Verification of Constraint Handling Rules using Linear Logic Phase Semantics

Rémy Haemmerlé[1] and Hariolf Betz[2]

[1] CLIP Group, Facultad de Informática, Universidad Politécnica de Madrid, Spain
remy@clip.dia.fi.upm.es
[2] Faculty of Engineering and Computer Sciences, University of Ulm, Germany
hariolf.betz@uni-ulm.de

Abstract. Constraint Handling Rules (CHR) is a declarative concurrent programming language. Like the class of Concurrent Constraint (CC) languages, CHR features a declarative semantics based on Girard's intuitionistic linear logic. The phase semantics of linear logic has been used in the past to prove safety properties for the class of CC languages. In this paper we show that we can adapt this result to prove safety properties for CHR as well.

1 Introduction

Constraint Handling Rules (CHR) is a concurrent committed-choice rule-based programming language introduced in the 1990s by Frühwirth [Frü98]. While it has been originally designed for the design and implementation of constraint solvers, it has come into use as a general-purpose concurrent programming language. Owing to its origins in the tradition of logic and constraint logic programming, CHR features a classical declarative semantics. Recently, Betz and Frühwirth [BF05,Bet07] proposed an alternative declarative semantics based on Girard's Linear Logic (LL) [Gir87].

The class of Concurrent Constraint programming language (CC) was introduced by Saraswat in 1987 [SR90] as an unifying framework for constraint logic programming and concurrent logic programming with a synchronisation mechanism bases on constraint entailment. From the logic programming tradition, the operational aspects of CC programming have been early connected to a logical semantics based on classical logic [SRP91], however Ruet [Rue96] has shown that a precise logical interpretation of these languages requires linear logic (LL). The class of Linear logic Concurrent Concurrent languages (LCC) is a general extension of CC languages where the constraints are based on linear logic instead of classical logic. In addition to the classical constraint programming featured by CC, the LCC class of languages provides state changes.

The phase semantics is the natural provability semantics of linear logic. In the spirit of classical model theory, it associates formulas with values and can thus be considered the most traditional semantics of linear logic. Despite having been named "the less interesting semantics" by Girard himself [Gir95], Fages,

Ruet and Soliman [FRS01] have proposed an original application of the phase semantics to prove safety properties of (L)CC using the links between LL and (L)CC previously introduced by Ruet.

In this paper, we show how the phase semantics of linear logic can be applied to Constraint Handling Rules in a similar way to Fages', Ruet's and Soliman's proposal for (L)CC paradigm. In practice, to prove a safety property of a CHR program, we will exhibit a phase space and an interpretation of the program in which that particular property does hold. This result illustrates the usefulness of both the phase semantics of linear logic and linear logic semantics of CHR.

The paper is structured as follow: First we recall the basics of Constraint Handling Rules (CHR) in Sect. 2 and its LL semantics in Sect. 2.3. In Sect. 3 we present Girard's phase semantics of LL and in Sect. 4 we explain how it can be applied to prove safety properties for CHR Programs. Finally, we apply this method in one introductory and one more advanced example in Sect. 5.

2 Constraint Handling Rules

In CHR, we distinguish two sets of atomic constraints: The set of *built-in constraints* is handled by a predefined constraint handler. It contains at least the constraints $true$ and $false$ as well as equality $=$. The set of *user-defined constraints* is disjoint from the built-in constraints and is handled by a CHR program.

2.1 Syntax

Definition 1 (CHR Program). *A CHR program is a finite sequence of CHR rules, where a CHR rule is either:*

- *a* simplification *rule of the form:*
  H `<=>` $G \mid B$
- *or a* propagation *rule of the form:*
  H `==>` $G \mid B$

where H (the head*) is a non-empty multi-set of user-defined constraints, G (the* guard*) a conjunction of built-in constraints, and B (the* body*) is a multiset of built-in and user-defined constraints.*

The empty guard $true$ can be omitted together with the symbol $\mid$. The notation $name$ @ R gives a name to a CHR rule R. For the sake of simplicity, we assume without loss of generality that a variable appears at most once in the head of a rule. Furthermore, we do not allow nested disjunction in the body of a rule.

