A New Proof-theoretical Linear Semantics for CHR

Igor Stéphan
Université d'Angers
igor.stephan@univ-angers.fr

Abstract

Constraint handling rules are a committed-choice language consisting of multiple-heads guarded rules that rewrite constraints into simpler ones until they are solved. We propose a new proof-theoretical declarative linear semantics for Constraint Handling Rules. We demonstrate completeness and soundness of our semantics w.r.t. operational ω_t semantics. We propose also a translation from this semantics to linear logic.

2012 ACM Subject Classification Theory of computation - Logic - Constraint and logic programming

Keywords and phrases Constraint Handling Rules, Linear Logic

Digital Object Identifier 10.4230/OASIcs...

1 Introduction

CHR (for *constraint handling rules*) [9, 10, 11, 12, 13, 14] are a committed-choice language consisting of multiple-heads guarded rules that rewrite constraints into simpler ones until they are solved. CHR are a special-purpose language concerned with defining declarative constraints in the sense of *Constraint logic programming* [16, 17, 18]. CHR are a language extension that allows to introduce *user-defined* constraints, i.e. first-order predicates, as for example less-than-or-equal ($\leq$), into a given host language as Prolog, Lisp, Java or C/C++. CHR define *simplification* of user-defined constraints, which replaces constraints by simpler ones while preserving logical equivalence. For example the anti-symmetry of less-than-or-equal constraint: $((X \leq Y), (Y \leq X) \Leftrightarrow (X = Y))$ where "$(X \leq Y), (Y \leq X)$" is the multiple head of the rule, X, Y are variables and "," denotes conjunction. This rule means "if constraints $(X \leq Y)$ and $(Y \leq X)$ are present then equality $(X = Y)$ is enforced and constraints are solved". CHR define also *propagation* over user-defined constraints that adds new constraints, which are logically redundant but may cause further simplifications. For example the transitivity of less-than-or-equal constraint: $((X \leq Y), (Y \leq Z) \Rightarrow (X \leq Z))$. This rule means "if constraints $(X \leq Y)$ and $(Y \leq Z)$ are present then constraint $(X \leq Z)$ is logically equivalent". CHR allow to use guards, which are sequences of host language statements. For example the reflexivity of less-than-or-equal constraint: $((X \leq Y) \Leftrightarrow (X = Y) \mid \mathit{true})$ where $(X = Y)$ is a test and *true* is a reserved symbol that has for operational semantics "add nothing". This rule means "if constraint $(X \leq Y)$ is present and $(X = Y)$ is true then constraint $(X \leq Y)$ is solved". CHR finally define *simpagation* over user-defined constraints that mixes and subsumes simplification and propagation. The general schema of CHR (simpagation) rules is then $(K_1, \ldots, K_n \backslash D_1, \ldots, D_m \Rightarrow \mathit{guard} \mid G)$ with $n + m \neq 0$ and $G = B_1, \ldots, B_p$ or $G = \mathit{true}$. Constraints $K_1, \ldots, K_n$ are kept like in propagation and constraints $D_1, \ldots, D_m$ are deleted like in simplification. If $n = 0$, a simpagation rule is a simplification rule, and if $m = 0$, a simpagation rule is a propagation rule. For example, the idempotency of less-than-or-equal

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

constraint: $((X \leq Y)\backslash(X \leq Y) \Leftrightarrow true)$. This rule means “if constraint $(X \leq Y)$ is present twice, only one occurrence is kept”. This last example suggests that CHR is more about consumption than truth. CHR rules are applied on multi-sets of constraints. Repeated application of those rules on a multi-set of initial constraints incrementally solves these constraints. The committed-choice principle expresses a *don't care* nondeterminism, which leads to efficient implementations.

From the very beginning, [9, 10] gives a declarative semantics in terms of first-order classical logic: simplification rules are considered as logical equivalences and propagation rules as implications (with an equivalence-based semantics ω_e [19]). But [10] gives also a first *abstract* (or *high-order* or *theoretical*) operational semantics ω_t based on a transition system over sets (with some extensions to avoid the trivial nontermination of propagation rules [1]). The *refined* operational semantics ω_r [8] is finer than the previous one w.r.t. to the classical implementations of CHR. Those operational semantics are in fact ad-hoc linear semantics [6]. In [5, 6, 4] two different proof-theoretical *intuitionistic* linear semantics for CHR are proposed based on (intuitionist) Linear Logic [15]. Those linear semantics for CHR have been extended to CHR$^\vee$[2] which introduces the *don't know* nondeterminism[1] in CHR [7].

