\rangle_{\vec{x}}$, where I stands for the set of indices $\{1,\ldots,n\}$. The set of all states will be denoted by Σ.

1.2 Programs

A *program* is a finite set of rules of the following form:

$$(r \mathbin{@} \mathbb{H} \iff \mathbb{G} \mid (\mathbb{B}_1|\mathbb{C}_1);\ldots;(\mathbb{B}_n|\mathbb{C}_n))$$

where $\mathbb{H}$ (the *head*) is a nonempty set of atoms, $\mathbb{G}$ (the *neck* or *guard*) is a conjunction of constraints, the $(\mathbb{B}_i|\mathbb{C}_i)$ are branches forming the so-called *body*, and r is an identifier assumed

unique in the program. An empty guard $\top$ can be omitted with the symbol $|$. For the sake of simplicity, we will always assume the guard and the body of a same rule share only variables that appear in their head. Disjunctions of branches within the body of a rule may be denoted by a finite family of branches in a way similar to branches within a state. The *local variables* of a rule are those variables that appear free in the body but do not appear either in the head or in the guard.

In the following, we distinguish special cases of rules.

- The so-called *backward rules*, which are those rules with the head repeated as an independent branch in the body; that is to say, rules of the form:

$$(r \mathbin{@} \mathbb{H} \iff \mathbb{G} \mid (\mathbb{B}_1|\mathbb{C}_1)\,;\ldots;(\mathbb{B}_n|\mathbb{C}_n)\,;(\mathbb{H}|\mathbb{G}\rho))$$

  where ρ renames $(\mathrm{fv}(\mathbb{G}) \setminus \mathrm{fv}(\mathbb{H}))$ by fresh variables.
  Such a rule can be written with the alternative syntax:

$$(r \mathbin{@} \mathbb{H} \Longleftarrow \mathbb{G} \mid (\mathbb{B}_1|\mathbb{C}_1)\,;\ldots;(\mathbb{B}_n|\mathbb{C}_n))$$

- The so-called *forward rules* (or propagation rules), which are those rules with the head repeated in each branch of the body; that is to say, rules of the form:

$$(r \mathbin{@} \mathbb{H} \iff \mathbb{G} \mid (\mathbb{H},\mathbb{B}_1|\mathbb{G}\rho,\mathbb{C}_1)\,;\ldots;(\mathbb{H},\mathbb{B}_n|\mathbb{G}\rho,\mathbb{C}_n))$$

  where ρ renames $(\mathrm{fv}(\mathbb{G}) \setminus \mathrm{fv}(\mathbb{H}))$ by fresh variables.
  Such a rule can be written with the alternative syntax:

$$(r \mathbin{@} \mathbb{H} \Longrightarrow \mathbb{G} \mid (\mathbb{B}_1|\mathbb{C}_1)\,;\ldots;(\mathbb{B}_n|\mathbb{C}_n))$$

Before considering the logical reading of programs, we introduce our running example. This is an enhanced version of the classic CHR introductory example.

1.3 Analytical Operation Semantics

The operational semantics of our language can be represented by a simple transition relation defined modulo a state equivalence.

Definition 1. *Given two multiset of atoms $(a_1,\ldots,a_m)$ and $(b_1,\ldots,b_n)$, we define an* evidence of the set-inclusion of $(a_1,\ldots,a_m)$ into $(b_1,\ldots,b_n)$ *as a conjunction of equality constraints $(a_1 = b_{i_1} \wedge \cdots \wedge a_m = b_{i_m})$ where $i_1,\ldots,i_m \subseteq \{1,\ldots,n\}$. Given two states $\langle(\mathbb{A}_i|\mathbb{C}_i)_{i\in I}\rangle_{\vec{x}}$ and $\langle(\mathbb{B}_j|\mathbb{D}_j)_{j\in J}\rangle_{\vec{y}}$ we will say that the branch $(\mathbb{A}_i|\mathbb{C}_i)$ is* covered *by the branch $(\mathbb{B}_j|\mathbb{D}_j)$ ($i \in I$ and $j \in J$), if the following implication holds:*

$$\mathcal{T} \vDash \mathbb{C}_i \to \exists_{\text{-}(\mathbb{A}_i,\vec{x})}\,(\mathbb{D}_j \wedge \mathbb{E}_{ij})$$

where E_{ij} is an evidence of set-inclusion of $\mathbb{A}_i$ into $\mathbb{B}_j$ and with the side condition $\mathrm{fv}(\mathbb{A}_i,\mathbb{C}_i) \cap \mathrm{fv}(\mathbb{B}_j,\mathbb{D}_j) \subseteq \vec{x} \cap \vec{y}$.

The state subsumption *is the least partial order $\sqsupseteq$ closed under alpha-renaming of local variables that satisfies $R \sqsupseteq S$ if any branch of R is covered by some branch of S. The* (set-based) state equivalence *is equivalence induced by the state subsumption, i.e., $R \equiv S$ iff $R \sqsupseteq S$ and $S \sqsupseteq R$.*

In the following $\Sigma_{\equiv}$ will denote the quotient set Σ by $\equiv$ and $[S]$ the equivalence class of the state S by $\equiv$.

Here follows some technical properties of the state equivalence.