Example 1. The following CHR rules [Frü98] define an ordering constraint solver. Note that equality $=$ is as usual a built-in constraint whereas we assume that $=<$ is a user-defined constraint here.

$reflexivity$ @ $\quad X\texttt{=<}Y \;\texttt{<=>}\; X\texttt{=}Y \mid true.$
$antisymmetry$ @ $\quad W\texttt{=<}X, Y\texttt{=<}Z \;\texttt{<=>}\; W\texttt{=}Z, X\texttt{=}Y \mid W\texttt{=}X.$
$transitivity$ @ $\quad W\texttt{=<}X, Y\texttt{=<}Z \;\texttt{==>}\; X\texttt{=}Y \mid W\texttt{=<}Z.$

The first rule eliminates =< constraints where both arguments are equal, the second replaces two symmetric inequality constraints by one equality constraint, and the third adds constraints in such a way as to implement transitive closure.

2.2 Operational Semantics

We introduce the abstract operational semantics of CHR. In this most general variant of the operational semantics, the language is inherently non-deterministic. More restricted variants of the operational semantics guarantee deterministic execution and avoidance of trivial non-determinsm but are less interesting from the theoretical point of view and with respect to concurrency.

Definition 2 (CHR state). *A* CHR state *is a tuple $\langle F, C\rangle$ where F is a multiset of built-in and CHR constraints called* goal store *and C a conjunction of built-in constraints, called* built-in store*.*

We assume without loss of generality that a *constraint theory* is a set of implications called *non-logical axioms* of the form: $\forall(C \supset D)$ where the C and the D are conjunction of built-in constraints.

Definition 3 (Operational Semantics[FA03]). *Given a CHR program $\mathcal{D}$ and a constraint theory CT, the transition relation $\rightarrow$ over states of the operational semantics, is defined inductively as the least relation satisfying the following rules:*

Solve $\quad \langle\{C\} \uplus F, D\rangle \rightarrow \langle F, C \wedge D\rangle$
if C is a built-in constraint
Simplify $\quad \langle G \uplus E, D\rangle \rightarrow \langle B \uplus E, G\texttt{=}H G \wedge \wedge D\rangle$
if (H <=> $C \mid B$) is in P renamed with fresh variables
and $CT \models D \rightarrow \exists(G\texttt{=}H \wedge C)$
Propagate $\quad \langle G \uplus E, D\rangle \rightarrow \langle B \uplus G \uplus E, G\texttt{=}H \wedge G \wedge D\rangle$
if (H ==> $C \mid B$) is in P renamed with fresh variables
and $CT \models D \rightarrow (G\texttt{=}H \wedge C)$ then

Example 2. One possible execution of the program of the previous example 1 is:

$\langle\{\texttt{Z=<X}, \texttt{X=<Y}, \texttt{Y=<Z}\}, true\rangle$
$\langle\{\texttt{X=<Z}, \texttt{Z=<X}, \texttt{X=<Y}, \texttt{Y=<Z}\}, true\rangle$ (**Propagate** `transitivity`)
$\langle\{\texttt{X=Z}, \texttt{X=<Y}, \texttt{Y=<Z}\}, true\rangle$ (**Simplify** `antisymmetry`)
$\langle\{\texttt{X=<Y} \wedge \texttt{Y=<Z}\}, \texttt{X=Z}\rangle$ (**Solve**)
$\langle\{\texttt{X=Y}\}, \texttt{X=Z}\rangle$ (**Simplify** `antisymmetry`)
$\langle\emptyset, \texttt{X=Y} \wedge \texttt{X=Z}\rangle$ (**Solve**)

2.3 Linear Logic Semantics

In this section we recall the result of [BF05]. The CHR rules and CHR states are translated into ILL as presented in the table 2.3.