As emphasized in [6], "Many implemented algorithms do not have a first-order classical logical reading, especially when these algorithms are deliberately non-confluent", i.e. the committed-choice matters. Moreover "Considering arbitrary derivation from a given goal, termination (and confluence) under the abstract semantics ω_t are preserved under the refined semantics ω_r, but not the other way around. While it fixes the constraint and rule order for execution, the refined operational semantics is still *nondeterministic*" [14]. But if anyone wants, for example, to compile another high level language to CHR paradigm there must be only two sources of nondeterminism: the *don't care* nondeterminism of the committed-choice and the *don't know* nondeterminism of the disjunction of CHR$^\vee$ and no other hidden nondeterminism not controllable by the programmer. But in the already defined semantics of the literature and the current implementations, there is a third source of nondeterminism due to the management of the constraints as an *unordered multi-set*: the order in which the constraints are reactivated by the wake-up-policy function[2] is left unspecified (page 68 of [14]). And there is even a forth source of nondeterminism due to the management of the multiple heads of the simpagation rules. The matching order in the application of a simpagation rule is not deterministic and we do not know which constraints from the multi-set may be chosen and kept or deleted, if more than one possibility exists (page 69 of [14]). Consider the following first-order CHR program with only one rule, which illustrates the first hidden nondeterminism:

$$(a(X), a(Y), s \Leftrightarrow true)$$

and $\{a(1), a(2), a(3), s\}$ as the store (an unordered multi-set) of constraints. The final state may be $\{a(1)\}$, $\{a(2)\}$ or $\{a(3)\}$. Even with the refined ω_r semantics, the semantics of the CHR program rests unknown.

We propose in this article a new proof-theoretical linear semantics for CHR by means of a sequent calculus system in which the store is managed as a multi-set as in the ω_t semantics. This system is proved to be sound and complete w.r.t. the ω_t semantics. We

[1] freely offered when the host language is Prolog
[2] With first-order constraints, instantiation of some variables of the constraints makes them eligible to the application of CHR rules.

propose also a second new proof-theoretical linear semantics for CHR by means of a sequent calculus system in which the store is managed as a *sequence*. This system is proved to be sound. But, more important, this system is completely deterministic and overcomes the two sources of hidden nondeterminism defined above. Finally, we propose for those two systems a translation into the Linear Logic and we prove the soundness of this translation.

Section 2 presents the needed background on Linear Logic (Subsection 2.1) and CHR syntax and semantics (Subsection 2.2). Section 3 presents our two new linear sequent calculi for CHR, the ω_l sequent calculus system in which the store is managed as a multi-set and the $\omega_l^{\otimes}$ sequent calculus system in which the store is managed as a sequence (Subsection 3.1). Those systems are then translated into the Linear Logic and we prove the soundness of this translation (Subsection 3.2). We conclude by a discussion about the possible links to *focusing proofs* of [3] and on some remarks about our two new proof-theoretical semantics for CHR.

2 Background

2.1 Linear logic

Linear Logic is a substructural logical formalism introduced in [15]. It is based on *tokens* which are built on predicate symbols and terms in the usual first-order manner. These tokens (w.r.t. atoms of classical first order logic) represent resources (w.r.t. truth). Linear Logic consumes and produces resources and is aware of their multiplicities. The linear-logic sequent calculus is based on the *sequent*, which is a pair of multi-sets of linear-logic formulas. Linear formulas are built on tokens and the following operators (we only present the useful ones for us): The symbol $\otimes$ stands for the multiplicative conjunction and is similar to conjunction of classical logic. The $\mathbf{1}$ symbol stands for the neutral of $\otimes$ and represents empty resource and corresponds to the *true* of classical logic. The symbol $\&$ stands for the additive conjunction. $a\&b$ represents an internal choice between a and b, it means that one can freely choose between a and b but not have a and b at the same time. The symbol $\multimap$ stands for the linear implication and apply *modus ponens* but by consuming the preconditions. The symbol $\mathbf{0}$ corresponds to the *false* of classical logic. The modality symbol $!$ marks the unlimited resources. The symbol $\exists$ (resp. $\forall$) stands for existential (resp. universal) first-order quantifications.

In what follows we only use the fragment of the linear-logic sequent calculus that is relevant for us in its two-sided version (F, F_1, F_2 and L some linear formulas, Γ, Γ_1, Γ_2, Δ, Δ_1, Δ_2 some multi-sets of formulas).