Proposition 2. *Let* $(\mathbb{A}|\mathbb{C})$ *and* $(\mathbb{B}|\mathbb{D})$ *be two constrained set of atoms,* Γ *a set of constrained sets of atom, and* $\bar{x}$ *a set of variables. The following properties hold:*

1. *For any renaming* ρ *with domain disjoint from* $\bar{x}$*:*

$$\langle (\mathbb{A}|\mathbb{C})\,;\Gamma\rangle_{\bar{x}} \equiv \langle (\mathbb{A}\rho|\mathbb{C}\rho)\,;\Gamma\rangle_{\bar{x}}$$

2. *For any variable* y *no free in* $\vec{x}$ *and any term* t

$$\langle (\mathbb{A}|y=t,\mathbb{C})\,;\Gamma\rangle_{\bar{x}} \equiv \langle (\mathbb{A}\,[y\backslash t]\,|y=t,\mathbb{C})\,;\Gamma\rangle_{\bar{x}}$$

3. *If* $\mathcal{T} \vDash \exists_{\text{-}(\mathbb{A}=\mathbb{B},\bar{x})}\,(\mathbb{C}) \leftrightarrow \exists_{\text{-}(\mathbb{A}=\mathbb{B},\bar{x})}\,(\mathbb{D})$ *then*

$$\langle (\mathbb{A};\mathbb{C})\,;\Gamma\rangle_{\bar{x}} \equiv \langle (\mathbb{B};\mathbb{D})\,;\Gamma\rangle_{\bar{x}}$$

4. *For any variable* y *not free in* Γ*:* $\langle\Gamma\rangle_{\bar{x}} \equiv \langle\Gamma\rangle_{\bar{x}y}$

5. $\langle (|\top)\,;\Gamma\rangle_{\bar{x}} \equiv \langle (|\top)\rangle_{\bar{x}}$ *and* $\langle (\mathbb{A}|\bot)\,;\Gamma\rangle_{\bar{x}} \equiv \langle\Gamma\rangle_{\bar{x}}$

6. *For all variables* y *and* z *and any term s.t.* $y \notin \mathrm{fv}(\mathbb{A},\mathbb{B},\vec{x})$*:*

$$\langle (\mathbb{A}\,[y\backslash z]\mid z=t,\mathbb{B})\,;\Gamma\rangle_{\bar{x}} \equiv \langle (\mathbb{A}\,[y\backslash t]\,|z=t\wedge\mathbb{B})\,;\Gamma\rangle_{\bar{x}}$$

Once the state equivalence has been stated, the operational semantics of our language can be defined by a single rule.

Definition 3. *Formally, the* analytical operational semantics *of a program* $\mathcal{P}$ *is the least binary relation* $\xrightarrow{\mathcal{P}}$ *on* $\Sigma_{\equiv}$ *satisfying the following rule:*

$$\frac{\left(r \mathbin{@} \mathbb{H} \Longleftrightarrow \mathbb{G} \mid (\mathbb{B}_i|\mathbb{C}_i)_{i\in I}\right) \in \mathcal{P}\rho}{[\langle (\mathbb{H},\mathbb{A}|\mathbb{G},\mathbb{D})\,;\Gamma\rangle_{\bar{x}}] \xrightarrow{\mathcal{P}} \left[\left\langle (\mathbb{B}_i,\mathbb{A}|\mathbb{G},\mathbb{C}_i,\mathbb{D})_{i\in I}\,;\Gamma\right\rangle_{\bar{x}}\right]}$$

where ρ *renames the local variables of* $\mathcal{P}$ *with fresh variables.* $\xrightarrow{\mathcal{P}}{}^{\equiv}$ *will denoted the reflexive closure of* $\xrightarrow{\mathcal{P}}$ *and* $\xrightarrow{\mathcal{P}}{}^{*}$ *the transitive closure of* $\xrightarrow{\mathcal{P}}{}^{\equiv}$*. For two states* S *and* S' *we may also write* $S \xrightarrow{\mathcal{P}} S'$ *instead of* $[S] \xrightarrow{\mathcal{P}} [S']$*.*

1.4 Monotony

The *monotonicity* is a fundamental property of any CC language. It means that of a transition is possible, then the same transition is possible in any larger state (those state that contain addition information in the form of atoms and constraints). In other word adding information to a state cannot inhibit the applicability of a rule. In this section we defined formally this notion and prove our language features it.

To help reduce the level of verbosity we introduce three operators on states: The *quantification*, the *disjunction*, and the *conjunction* of states are respectively defined as:

$$\exists \bar{z}.\langle \Gamma \rangle_{\bar{x}} \stackrel{\text{def}}{=} \langle \Gamma \rangle_{(\bar{x} \setminus \bar{z})}$$

$$\langle \Delta \rangle_{\bar{x}} \oplus \langle \Gamma \rangle_{\bar{y}} \stackrel{\text{def}}{=} \langle \Delta; \Gamma \rangle_{\bar{x}\bar{y}}$$

$$\langle (\mathbb{A}_i | \mathbb{C}_i)_{i \in I} \rangle_{\bar{x}} \otimes \langle (\mathbb{B}_j | \mathbb{D}_j)_{j \in J} \rangle_{\bar{y}} \stackrel{\text{def}}{=} \langle (\mathbb{A}_i, \mathbb{B}_j | \mathbb{C}_i, \mathbb{D}_j)_{(i,j) \in I \times J} \rangle_{\bar{x}\bar{y}}$$

where $\left(\text{fv}((\mathbb{A}_i, \mathbb{C}_i)_{i \in I}) \cap \text{fv}(\text{fv}((\mathbb{B}_j, \mathbb{D}_j)_{j \in J}\right) \subseteq \bar{x} \cap \bar{y}$ and $\text{fv}(\Delta) \cap \text{fv}(\Gamma) \subseteq \bar{x} \cap \bar{y}$.

It appears the set of states supplied with these operators form a lattice. By a slight abuse of notation we will confuse $\top$ and $\bot$ with the *trivial state* $\langle (\emptyset | \top) \rangle$ and the *inconsistent state* $\langle \emptyset \rangle$ respectively.