true	$true^\dagger = \mathbf{1}$
non-logical axioms	$\forall(C \supset D) = !\forall(C^\dagger \multimap D^\dagger)$
built-in constraints	$C^\dagger = !C$
CHR constraint	$C^\dagger = C$
empty multiset	$\emptyset^\dagger = \mathbf{1}$
conjunction	$(C_1 \wedge \cdots \wedge C_n)^\dagger = C_1^\dagger \otimes \cdots \otimes C_n^\dagger$
simplification rules	$(H \iff G \mid B)^\dagger = \forall((G^\dagger \otimes H^\dagger) \multimap \exists \bar{y}.B^\dagger)$
propagation rules	$(H \implies G \mid B)^\dagger = \forall((G^\dagger \otimes H^\dagger) \multimap H^\dagger \otimes \exists \bar{y}.B^\dagger)$
programs	$\{R_1, \ldots, R_n\}^\dagger = \{R_1^\dagger, \ldots, R_n^\dagger\}$
with $\bar{y} = \mathrm{fv}(G, B) \setminus \mathrm{fv}(H)$	

Table 1. Translation of CHR into ILL

In the following, $CT^\dagger$ is the translation of some constraint theory CT using usual Girard's translation of classical logic into linear logic.

Theorem 1 (Soundness [BF05]). *If a CHR state T is derivable from another CHR state S under a program $\mathcal{D}$ and a constraint theory CT then the following holds:*

$$CT^\dagger, P^\dagger \vdash \forall(S^\dagger \multimap T^\dagger)$$

Theorem 2 (Completeness [BF05]). *Let S and T be two CHR states. If two states are such that :*

$$CT^\dagger, P^\dagger \vdash \forall(S^\dagger \multimap T^\dagger)$$

then there exists a state T' derivable from S such as :

$$CT^\dagger \vdash (S^\dagger \multimap T^\dagger)$$

3 Phase Semantics

Phase semantics is the natural provability semantics of linear logic [Gir87]. It has been successfully applied by Fages et al. in order to prove safety properties of LCC programs through the logical semantics of LCC.

Definition 4 (Phase Space). *A phase space $\mathbf{P} = (P, \cdot, 1, \mathcal{F})$ is a commutative monoid $(P, \cdot, 1)$ together with a set $\mathcal{F}$ of subsets of P, whose elements are called* fact*, such that:*

– *$\mathcal{F}$ is closed under arbitrary intersection,*
– *for all $A \subset P$, for all $F \in \mathcal{F}$, $A \multimap F = \{x \in P : \forall a \in A, a \cdot x \in F\}$ is a fact.*

A parametrical fact *A is a total function from $\mathcal{V}$ to $\mathcal{F}$ assigning to each variable x a fact $A(x)$. Any fact can be seen as a constant parametrical fact, and any operation defined on fact: $(A \star B)(x) = A(x) \star B(x)$.*

Given A and B two parametrical facts, we define the following facts:

$$\begin{aligned} A\&B &= A \cap B \\ A \otimes B &= \bigcap \{F \in \mathcal{F} : A \cdot B \subset F\} \\ A \oplus B &= \bigcap \{F \in \mathcal{F} : A \cup B \subset F\} \\ \exists x.A &= \bigcap \left\{F \in \mathcal{F} : (\textstyle\bigcup_{x \in \mathcal{V}} A(x)) \subset F\right\} \\ \forall x.A &= \bigcap \left\{F \in \mathcal{F} : (\textstyle\bigcap_{x \in \mathcal{V}} A(x)) \subset F\right\} \end{aligned}$$

Here are a few notable facts: the greatest fact $\top = P$, the smallest fact $\mathbf{0}$ and $\mathbf{1} = \bigcap \{F \in \mathcal{F} : 1 \in F\}$.

Definition 5 (Enriched Phase Space). *An* enriched phase space *is a phase space $(P, \cdot, 1, \mathcal{F})$ together with a subset $\mathcal{O}$ of $\mathcal{F}$, whose elements are called* open facts*, such that:*

– *$\mathcal{O}$ is closed under arbitrary $\oplus$,*
– *$\mathbf{1}$ is the greatest open fact,*
– *$\mathcal{O}$ is closed under finite $\otimes$,*
– *$\otimes$ is idempotent on $\mathcal{O}$ (if $A \in \mathcal{O}$ then $A \otimes A = A$).*

$!A$ is defined as the greatest open fact contained in A.