- Identity rules

$$\frac{}{F \vdash F}\ I \qquad\qquad \frac{\Gamma_1 \vdash L \qquad L, \Gamma_2 \vdash \Delta}{\Gamma_1, \Gamma_2 \vdash \Delta}\ Cut$$

- Multiplicative rules

$$\frac{\Gamma, F_1, F_2 \vdash \Delta}{\Gamma, F_1 \otimes F_2 \vdash \Delta}\ \otimes L \qquad\qquad \frac{\Gamma \vdash \Delta}{\mathbf{1}, \Gamma \vdash \Delta}\ \mathbf{1}L$$

$$\frac{\Gamma_1 \vdash \Delta_1, F_1 \qquad \Gamma_2 \vdash \Delta_2, F_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, F_1 \otimes F_2}\ \otimes R \qquad\qquad \frac{\Gamma_1 \vdash F_1, \Delta_1 \qquad \Gamma_2, F_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2, F_1 \multimap F_2 \vdash \Delta_1, \Delta_2}\ \multimap L$$

- Additive rules

$$\frac{\Gamma, F_1 \vdash \Delta}{\Gamma, F_1 \& F_2 \vdash \Delta}\ \& L_1 \qquad\qquad \frac{\Gamma, F_2 \vdash \Delta}{\Gamma, F_1 \& F_2 \vdash \Delta}\ \& L_2$$

- Quantifier rules (t is a term)

$$\frac{\Gamma, [x \leftarrow t](F) \ \vdash \ \Delta}{\Gamma, (\forall x\ F) \ \vdash \ \Delta}\ \forall L \qquad \frac{\Gamma, [x \leftarrow y](F) \ \vdash \ \Delta}{\Gamma, (\exists x\ F) \ \vdash \ \Delta}\ \exists L$$

The usual proviso for the ∃L rule is assumed: the variable y must not be free in the formulas of the sequent conclusion of the inference rule.

- Exponential rules

$$\frac{\Gamma, !F, !F \ \vdash \ \Delta}{\Gamma, !F \ \vdash \ \Delta}\ !C \qquad \frac{\Gamma, F \ \vdash \ \Delta}{\Gamma, !F \ \vdash \ \Delta}\ !D \qquad \frac{\Gamma \ \vdash \ \Delta}{\Gamma, !F \ \vdash \ \Delta}\ !W$$

A proof tree is a finite labeled tree whose nodes are labeled with sequents such that every sequent node is the consequence of its direct children according to one of the inference rules of the calculus. A proof tree is a *linear proof* if all its leaves are axioms (i.e. instances of the Identity rule I).

2.2 CHR language and its semantics

In this article, we consider a first-order CHR program as an intensional version of the grounded corresponding propositional program with respect to its Herbrand universe based on the function and constant symbols of the program. A constraint is a predicate symbol with elements of the Herbrand universe as arguments. With this point of view, we omit the guard and there is no need of equivalence relation between variables. Moreover, there is no need for a wake up rule since there is no more variable to be woken up in the store of constraints.

2.2.1 The syntax.

The CHR formalism is defined as follows : a CHR rule is a rule of the form ($K_1, \ldots, K_m$, $D_1, \ldots, D_n, B_1, \ldots, B_p$ some constraints):

- [Simpagation rule] $(K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow B)$ with $n > 0$, $m > 0$ or
- [Propagation rule] $(K_1, \ldots, K_m \Rightarrow B)$ with $m > 0$ or
- [Simplification rule] $(D_1, \ldots, D_n \Leftrightarrow B)$ with $n > 0$

and $B = B_1, \ldots, B_p$ with $p > 0$ or *true* or *false* (two reserved symbols).

2.2.2 The operational ω_t semantics.

An *identified* constraint $A\#i$ is a constraint A with some unique integer i, its identity. Function *const*, resp. *id*, gets from an identified constraint its constraint, resp. identity: $const(A\#i) = A$, resp. $id(A\#i) = i$. The *id* function and *const* are extended to sequences, sets and multi-sets of identified constraints in the obvious manner. An *execution state* is a tuple $\langle \Omega, S, H \rangle_c$ where Ω (the current *goal*) stands for a multi-set of constraints to be executed, S (the current *store*) stands for a multi-set of identified constraints, H (the current *propagation history*) stands for a set of words, each recording the name of a rule and identities of identified constraints, c stands for a counter that represents the next free integer which can be used to number an identified constraint. For an initial goal Ω, the initial state is $\langle \Omega, \emptyset, \emptyset \rangle_1$. The operational semantics ω_t is based on the following two transitions, which map a state to an other state (symbol $\uplus$ stands for union of multi-sets):