Proposition 4. *$\mathcal{S}$ ordered by $\sqsupseteq$ is a lattice, where $\bot$ and $\top$ are respectivelly the suppremum and the infimum of $\mathcal{S}$, and $\otimes$ and $\oplus$ compute respectively the suppremum and the infimum of two elements.*

Proof. The reflexivity and the transitivity of $\sqsupseteq$ follows directly by reflexivity and the transitivity of the logical implication. We know hence that $\sqsupseteq$ is a partial order relation. Now we have to prove that $(S \otimes S')$ and $(S \oplus S')$ are respectively an infimum and a supremun for S and S'. In other words we need to establish (1) $((S \otimes S') \sqsupseteq S)$, (2) $((S \otimes S') \sqsupseteq S')$, (3) $(S \sqsupseteq R \,\&\, S \sqsupseteq R')$ implies $(S \sqsupseteq (R \otimes R'))$, (4) $(S \sqsupseteq (S \oplus S'))$, (5) $(S' \sqsupseteq (S \oplus S'))$, and (6) $(S \sqsupseteq R \,\&\, S' \sqsupseteq R)$ implies $((S \oplus S') \sqsupseteq R)$. For (1) and (2) it is sufficient to notice that $\vDash (\mathbb{C} \wedge \mathbb{C}') \rightarrow \exists \bar{x} \left(\mathbb{C} \wedge \left(\mathbb{A} \,\dot{\subseteq}\, (\mathbb{A}, \mathbb{A}')\right)\right)$. For (3) note that if $\mathcal{T} \vDash \mathbb{D} \rightarrow \exists \vec{y}(\mathbb{C} \wedge (\mathbb{A} \,\dot{\subseteq}\, \mathbb{B}))$ and $\mathcal{T} \vDash \mathbb{D}' \rightarrow \exists \vec{y}(\mathbb{C} \wedge (\mathbb{A} \,\dot{\subseteq}\, \mathbb{B}'))$, then $\mathcal{T} \vDash \mathbb{D} \wedge \mathbb{D}' \rightarrow \exists \vec{y}(\mathbb{C} \wedge (\mathbb{A} \,\dot{\subseteq}\, (\mathbb{B}, \mathbb{B}')))$. (5), (6), and (7) follow directly by definition of $\sqsupseteq$. To conclude just notice that for any state S, $\bot \sqsupseteq S \sqsupseteq \top$ hold. □ □

We will say that a binary relation $\mathcal{R}$ is *monotone* if it is stable with respect to disjunction, conjunction, and quantification. Formally stability is defined as:

- $\mathcal{R}$ is $\oplus$-*stable*. if for all $S, S' \in \Sigma$ $S \,\mathcal{R}\, S'$ implies $(S \oplus R)\, \mathcal{R}^*\, (S' \oplus R)$ for any state R.
- $\mathcal{R}$ is $\otimes$-*stable*, if for all $S, S' \in \Sigma$ $S \,\mathcal{R}\, S'$ implies $(S \otimes R)\, \mathcal{R}^*\, (S' \otimes R)$ for any R.
- $\mathcal{R}$ is $\exists$-*stable*, if for all $S, S' \in \Sigma$ $S \,\mathcal{R}\, S'$ implies $(\exists \vec{x}.S)\, \mathcal{R}^*\, (\exists \vec{x}.S')$ for any set of variable $\vec{x}$.

where $\mathcal{R}^*$ is the relfexive-transitive closure of $\mathcal{R}$.

Unfortunalty, the transition $\xrightarrow{\mathcal{P}}$ is not strictly monotone. It fact a transition in a state may correspond to several transitions in the augmented states. That comes form the fact the operational semantics is defined only on states in disjunctive normal form. Nevertheless all is not lost. Indeed the several transitions need in the bigger state apply allways to different branches. We can therefore defined a notion of transition that allows the application of a rule to each different branches in one step. As shown by Proposition 6, this notion we call parallel transition has the good property to be monotone.

Example 5. *Consider the program $\mathcal{P}$ built form the unique rule $(r \mathbin{@} a \iff b)$. Clearly $\langle (a) \rangle \xrightarrow{\mathcal{P}} \langle (b) \rangle$. Monotony would implies that for any state S, $(\langle (a) \rangle \wedge S) \xrightarrow{\mathcal{P}} (\langle (b) \rangle \wedge S)$. It is easy to very the property does not hold for $S \equiv \langle (c) \,;(d) \rangle$. Indeed $(\langle (a) \rangle \wedge \langle (c) \,;(d) \rangle) \equiv (\langle (a,c) \,;(a,d) \rangle) \not\xrightarrow{\mathcal{P}} (\langle (b,c) \,;(b,d) \rangle) \equiv (\langle (b) \rangle \wedge \langle (c) \,;(d) \rangle)$. In fact, a number of steps is equivalent to the number of branches in S (2 for our counter examples) is necessary.*

We can however recover a weak form of monotony by allowing parallel rewritting in distinct branches.

The *(disjunctive) parallel closure* of given binary relation $\mathcal{R}$ over states is the least relation $\mathcal{R}^{\oplus}$ satisfying the following rule:

$$\frac{S_1 \,\mathcal{R}\, S'_1 \qquad \cdots \qquad S_n \,\mathcal{R}\, S'_n}{\exists \bar{x}\, (S \oplus S_1 \oplus \cdots \oplus S_n) \,\mathcal{R}^{\oplus}\, \exists \bar{x}\, (S \oplus S'_1 \oplus \cdots \oplus S'_n)}$$

Proposition 6 (Weak Monotonicity)**.** *The relation $\xrightarrow{\mathcal{P}}{}^{\oplus}$ is monotone.*

The proof of the proposition used the two following lemmas.