Definition 6 (Valuation). *Given an enriched phase space, a* valuation *is a mapping η from atomic ILL formulas to facts such that $\eta(\top) = \top$, $\eta(\mathbf{1}) = \mathbf{1}$ and $\eta(\mathbf{0}) = \mathbf{0}$.*

Definition 7 (Interpretation). *The interpretation $\eta(A)$ of a formula A is defined inductively a follows:*

$$\begin{aligned} \eta(A \otimes B) &= \eta(A) \otimes \eta(B) \\ \eta(A \multimap B) &= \eta(A) \multimap \eta(B) \\ \eta(A\&B) &= \eta(A)\&\eta(B) \\ \eta(A \oplus B) &= \eta(A) \oplus \eta(B) \\ \eta(!A) &= !\eta(A) \\ \eta(\exists x.A) &= \exists x.\eta(A) \\ \eta(A) &= \eta(A) \end{aligned}$$

We extend this interpretation to multi-sets of formulas in the obvious way by considering the coma as $\otimes$, that is to say $\eta(\emptyset) = \mathbf{1}$ and $\eta(A_1, \ldots, A_n) = \eta(A_1) \otimes \cdots \otimes \eta(A_n)$.

This takes us to defining a notion of validity:

Definition 8 (Validity).

$$\mathbf{P}, \eta \models (\Gamma \vdash A) \text{ if and only if } \eta(\Gamma) \subset \eta(A)$$

$$\mathbf{P} \models (\Gamma \vdash A) \text{ if for every valuation } \eta \ \mathbf{P}, \eta \models (\Gamma \vdash A)$$

$$\models (\Gamma \vdash A) \text{ if for every phase space } \mathbf{P} \ \mathbf{P} \models (\Gamma \vdash A)$$

Theorem 3 (Soundness [Gir87,Oka94]).

If there is a sequent calculus proof of $\Gamma \vdash A$ then $\models (\Gamma \vdash A)$.

Theorem 4 (Completeness [Gir87,Oka94]).

If $\models (\Gamma \vdash A)$ then there exists a sequent calculus proof of $\Gamma \vdash A$.

4 Proving Safety Properties

As Fages et al. [FRS01] have done for the (L)CC paradigm, we can use the phase semantics presented above to prove safety properties for CHR. Indeed, by considering CHR under the abstract operational semantics presented in Definition 3 CHR can be viewed as a subset of a Linear CC language.

Due to the soundness theorem 3 with respect to ILL, we know that:

$$\text{If } \forall CT^\dagger, \mathcal{D}^\dagger \vdash \forall(S^T \multimap T^\dagger) \text{ then } \eta(CT^\dagger, \mathcal{D}^\dagger) \subset \eta(\forall(S^T \multimap T^\dagger)).$$

By contrapositive we get that there exists a phase space $\mathbf{P}$ and a valuation η such that

$$\text{If } \eta(CT^\dagger, \mathcal{D}^\dagger) \not\subset \eta(\forall(S^T \multimap T^\dagger)) \text{ then } \forall CT^\dagger, \mathcal{D}^\dagger \not\vdash \forall(S^T \multimap T^\dagger).$$

Finally, by using the contrapositive of the soundness theorem 1 we have

$$\text{If } \forall CT^\dagger, \mathcal{D}^\dagger \not\vdash \forall(S^T \multimap T^\dagger) \text{ then } S \not\rightarrow T.$$

We have hence the following proposition which allows us to reduce a problem of non-existence of a derivation between two CHR states – i.e. a *safety property* – to a problem of existence of a phase space and an interpretation of the program in which a simple inclusion is not possible. As explained in [FRS01], only the completeness of the logical semantics is used in to prove the property. Nonetheless, the soundness theorem gives us the certitude that a such semantical proof of a true property exists.