- *[Introduce]* $\langle \{A\} \uplus \Omega, S, H \rangle_c \rightsquigarrow_t \langle \Omega, \{A\#c\} \uplus S, H \rangle_{c+1}$
- *[Apply]* $\langle \Omega, K_\# \uplus D_\# \uplus S, H \rangle_c \rightsquigarrow_t \langle B \uplus \Omega, K_\# \uplus S, H \uplus \{r.i_1 \ldots i_m\} \rangle_c$ where there exists a simpagation rule $r@(K \backslash D \Leftrightarrow B)$ such that $K_\# = \{K_1\#i_1, \ldots, K_m\#i_m\}$ and $D_\# = \{D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}\}$ and $K_1, \ldots, K_m = K$ and $D_1, \ldots, D_n = D$ and $r.i_1 \ldots i_m \notin H$ ($r.i_1 \ldots i_m$ is the *identity* of the instantiated rule and r is a name for the rule).

The [*Introduce*] transition transports a constraint from the goal to the store and associates an identity to this constraint. A CHR rule $(K\backslash D \Leftrightarrow B)$ is *applicable* if the head of the rule (considered as a multi-set) $K \uplus D$ is a subset of the multi-set $const(S)$ of the constraints of the store S. If a CHR rule $(K\backslash D \Leftrightarrow B)$ is applicable then the CHR rule is *applied*: [*Apply*] transition removes identified constraints $D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}$ from the store and adds the constraints of B to the goal. If $B = true$ nothing is added to the goal. This can only be done if the CHR rules has not already been fired with the same identity in order to forbid trivial loops. In the [*Apply*] transition, if $B = false$ there is no transition at all. The transitions are non-deterministically applied until either no more transition is applicable (a *successful derivation*), or $B = false$ (a *failed derivation*). In both cases a *final state* has been reached.

▶ **Example 1.** Consider the following first-order CHR program of the introduction with only one rule

$$(a(X), a(Y), s \Leftrightarrow true)$$

and $\{a(1), a(2), a(3), s\}$ as the store of constraints.

We give an ω_t derivation:

$$\begin{array}{rcl}
 & & \langle\{a(1), a(2), a(3), s\}, \emptyset, \emptyset\rangle_1 \\
[\mathit{Introduce}] & \rightsquigarrow_t & \langle\{a(2), a(3), s\}, \{a(1)\#1\}, \emptyset\rangle_2 \\
[\mathit{Introduce}] & \rightsquigarrow_t & \langle\{a(2), s\}, \{a(1)\#1, a(3)\#2\}, \emptyset\rangle_3 \\
[\mathit{Introduce}] & \rightsquigarrow_t & \langle\{a(2)\}, \{a(1)\#1, a(3)\#2, s\#3\}, \emptyset\rangle_4 \\
[\mathit{Apply}] & \rightsquigarrow_t & \langle\{a(2)\}, \emptyset, \{r.1.2.3\}\rangle_4 \\
[\mathit{Introduce}] & \rightsquigarrow_t & \langle\emptyset, \{a(2)\#4\}, \{r.1.2.3\}\rangle_5
\end{array}$$

The store in the final state is $\{a(2)\}$ but may be $\{a(1)\}$ or $\{a(3)\}$ since the order of [*Introduce*] derivation steps is arbitrary.

The semantics of this program is only clear if we consider its extensional version with the grounded rules in this (arbitrary) order:

$$(a(1), a(2), s \Leftrightarrow true) \qquad (a(1), a(3), s \Leftrightarrow true) \qquad (a(2), a(3), s \Leftrightarrow true)$$

and the initial store (a sequence) of constraints as, for example, $a(1), a(2), a(3), s$. When the constraint s is considered the constraints $a(1)$, $a(2)$ and $a(3)$ are already in the store of constraints. The first rule is tried and the matching of its multiple head with the store of constraints is a success. Since it is a simplification rule, the constraints $a(1)$ and $a(2)$ are deleted from the store of constraints. The final store of constraints is then $\{a(3)\}$.

The operational ω_r semantics. There exists a *refined* operational semantics ω_r [8] which considers the goal as a sequence instead of a multi-set. This semantics is very closed to the way it is usually implemented. It also uses a transition system with identified constraints, identities and propagation history. The operational semantics ω_t is based on six transitions which map a state to another state.

Linear-logic semantics of [6, 5]. This linear-logic semantics is directly inspired by the classical first-order logic semantics: goals (and stores of constraints) are translated to multiplicative conjunctions, simpagation rules $(K\backslash D \Leftrightarrow B)$ to the linear-logic formulas: $!(K \otimes D) \multimap (K \otimes B)$ and a CHR program to a large conjunction of linear-logic formulas. We denote by $(.)^L$ the above translation. A CHR program P has a computation with initial store S_0 and final store S_n if and only if $(P)^L \vdash ((S_0)^L \multimap (S_n)^L)$.