Lemma 7. *$\sqsupseteq$ and $\equiv$ are monotone relations over states.*

Proof. The $\oplus$-stability and the $\otimes$-stability of $\sqsupseteq$ are immediate since $\mathcal{S}$ is a lattice. For the $\exists$-stability of $\sqsupseteq$, just notice that if $\mathcal{T} \vDash \phi \to \exists_{\text{-}\vec{x}} \phi$ then for any set $\bar{y}$ bigger than $\bar{x}$ $\mathcal{T} \vDash \phi \to \exists_{\text{-}\vec{y}} \phi$. Hence we established the monotonicity of $\sqsupseteq$. The monotonicity of $\equiv$ is an immediate corollary of the monotonicity of $\sqsupseteq$ □ □

Lemma 8. *If $S \xrightarrow{\mathcal{P}} S'$ then $(R \otimes S) \xrightarrow{\mathcal{P}}{}^{\oplus} (R \otimes S)$.*

Proof. Let assume the rule applied is $(r \mathbin{@} \mathbb{H} \iff \mathbb{G} \mid (\mathbb{B}_i | \mathbb{C}_i))$. By definition of $\xrightarrow{\mathcal{P}}$, we know that S and S' are of the forms

- $S \equiv \langle (\mathbb{H}, \mathbb{A} | \mathbb{G}, \mathbb{D}) \,; \Gamma \rangle_{\bar{x}}$
- $S' \equiv \left\langle (\mathbb{B}_i, \mathbb{A} | \mathbb{C}_i \mathbb{G}, \mathbb{D})_{i \in I} \,; \Gamma \right\rangle_{\bar{x}}$

We prove the case where R has a single branch.

By induction on the number of branches in R. The base case is immediate (R has no branch), the result is immediate since $\langle (\emptyset | \bot) \rangle \equiv R \equiv R \otimes S \equiv R \otimes S'$. For the inductive case (R has $n+1$ branches).

□

2 Limit Semantics

2.1 Minimal Operation Semantics

This section presents a restriction operational semantics, the so-called *minimal semantics*, which under-approximate the general semantics in the sens of $\sqsupseteq$. Those minimal operation

semantics prevent the rules from consuming any atoms in a branch: the result of each rule's application is stored in a new branch, while the original branches are kept intact.

Definition 9. *The* minimal (operational) semantics *of a program $\mathcal{P}$ are given by the least relation $\xrightarrow{\mathcal{P}}_{\curlywedge}$ on $\Sigma_{\equiv}$ satisfying the following rule:*

$$\frac{\left(r \mathbin{@} \mathbb{H} \iff \mathbb{G} \mid (\mathbb{B}_i|\mathbb{C}_i)_{i\in I}\right) \in \mathcal{P}\rho}{\left[\langle(\mathbb{H},\mathbb{A}|\mathbb{G},\mathbb{D});\Gamma\rangle_{\bar{x}}\right] \xrightarrow{\mathcal{P}}_{\curlywedge} \left[\left\langle(\mathbb{B}_i,\mathbb{A}|\mathbb{G},\mathbb{C}_i,\mathbb{D})_{i\in I}\,;(\mathbb{H},\mathbb{A}|\mathbb{G},\mathbb{D});\Gamma\right\rangle_{\bar{x}}\right]}$$

where ρ renames the local variables of $\mathcal{P}$ with fresh variables. We will use $\xrightarrow{\mathcal{P}}_{\curlywedge}{}^{\equiv}$ to denote its reflexive closure and $\xrightarrow{\mathcal{P}}_{\curlywedge}{}^{*}$ its reflexive closure. For two states S and S' we may also write $S \xrightarrow{\mathcal{P}}_{\curlywedge} S'$ instead of $[S] \xrightarrow{\mathcal{P}}_{\curlywedge} [S']$.

The minimal semantics feature the property of (strong) monotonicity.

Proposition 10 (Monotonicity)**.** *For any program $\mathcal{P}$ containing a rule r, $\xrightarrow{\mathcal{P}}_{\curlywedge r}$ is monotone.*

The following proposition justifies the name of the semantics. It basically states that the minimal semantics underapproximate (in the sense of state subsumption) the general semantics.

Proposition 11. *Let $\mathcal{P}$ be an arbitrary program and S, S', and R be three states.*

- *(Soundness) If $S \xrightarrow{\mathcal{P}}_{\curlywedge} S'$ then $S \xrightarrow{\mathcal{P}}_{\rightarrow} S'$.*
- *(Completness) If $R \sqsupseteq R' \xrightarrow{\mathcal{P}}_{\rightarrow} S$, then there exists R', such that $R \xrightarrow{\mathcal{P}}_{\curlywedge}{}^{*} S' \sqsubseteq S'$.*

2.2 Maximal Operation Semantics

This section presents a restriction operational semantics, the so-called *maximal semantics*, which over-approximate the general semantics in the sens of $\sqsupseteq$.

The so-called *maximal semantics* prevent rules from consume their heads. Furthermore, they never create more branches than the body of a rule imposes. We argue that this captures the pure forward chaining transitions of the general semantics.

Definition 12. *The* maximal (operational) semantics *of a program $\mathcal{P}$ are given by the least relation $\xrightarrow{\mathcal{P}}_{\searrow}$ on $\Sigma_{\equiv}$ satisfying the following rule:*

$$\frac{\left(r \mathbin{@} \mathbb{H} \iff \mathbb{G} \mid (\mathbb{B}_i|\mathbb{C}_i)_{i\in I}\right) \in \mathcal{P}\rho}{\left[\langle(\mathbb{H},\mathbb{A}|\mathbb{G},\mathbb{D});\Gamma\rangle_{\bar{x}}\right] \xrightarrow{\mathcal{P}}_{\searrow} \left[\left\langle(\mathbb{H},\mathbb{B}_i,\mathbb{A}|\mathbb{G},\mathbb{C}_i,\mathbb{D})_{i\in I}\,;\Gamma\right\rangle_{\bar{x}}\right]}$$

where ρ renames the local variables of $\mathcal{P}$ with fresh variables. We will use $\xrightarrow{\mathcal{P}}_{\searrow}{}^{\equiv}$ to denote its reflexive closure and $\xrightarrow{\mathcal{P}}_{\searrow}{}^{*}$ its reflexive-transitive closure. For two states S and S' we may also write $S \xrightarrow{\mathcal{P}}_{\searrow} S'$ instead of $[S] \xrightarrow{\mathcal{P}}_{\searrow} [S']$.