Proposition 1. *Let CT be a constraint theory and $\mathcal{D}$ a CHR program. To prove a safety property of the form $S \not\rightarrow T$, it is enough to prove there exists a phase space $\mathbf{P}$, a valuation η a substitution σ and a element $a \in \eta(S\sigma^\dagger)$ such that:*

1. *For any non-logical axiom: $\forall(C_1 \wedge \cdots \wedge C_m) \supset (D_1 \ldots D_n)$ of CT, the inclusion $(\eta(!C_1) \otimes \cdots \otimes \eta(!C_m)) \subset (\eta(!D_1) \otimes \cdots \otimes \eta(!D_n))$ holds;*

2. *For any CHR rule* $H_1, \ldots, H_l$ <=> $G_1 \wedge \cdots \wedge G_m \mid B_1, \ldots, B_n$ *of* $\mathcal{D}$ *the inclusion* $(\eta(H_1) \otimes \cdots \otimes \eta(H_l) \otimes \eta(!G_1) \otimes \cdots \otimes \eta(!G_m)) \subset (\eta(B_1^\dagger) \otimes \cdots \otimes \eta(B_n^\dagger)))$ *holds;*
3. $a \notin \eta((T\sigma)^\dagger)$.

Proof. First notice that conditions *1* and *2* imply that $\mathbf{P}, \eta \models CT^\dagger, \mathcal{D}^\dagger$ and then $\mathbf{1} \in \eta(CT^\dagger, \mathcal{D}^\dagger)$. Now let us suppose that that $a \in \eta((S\sigma)^\dagger)$ and $a \notin \eta((T\sigma)^\dagger)$. Hence we infer that $\mathbf{1} \notin \eta((S\sigma)^\dagger) \multimap \eta((T\sigma)^\dagger)$. Therefore $\mathbf{1} \notin \eta(\forall((S)^\dagger) \multimap \eta((T)^\dagger))$ and then $\eta(CT^\dagger, \mathcal{D}^\dagger) \not\subset \forall\eta((S)^\dagger) \multimap \eta((T)^\dagger)$. Using the soundness theorem 3 we infer that $(CT^\dagger, \mathcal{D}^\dagger) \not\vdash \forall\eta((S)^\dagger) \multimap \eta((T)^\dagger)$. Using the soundness theorem 1, we conclude that $S \not\rightarrow T$. □

5 Examples

5.1 The Three Dining Philosophers Problem

In this example, we formulate a CHR program that implements the Dining Philosophers Problem for three philosophers. Subsequently, we use the phase semantics of linear logic to prove that from the canonic initial state, our program will never reach a state in which both philosopher #1 and philosopher #2 are eating at the same time.

i) The Program.

Let $\mathcal{D}$ be the following program defined under the trivial constraint theory CT:

$fork(1), fork(2)$ <=> $eat(1)$
$fork(2), fork(3)$ <=> $eat(2)$
$fork(3), fork(1)$ <=> $eat(3)$
$eat(1)$ <=> $fork(1), fork(2)$
$eat(2)$ <=> $fork(2), fork(3)$
$eat(3)$ <=> $fork(3), fork(4)$

ii) Formulate the Property.
Our goal here is to prove, using Proposition 1, the following safety property:

$$\langle true, fork(1), fork(2), fork(3)\rangle \not\rightarrow \langle C, eat(1), eat(2), H\rangle$$

where H is an arbitrary multiset of constraints.

iii) The Phase Space.
Consider the following structure $\mathbf{P}$:

- the monoid is $\{\mathbb{N}, \cdot, 1\}$
- $\mathcal{F} = \mathcal{D}(\mathbb{N})$ (the set of parts of $\mathbb{N}$)
- $\mathcal{O} = \{\emptyset, \{1\}\}$

For such phase space, any valuation η respects the two conditions: $\eta(\mathbf{1}) = \{1\}$ and $\eta(\top) = \mathbb{N}$.

iv) The Valuation.
We define η as follows :

- $\eta(fork(1)) = \{2\}$
- $\eta(fork(2)) = \{3\}$
- $\eta(fork(3)) = \{5\}$
- $\eta(eat(1)) = \{6\}$
- $\eta(eat(2)) = \{15\}$
- $\eta(eat(3)) = \{10\}$
- $\eta(X = Y) = \begin{cases} \{1\} & \text{if } X = Y \\ \emptyset & \text{otherwise} \end{cases}$

v) Verify the Validity of the Constraint System (condition *1*).