As the general operational semantics, the maximal one feature a weak form of monotonicity.

Proposition 13 (Weak Monotonicity)**.** *For any program $\mathcal{P}$ containing a rule r, $\xrightarrow{\mathcal{P}}_{\searrow r}{}^{\oplus}$ is monotone.*

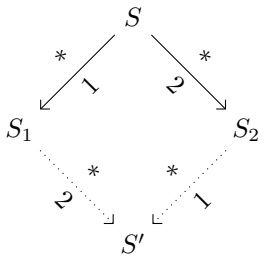

Figure 1: Commutation

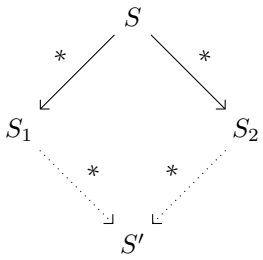

Figure 2: Confluence

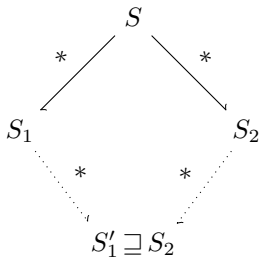

Figure 3: Commutation up to $\sqsupseteq$

The following proposition states that the maximal semantics overapproximate the general semantics.

Proposition 14. *Let $\mathcal{P}$ be a program and S, S', and R be three states.*

- *(Soundness) If $S \xrightarrow{\mathcal{P}}\!\!\!\!\!\searrow S'$ then $S \xrightarrow{\mathcal{P}} S'$.*
- *(Completness) If $R \sqsupseteq R' \xrightarrow{\mathcal{P}} S$ then there exists S' such that $R \xrightarrow{\mathcal{P}}\!\!\!\!\!\searrow^* S' \sqsupseteq S$.*

3 Confluence

Confluence refers to the fact that two finite computations starting from a common state can always be prolonged so as to eventually meet in a common state again. Because it justifies the use of committed choices for the rules application we are interested by defining a proper notion of confluence for programs. For this purpose we will extend the decreasing diagrams criterion on we recently proposes for classical CHR.

From the point of view of abstract rewriting system, two rewrite $\rightarrow_1$ and $\rightarrow_2$ *commute* if $(\leftarrow_2^* \cdot \rightarrow_1^*) \subseteq (\rightarrow_1^* \cdot \leftarrow_2^*)$ holds. A rewrite is confluent if it commutes with itself. (Figures 1 and 2 represent graphically the two notions.) Concretely we will say that a program $\mathcal{P}$ is confluent if so is $\xrightarrow{\mathcal{P}}$.

3.1 Commutation of the limit semantics

Before interesting ourselves in a concrete way to establish confluence of programs with respect to the general semantics, we show that confluence can be reformulate into an interesting notion of commutation with respect to the limit semantics. the intuition about this new notion can be understood has a formal way proving the limit semantics of a program do not contradict each other.

Definition 15 (Commutation of the limit semantics)**.** *The limit semantics of a program $\mathcal{P}$* commute up to $\sqsupseteq$ *if for all states S, R and T such that $S \xrightarrow{\mathcal{P}}\!\!\!\!\!\nearrow^* R$ and $S \xrightarrow{\mathcal{P}}\!\!\!\!\!\searrow^* T$ there exist two states R' and T' such that $R \xrightarrow{\mathcal{P}}\!\!\!\!\!\searrow^* R'$ and $S \xrightarrow{\mathcal{P}}\!\!\!\!\!\nearrow^* T'$ and $R' \sqsupseteq T'$. The definition is graphically represented at Figure 3.*

The following theorem states that the commutation of the limit semantics of a program $\mathcal{P}$ is equivalent to the general confluence $\mathcal{P}$.

Theorem 16. *Let $\mathcal{P}$ be CHR$^\vee$ program. The following properties are equivalent:*

(i) *$\mathcal{P}$ is confluent.*

(ii) *The limits semantics of a program $\mathcal{P}$ commute up to $\sqsupseteq$.*

(iii) *For any peak $R \xleftarrow{\mathcal{P}}{}^* S\ (\xrightarrow{\mathcal{P}} \cdot \sqsupseteq)^*\ T$ there is a valley $R\ (\xrightarrow{\mathcal{P}} \cdot \sqsupseteq)^*\ S' \xleftarrow{\mathcal{P}}{}^* T$.*

The proof of the theorem use the following lemma. This latter states that $\xrightarrow{\mathcal{P}}_r S$ can be permuted with $\sqsupseteq$ and $\sqsubseteq$.

Lemma 17. *Let $\mathcal{P}$ be a CHR$^\vee$ program. For all states R, S, and T the following properties holds:*

(i) *If $R \xrightarrow{\mathcal{P}} S \sqsupseteq T$ then there exists S' s.t $R \sqsupseteq S' \xrightarrow{\mathcal{P}}{}^{\oplus} T$.*

(ii) *If $R \sqsupseteq S \xrightarrow{\mathcal{P}} T$ then there exists S' s.t. $R \xrightarrow{\mathcal{P}}_r S' \sqsupseteq T$.*

(iii) *If $R \xrightarrow{\mathcal{P}} S \sqsubseteq T$ then there exists S' s.t. $R \sqsubseteq S' \xrightarrow{\mathcal{P}} T$.*

(iv) *If $R \sqsubseteq S \xrightarrow{\mathcal{P}} T$ then there exists S' s.t. $R \xrightarrow{\mathcal{P}}{}^{\oplus} S' \sqsubseteq T$.*

Proof. Using the monotony of $\xrightarrow{\mathcal{P}}$ and the fact that $\mathcal{S}$ is a lattice we have: For case (i), $R \sqsupseteq (R \otimes T) \xrightarrow{\mathcal{P}}{}_r^{\oplus} (S \otimes T) \equiv T$. For case (ii), $R \equiv (R \oplus S) \xrightarrow{\mathcal{P}}{}_r (R \oplus T) \sqsupseteq T$. For case (iii), $R \sqsubseteq (R \oplus T) \xrightarrow{\mathcal{P}}_r (S \oplus T) \equiv T$. Finally for case (iv), $R \equiv (R \otimes S) \xrightarrow{\mathcal{P}}_r^{\oplus} (R \otimes T) \sqsubseteq T$. □

Proof of Theorem 16.