We need to prove now that the constraint system is valid with respect the phase space $\mathbf{P}$ and the valuation η. In this basic example, we can suppose without loss of generality that the constraint system is the trivial one, i.e. in only contains the basic non-logical axioms for equality:

- (reflexivity) $\forall X.(true \supset X < X + 1)$
- (symmetry) $\forall XY.(X = Y \supset Y = X)$
- (transitivity) $\forall XYZ.((X = Y \wedge Y = Z) \supset X = Y)$

Firstly, note than since $\eta(X = Y)$ is an open fact, $\eta(!X = Y) = \eta(X = Y)$. For (reflexivity) note that $\eta(\mathbf{1}) = \eta(X = X) = \{1\}$, for (symmetry) note that obviously $\eta(X = Y) = \eta(Y = X)$. For (transitivity), either X, Y and Z are equal, in which case $\eta(X = Y) \otimes \eta(Y = Z) = \eta(X = Z) = \{1\}$, or at least one of them is different from the others, in which case $\eta(X = Y) \otimes \eta(Y = Z)$ equals the empty set and is therefore trivially included in $\eta(X = Z)$.

vi) Verify the Validity of the Program (condition *2*).

In order to prove the validity of the program we only have to notice that:

- $\eta(eat(1)) = \eta(fork(1) \otimes fork(2)) = \{6\}$
- $\eta(eat(2)) = \eta(fork(2) \otimes fork(3)) = \{15\}$
- $\eta(eat(3)) = \eta(fork(3) \otimes fork(1)) = \{10\}$

vii) Counter-example (condition *3*).

It can now be easily verified that:
$\eta(\langle true, fork(1), fork(2), fork(3)\rangle^{\dagger}) = \eta(fork(1) \otimes fork(2) \otimes fork(3)) = \{30\}$
$\eta(\langle C, eat(1), eat(2), H\rangle^{\dagger}) \subset \eta(eat(1) \otimes eat(2) \otimes \top) = 90 \cdot \mathbb{N}$

We deduce hence that $30 \in \eta(\langle true, fork(1), fork(2), fork(3)\rangle^{\dagger})$ and $30 \notin \eta(\langle C, eat(1), eat(2), H\rangle^{\dagger})$. Therefore we infer that the intended safety property $(\langle true, fork(1), fork(2), fork(3)\rangle \not\rightarrow \langle C, eat(1), eat(2), H\rangle)$ holds.

5.2 The n Dining Philosophers Problem

In this example, we implement the Dining Philosophers Problem for an arbitrary number of philosphers and we show using the phase semantics that the program can never reach a state in which any two philosophers directly neighboring each other are eating at the same time.

i) The Program and the Constraint Systems.

For the sake of simplicity, we add to each CHR constraint an extra argument N for the total number of philosophers. We suppose that CT includes the constraint theory for natural numbers.

eat_0 @ $fork(M, s(M)), fork(0, s(M))$ <=> $eat(0, s(M))$.
$think_0$ @ $eat(0, s(M))$ <=> $fork(M, s(M)), fork(0, s(M))$.
$eat_{s(X)}$ @ $fork(I, N), fork(s(I), N)$ <=> $eat(s(I), N)$.
$think_{s(X)}$ @ $eat(s(I), N)$ <=> $fork(I, N), fork(s(I), N)$.
$base\_case$ @ $put\_fork(0, N)$ <=> $true$
rec @ $put\_fork(s(I), N)$ <=> $fork(I, N), put\_fork(I, N)$.

ii) Reformulate the Property.

We want to prove that two philosophers (among N philosophers) which are seated side by side cannot be eating at the same time. This can be formalized by the two following safety properties (we naturally assume there are at least two philosophers):

- case of the philosophers 0 and N:

$$\forall M.(\langle true, put\_fork(s(s(M)), s(s(M)))\rangle \not\rightarrow \langle C, eat(s(M), s(s(M))), eat(0, s(s(M))), H\rangle)$$

- case of the philosophers I and $I + 1$:

$$\forall M.(\langle true, put\_fork(s(s(M)), s(s(M)))\rangle \not\rightarrow \langle C, eat(I, s(s(M))), eat(s(I), s(s(M))), H\rangle)$$

iii) The Phase Space.