- *(i) $\Longrightarrow$ (ii).*
  If $R \xleftarrow{\mathcal{P}}{}^* S \xrightarrow{\mathcal{P}}{}^* T$.
  Then $R \xleftarrow{\mathcal{P}}{}^* S \xrightarrow{\mathcal{P}}{}^* T$ (by soundness of $\xrightarrow{\mathcal{P}}$ and $\xrightarrow{\mathcal{P}}$).
  Then $R \xrightarrow{\mathcal{P}}{}^* \cdot \xleftarrow{\mathcal{P}}{}^* T$ (by hypothesis).
  Then $R \xrightarrow{\mathcal{P}}{}^* \cdot \sqsupseteq \cdot \xleftarrow{\mathcal{P}}{}^* T$ (by completeness of $\xrightarrow{\mathcal{P}}$ and $\xrightarrow{\mathcal{P}}$).

- *(ii) $\Longrightarrow$ (iii).*
  If $R \xleftarrow{\mathcal{P}}{}^* S\ (\xrightarrow{\mathcal{P}} \cdot \sqsupseteq)^*\ T$
  Then $R \xleftarrow{\mathcal{P}}{}^* S \xrightarrow{\mathcal{P}}{}^* \cdot \sqsupseteq T$ (by Lemma 17).
  Then there exist R' and T' s.t. $R \sqsupseteq R' \xleftarrow{\mathcal{P}}{}^* S \xrightarrow{\mathcal{P}}{}^* T' \sqsupseteq T$
  (by completeness of $\xrightarrow{\mathcal{P}}$ and $\xrightarrow{\mathcal{P}}$).
  Then $R \sqsupseteq R' \xrightarrow{\mathcal{P}}{}^* \cdot \sqsupseteq \cdot \xleftarrow{\mathcal{P}}{}^* T' \sqsupseteq T$ (by hypothesis).
  Then $R \sqsupseteq R' \xrightarrow{\mathcal{P}}{}^* \cdot \sqsupseteq \cdot \xleftarrow{\mathcal{P}}{}^* T' \sqsupseteq T$ (by soundness of $\xrightarrow{\mathcal{P}}$ and $\xrightarrow{\mathcal{P}}$).
  Then $R \xrightarrow{\mathcal{P}}{}^* \cdot \sqsupseteq \cdot \xleftarrow{\mathcal{P}}{}^* T$ (by Lemma 17).
  Then $R\ (\xrightarrow{\mathcal{P}} \cdot \sqsupseteq)^* \cdot \xleftarrow{\mathcal{P}}{}^* T$ (by reflexivity of $\sqsupseteq$).

- *(iii) $\Longrightarrow$ (i).*
  If $R \xleftarrow{\mathcal{P}}{}^* S \xrightarrow{\mathcal{P}}{}^* T$.
  Then by hypothesis there exist two states U and V such that
  $R\ (\xrightarrow{\mathcal{P}} \cdot \sqsupseteq)^*\ U \xleftarrow{\mathcal{P}}{}^* T$ and $R \xrightarrow{\mathcal{P}}{}^* V\ (\sqsubseteq \cdot \xleftarrow{\mathcal{P}})^*\ T$.

Then by Lemma 17 there exist two states U' and V' such that
$R \sqsupseteq U' \xrightarrow{\mathcal{P}}{}^* U \xleftarrow{\mathcal{P}}{}^* T$ and $R \xrightarrow{\mathcal{P}}{}^* V \xleftarrow{\mathcal{P}}{}^* V' \sqsubseteq T$.
Then finally because Σ is a lattice and $\xrightarrow{\mathcal{P}}{}^*$ is monotone,
$R \equiv (R \oplus U') \xrightarrow{\mathcal{P}}{}^* (U \oplus V) \xleftarrow{\mathcal{P}}{}^* (T \oplus V') \equiv T$. □

□

3.2 Multiset-based Confluence vs Set-based Confluence

In this section we show that classical multiset-based confluence [1] requires strictly stronger requirements that the analytical set-based confluence. This implies that multiset-based confluence implies the set-based one. This also provide a simple technique to prove confluence of programs when run under the set-based semantics.

The muliset-based equivalence is the least relation $\equiv_m$ satisfying the rule:

$$\frac{\mathcal{T} \vDash \bigwedge_{i\in I} \left(\left(\mathbb{C}_i \to \exists_{\text{-}(\mathbb{A}_i,\vec{x})} \left(\mathbb{D}_i \wedge \mathbb{A}_i = \mathbb{B}_i\right)\right) \wedge \left(\mathbb{D}_i \to \exists_{\text{-}(\mathbb{B}_i,\vec{y})} \left(\mathbb{C}_i \wedge \mathbb{B}_i = \mathbb{A}_i\right)\right)\right)}{\langle (\mathbb{A}_i|\mathbb{C}_i)_{i\in I}\rangle_{\vec{x}} \equiv_m \langle (\mathbb{B}_i|\mathbb{D}_i)_{i\in I}\rangle_{\vec{y}}}$$

with the side condition $\mathrm{fv}(\mathbb{A}_i, \mathbb{C}_i) \cap \mathrm{fv}(\mathbb{B}_i, \mathbb{D}_i) \subseteq \vec{x} \cap \vec{y}$. In the following Σ_m will denote the quotient set Σ by $\equiv_m$ and $[S]_m$ the equivalence class of the state S by $\equiv_m$.