We consider the same structure $\mathbf{P}$ as previously:

- the monoid is $\{\mathbb{N}, \cdot, 1\}$
- $\mathcal{F} = \mathcal{D}(\mathbb{N})$ (the set of parts of $\mathbb{N}$)
- $\mathcal{O} = \{\emptyset, \{1\}\}$

iv) The Valuation.

Let ϕ be an arbitrary bijection between natural numbers and prime numbers. Now, let us define η as:

- $\eta(fork(I, N)) = \{\phi(I)\}$

– $\eta(eat(I,J)) = \begin{cases} \{1\} & \text{if } I = 0 \text{ and } J = 0 \\ \{\phi(M) \cdot \phi(0)\} & \text{if } I = 0 \text{ and } J = s(M) \\ \{\phi(K) \cdot \phi(s(K))\} & \text{if } I = s(K) \end{cases}$

– $\eta(put\_fork(I,N)) = \begin{cases} \left\{ \prod_{j=0}^{K} \phi(j) \right\} & \text{if } I = s(K) \\ \{1\} & \text{if } I = 0 \end{cases}$

– $\eta(I = N) = \begin{cases} \{1\} & \text{if } I = N \\ \emptyset & \text{otherwise} \end{cases}$

v) Verify the Validity of the Constraint System (condition *1*).

We can assume the very simple following axiomatization:

1. $\forall X(true \supset X = X)$
2. $\forall XY(X = Y \supset Y = X)$
3. $\forall XYZ(X = Y \wedge Y = Z \supset Y = X)$
4. $\forall XY.(s(X) = s(Y) \supset X = Y)$

The verification of the six first axioms is quite straightforward.

vi) Verify the Validity of the Program (condition *2*).

– **Validity of the rules** eat_0 **and** $think_0$: Notice that for any M we have

$$\eta(fork(M, s(M))) \otimes \eta(fork(0, s(M))) = \{\phi(M) \cdot \phi(0)\} = \eta(eat(0, s(M))).$$

– **Validity of the rules** eat_n **et** $think_n$: Notice that for any K and any N:

$$\eta(fork(K, N)) \otimes \eta(fork(s(K), N)) = \{\phi(K) \cdot \phi(s(K))\} = \eta(eat(s(I), N)).$$

– **Validity of the rule** *base_case*: Notice that for any for any N ze have:

$$\eta(put\_fork(0, N)) = \{1\} = \eta(\mathbf{1}).$$

– **Validity of the rule** *rec*: The proof is by cases on the first argument $s(I)$ of the *put_fork* constraint:

- $s(I) = s(0)$: in this case notice that for any N:

$$\begin{aligned} \eta(put\_fork(s(0), N)) &= \left\{ \prod_{i=0}^{0} \phi(i) \right\} = \{\phi(0)\} \otimes \{1\} \\ &= \eta(fork(0)) \otimes \eta(put\_fork(0, N)) \end{aligned}$$

- $s(I) = s(s(K))$ for some K: in this case notice that for any N:

$$\begin{aligned} \eta(put\_fork(s(s(K)), N)) &= \left\{ \prod_{i=0}^{i=s(K)} \phi(i) \right\} = \{\phi(s(K))\} \otimes \left\{ \prod_{i=0}^{i=s(K)} \right\} \\ &= \eta(fork(s(s(K)))) \otimes \eta(put\_fork(s(K)) \end{aligned}$$

vii) Counter-example (condition *3*).

We now have to present two counter-examples, one for each safety property. First, we easily verify that (for any constraint multisets C and H):

- $\eta(put\_fork(s(s(M)), s(s(M)))) = \prod_{i=0}^{i=s(M)}$
- $\eta(eat(s(M), s(s(M))), eat(0, s(s(M))), C, H) \subset \phi(0) \cdot \phi(M) \cdot \phi(s(M))^2 \cdot \mathbb{N}$
- $\eta(eat(I, s(s(M))), eat(s(I), s(s(M))), C, H) \subset \phi(I) \cdot \phi(s(I))^2 \cdot \phi(s(s(I))) \cdot \mathbb{N}$

Since $\phi(0)$, $\phi(M)$ and $\phi(s(M))$ are pairwise distinct prime numbers we have $\prod_{i=0}^{i=s(M)} \notin \phi(0) \cdot \phi(M) \cdot \phi(s(M))^2 \cdot \mathbb{N}$. Similarly since $\phi(I)$, $\phi(s(I))$ and $\phi(s(s(I)))$ are pairwise distinct prime numbers we have $\prod_{i=0}^{i=s(M)} \notin \phi(I) \cdot \phi(s(I))^2 \cdot \phi(s(s(I))) \cdot \mathbb{N}$. By Proposition 1, we prove the two safety properties.

6 Conclusion

Relying on the linear logic semantics of CHR, we showed that the method described by Fages, Ruet and Soliman in [FRS01] to verify safety properties of (L)CC programs can be adapted to CHR programs as well. This adaptation is straightforward as from the point of view of its linear logic semantics, CHR can indeed be viewed as a subset of LCC. We have given a detailed explanation of our method, illustrated with two examples.

Our result provides evidence that the linear logic semantics is a useful tool for the analysis and verification of CHR programs.

While a fully automatic application of our method might not be feasible, it should be possible to significantly speed up the process with a semi-automatic system that propagates a given valuation of the facts over a program and checks whether or not this valuation proves a certain property defined by the user. This could be done with a specific finite domain solver implemented in CHR and optimized for our purpose. Such a system could spare the user the tedious and error-prone process of propagating a valuation manually.

For the future, further investigation of the apparently close relationship between CHR and (L)CC as well as the relationship between CHR and algebraic structures such as the phase semantics seems a promising approach and will hopefully produce further useful results with respect to analysis and verification of CHR programs.

Acknowledgments

We are grateful to Sylvain Soliman for the useful discussion about the phase semantics of linear logic.

This work was funded in part by the Madrid Regional Government under the *PROMESAS* project, the Spanish Ministry of Science under the TIN-2005-09207 *MERIT* project, and the IST program of the European Commission, under

the IST-15905 *MOBIUS*, IST-215483 *SCUBE*, and ITEA 06042 (PROFIT FIT-340005-2007-14) ESPASS projects.

Hariolf Betz is being funded by the University of Ulm under LGFG grant #0518.

References

[Bet07] Hariolf Betz. A linear logic semantics for constraint handling rules with disjunction. In *Proceedings of the 4th Workshop on Constraint Handling Rules*, pages 17–31, 2007.

[BF05] Hariolf Betz and Thom W. Frühwirth. A linear-logic semantics for constraint handling rules. In *Proceedings of CP 2005, 11th*, pages 137–151. Springer-Verlag, 2005.

[FA03] T. Frühwirth and S. Abdennadher. *Essentials of Constraint Programming*. Springer-Verlag, February 2003.

[FRS01] François Fages, Paul Ruet, and Sylvain Soliman. Linear concurrent constraint programming: operational and phase semantics. *Information and Computation*, 165(1):14–41, February 2001.

[Frü98] Thom Frühwirth. Theory and practice of constraint handling rules. *Journal of Logic Programming, Special Issue on Constraint Logic Programming*, 37(1-3):95–138, October 1998.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1), 1987.

[Gir95] Jean-Yves Girard. Linear logic: its syntax and semantics. In *Proceedings of the workshop on Advances in linear logic*, pages 1–42. Cambridge University Press, 1995.

[Oka94] Mitsuhiro Okada. Girard's phase semantics and a higher-order cut-elimination proof. Technical report, Institut de Mathématiques de Luminy, 1994.

[Rue96] Paul Ruet. Logical semantics of concurrent constraint programming. In *Proceedings of CP'96, 2nd International Conference on Principles and Practice of Constraint Programming*. Springer-Verlag, 1996.

[SR90] Vijay A. Saraswat and Martin C. Rinard. Concurrent Constraint Programming. In *POPL'90: Proceedings of the 17th ACM Symposium on Principles of Programming Languages*, 1990.

[SRP91] Vijay A. Saraswat, Martin C. Rinard, and Prakash Panangaden. Semantic foundations of concurrent constraint programming. In *POPL'91: Proceedings of the 18th ACM Symposium on Principles of Programming Languages*, 1991.