The *multiset-based operational semantics* of a non-disjunctive program $\mathcal{P}$ is the least binary relation $\overset{\mathcal{P}}{\succ\!\!\rightarrow}$ on $\Sigma_\equiv$ satisfying the following rule:

$$\frac{(r \mathbin{@} \mathbb{H} \iff \mathbb{G} \mid (\mathbb{B}|\mathbb{C})\,;\Gamma) \in \mathcal{P}\rho}{[\langle (\mathbb{H}, \mathbb{A}|\mathbb{G}, \mathbb{D})\,;\Gamma\rangle_{\vec{x}}]_m \overset{\mathcal{P}}{\succ\!\!\rightarrow} [\langle (\mathbb{B}, \mathbb{A}|\mathbb{G}, \mathbb{C}, \mathbb{D})\rangle_{\vec{x}}]_m}$$

where ρ renames the local variables of $\mathcal{P}$ with fresh variables. $\overset{\mathcal{P}}{\succ\!\!\rightarrow}{}^{\equiv}$ will denoted the reflexive closure of $\overset{\mathcal{P}}{\succ\!\!\rightarrow}$ and $\overset{\mathcal{P}}{\succ\!\!\rightarrow}{}^*$ the transitive closure of $\overset{\mathcal{P}}{\succ\!\!\rightarrow}{}^{\equiv}$. For two states S and S' we may also write $S \overset{\mathcal{P}}{\succ\!\!\rightarrow} S'$ instead of $[S]_m \overset{\mathcal{P}}{\succ\!\!\rightarrow} [S']_m$.

Lemma 18. *Let $\mathcal{P}$ be a program confluent with respect to the multiset-based semantics. For any state S, S', S_1 and S_2 and S_2 such that*

$$\mathcal{S}_1 \xleftarrow{\mathcal{P}}{\prec}{}^* S \sqsubseteq S' \overset{\mathcal{P}}{\succ\!\!\rightarrow}{}^* S_2$$

there exist states S_c, S_1', and S_2' such that

$$S_1 \sqsupseteq S_1' \overset{\mathcal{P}}{\succ\!\!\rightarrow}{}^* S_c \xleftarrow{\mathcal{P}}{\prec}{}^* S_2' \sqsubseteq S_2$$

Proof. Let assume

$$S \equiv_m \langle (\mathbb{A}|\mathbb{C})\rangle_{\vec{x}} \overset{\mathcal{P}}{\succ\!\!\rightarrow}{}^* \langle (\mathbb{A}'|\mathbb{C}')\rangle_{\vec{x}} \equiv_m S_1$$
$$S' \equiv_m \langle (\mathbb{B}|\mathbb{D})\rangle_{\vec{x}} \overset{\mathcal{P}}{\succ\!\!\rightarrow}{}^* \langle (\mathbb{B}'|\mathbb{D}')\rangle_{\vec{x}} \equiv_m S_2$$

Since $S \sqsubseteq S'$, $\mathbb{A} = \{a_1, \ldots, a_m\}$, $\mathbb{B} = \{b_1, \ldots, b_n\}$ for some $a_1, \ldots, a_m, b_1, \ldots, b_n$ such that:

$$\mathcal{T} \vDash \mathbb{D} \to \exists_{\text{-}\mathbb{B},\vec{x}} (\mathbb{C} \wedge a_1 = b_{j_1} \wedge \cdots \wedge a_m = b_{j_m}) \qquad \text{with } j_1, \ldots, j_m \subseteq \{1, \ldots, n\}$$

Now let $R = \langle (\mathbb{A}^n | \mathbb{C}) \rangle_{\vec{x}}$ (where $\mathbb{A}^n$ is the repetition n times of $\mathbb{A}$). Then for $S_1' = \left\langle \left(\mathbb{A}'^n | \mathbb{C}'\rho_2, \ldots, \mathbb{C}'\rho_n\right) \right\rangle_{\vec{x}}$ and $S_2' = \langle \mathbb{A}^n \setminus \{a_{i_1}, \ldots, a_{i_n}\}, \mathbb{B}' | \mathbb{C}, \mathbb{D}' \rangle_{\vec{x}}$ (where each ρ_i renames $\text{fv}(\mathbb{A}', \mathbb{C}') \setminus \text{fv}(\mathbb{A}, \mathbb{C})$ with distinct fresh variables), we have:

$$S_1 \sqsubseteq S_1' \xleftarrow{\mathcal{P}}{<}^* R \overset{\mathcal{P}}{>}^* S_2' \sqsubseteq S_2$$

Then by confluence of $\overset{\mathcal{P}}{>}$, there exists S_c such that:

$$S_1 \sqsubseteq S_1' \overset{\mathcal{P}}{>}^* S_c \xleftarrow{\mathcal{P}}{<}^* S_2' \sqsubseteq S_2$$

□

We define binary relations $\overset{\mathcal{P}}{>\!\triangleright}$ and $\overset{\mathcal{P}}{>\!\blacktriangleright}$ as

$$\overset{\mathcal{P}}{>\!\blacktriangleright} = \left(\sqsubseteq \cdot \overset{\mathcal{P}}{>}^* \cdot \sqsubseteq \right) \qquad \overset{\mathcal{P}}{>\!\triangleright} = \left(\sqsupseteq \cdot \overset{\mathcal{P}}{>}^* \cdot \sqsupseteq \right)$$

Lemma 19. $(\xleftarrow{\mathcal{P}}{<}^* \cdot \overset{\mathcal{P}}{>\!\blacktriangleright}^*) \subseteq (\overset{\mathcal{P}}{>\!\blacktriangleright}^* \cdot \xleftarrow{\mathcal{P}}{<}^*)$

Proof. Tanks to Lemma 18 and transitivity of $\sqsupseteq$ we infer easily that

$$(\xleftarrow{\mathcal{P}}{<} \cdot \overset{\mathcal{P}}{>\!\blacktriangleright}) \subseteq (\overset{\mathcal{P}}{>\!\blacktriangleright} \cdot \xleftarrow{\mathcal{P}}{<})$$

We shall conclude using Hindley's Lemma (See Exercise 1.3.6 in Terese's book [4]). □

Lemma 20. *For all states R, S,the following properties holds:*

(i) If $R \xrightarrow{\mathcal{P}}{\rightarrow} S$ then there exists R' and S' such that $R \equiv_s R' \overset{\mathcal{P}}{>} S' \equiv_s S$

(ii) If $R \xrightarrow{\mathcal{P}}{\rightarrow} S$ then there exists R' and S' such that $R \equiv_s R' \overset{\mathcal{P}}{>} S' \equiv_s S$

Proof. We prove the proposition independently:

(i) By def. of $\xrightarrow{\mathcal{P}}{\rightarrow}$ we have:

- $\left(r \mathbin{@} \mathbb{H} \Longleftrightarrow \mathbb{G} \mid (\mathbb{B}_i | \mathbb{C}_i)_{i \in I}\right) \in \mathcal{P}\rho$
- $R \equiv_s \langle (\mathbb{H}, \mathbb{A} | \mathbb{G}, \mathbb{D}); \Gamma \rangle_{\bar{x}}$,
- $S \equiv_s \left\langle (\mathbb{B}_i, \mathbb{A} | \mathbb{G}, \mathbb{C}_i, \mathbb{D})_{i \in I} ; (\mathbb{H}, \mathbb{A} | \mathbb{D}); \Gamma \right\rangle_{\bar{x}}$.

Therefore we have:

$$R \equiv_s \langle (\mathbb{H}, \mathbb{A} | \mathbb{G}, \mathbb{D}); (\mathbb{H}, \mathbb{A} | \mathbb{G}, \mathbb{D}); \Gamma \rangle_{\bar{x}} \overset{\mathcal{P}}{>} \left\langle (\mathbb{B}_i, \mathbb{A} | \mathbb{G}, \mathbb{C}_i, \mathbb{D})_{i \in I} ; (\mathbb{H}, \mathbb{A} | \mathbb{D}); \Gamma \right\rangle_{\bar{x}} \equiv_s S$$

(ii) By def. of $\xrightarrow{\mathcal{P}}$ we have:

- $\big(r \mathbin{@} \mathbb{H} \Longleftrightarrow \mathbb{G} \mid (\mathbb{B}_i|\mathbb{C}_i)_{i\in I}\big) \in \mathcal{P}\rho$
- $R \equiv_s \langle(\mathbb{H}, \mathbb{A}|\mathbb{G}, \mathbb{D}); \Gamma\rangle_{\bar{x}}$
- $S \equiv_s \big\langle(\mathbb{H}, \mathbb{B}_i, \mathbb{A}|\mathbb{G}, \mathbb{C}_i, \mathbb{D})_{i\in I}\,; \Gamma\big\rangle_{\bar{x}}$

Therefore we have:

$$R \equiv_s \langle(\mathbb{H}, \mathbb{H}, \mathbb{A}|\mathbb{G}, \mathbb{D}); \Gamma\rangle_{\bar{x}} \overset{\mathcal{P}}{>\!\!\!\rightarrow} \big\langle(\mathbb{H}, \mathbb{B}_i, \mathbb{A}|\mathbb{G}, \mathbb{C}_i, \mathbb{D})_{i\in I}\,; \Gamma\big\rangle_{\bar{x}} \equiv_s S$$

□

Theorem 21. *If $\mathcal{P}$ is confluent with respect to the multiset semantics, then it is confluent with respect to the set semantics.*

Proof. If $S_1 \overset{\mathcal{P}}{\leftarrow\!\!\!<}{}^* S \xrightarrow{\mathcal{P}}{}^* S_2$,
then $S_1 \overset{\mathcal{P}}{\lhd\!\!\!<}{}^* S \overset{\mathcal{P}}{>\!\!\!\blacktriangleright}{}^* S_2$ (by Lemma 20),
then $S_1 \overset{\mathcal{P}}{>\!\!\!\blacktriangleright}{}^* S' \overset{\mathcal{P}}{\lhd\!\!\!<}{}^* S_2$ (by Lemma 19),
then $S_1 \left(\sqsubseteq \cdot \xrightarrow{\mathcal{P}} \cdot \sqsubseteq\right)^* S' \left(\sqsupseteq \cdot \overset{\mathcal{P}}{\leftarrow} \cdot \sqsupseteq\right)^* S_2$ (by Propositions 11 and 14),
then $S_1 \xrightarrow{\mathcal{P}}{}^* \cdot \sqsubseteq \cdot \overset{\mathcal{P}}{\leftarrow}{}^* S_2$ (by Lemma 17),
then $\mathcal{P}$ is confluent with respect to the set semantics (by Theorem 16). □ □

It is worth noting the opposite does not holds in the sens there is program confluent with respect to the set-based semantics which are not confluent with respect to the multiset-based semantics.

References

1. S. Abdennadher, T. Frühwirth, and H. Meuss. On confluence of Constraint Handling Rules. In *CP*, LNCS 1118:1–15. Springer, 1996.

2. T. Frühwirth. *Constraint Handling Rules*. Cambridge, 2009.

3. R. Haemmerlé. On Combining Backward and Forward Chaining in Constraint Logic Programming. In *PPDP'14*. ACM Press, 2014.

4. Terese. *Term Rewriting Systems*. Cambridge University Press, 2003.