Axiomatic linear semantics of [5, 7]. The *axiomatic linear semantics* is based on the cut-rule of the linear logic and proper axioms: each CHR rule of the program is interpreted as an axiom. A goal is solved if there exists a linear proof of *true* in a linear-logic sequent calculus augmented by the proper axioms.

None of the previous semantics offers a semantics for the example of the introduction since they all manage the store of constraints as an unordered multi-set.

3 ω_l and $\omega_l^{\otimes}$ sequent calculus

In this section, we first define two sequent calculi: the ω_l and the $\omega_l^{\otimes}$ sequent calculi. The first one keeps the multi-sets of the ω_t and ω_r semantics while the second uses a sequence. Then we prove that the ω_l system is sound and complete w.r.t. the ω_t semantics while the $\omega_l^{\otimes}$ system is sound (but not complete) w.r.t. the ω_t semantics. Finally we give a translation from the ω_l (and $\omega_l^{\otimes}$) system to the linear-logic sequent calculus and prove the soundness of this translation.

3.1 ω_l and $\omega_l^{\otimes}$ systems

We first define the notion of store for the ω_l and $\omega_l^{\otimes}$ systems.

▶ **Definition 2** (ω_l and $\omega_l^{\otimes}$ stores)**.** An ω_l store is a multi-set of identified constrains. An $\omega_l^{\otimes}$ store is a sequence of identified constraints.

The ω_l and $\omega_l^{\otimes}$ systems are based on two kinds of sequents: the *focused* sequent is focused on a particular identified constraint, the current identified constraint, while the *non focused* sequent works on a sequence of identified constraints, the current goal.

We first define our sequents for the ω_l and $\omega_l^{\otimes}$ systems.

▶ **Definition 3** (non focused and focused ω_l and $\omega_l^{\otimes}$ sequents)**.**

- A *non focused* sequent is a quadruple $(\Gamma \blacktriangleright \Omega_\# \blacktriangleleft S_\uparrow \vdash \ S_\downarrow)$ where $S_\downarrow$, the *down store*, and $S_\uparrow$, the *up store*, are two stores of identified constraints, Γ is a sequence of CHR rules and $\Omega_\#$, the *goal*, is a sequence of identified constraints[3].
- A *focused* sequent is a quintuple $(\Gamma \,!\, \Delta \triangleright a \triangleleft S_\uparrow \ \vdash \ S_\downarrow)$ where $S_\downarrow$, $S_\uparrow$ and Γ are defined as for the non focused sequent, Δ is an ending sequence of Γ and a is an identified constraint.

The intuitive meaning of a non focused sequent $(\Gamma \blacktriangleright \Omega_\# \blacktriangleleft S_\uparrow \vdash \ S_\downarrow)$ is to try and consume the identified constraints $\Omega_\#$ [4] with the sequence of CHR rules Γ thanks to the store $S_\uparrow$. The elements of the store $S_\downarrow$ are the unconsumed identified constraints: the identified constraints of $S_\uparrow$ that have not been consumed and those produced by $\Omega_\#$ and not consumed during this production.

The intuitive meaning of a focused sequent $(\Gamma \,!\, \Delta \triangleright A\#i \triangleleft S_\uparrow \ \vdash \ S_\downarrow)$ is the same as for a non focused sequent but restricted to a unique identified constraint $A\#i$ which may be consumed only by the sequence of CHR rules Δ [5].

In our sequent calculi, the final store of identified constraints is what we have to prove. Solve the problem represented by a CHR program and an initial goal is to prove *true*.

Now we define the $\omega_l^{\otimes}$ sequent calculus:

[3] Note that in the ω_t semantics the goal is a set of constraints
[4] i.e. to solve the constraints of $const(\Omega_\#)$
[5] the identified constraints produced by $A\#i$ may be consumed by the CHR rules of Γ

▶ **Definition 4** ($\omega_l^{\otimes}$ sequent calculus system). The $\omega_l^{\otimes}$ system is based on four types of $\omega_l^{\otimes}$ inference rules ($S_{\downarrow}$, $S_{\downarrow}^a$, $S_{\uparrow}$, $S_{\uparrow}^a$, $S_{\uparrow}^B$, $S_{\downarrow}^B$, $S_{\downarrow}^{\Omega}$, $S_{\uparrow}^{\Omega}$, S, S^K, S^D, $S^{\subseteq K}$, $S_{\uparrow}^{\subseteq K}$ some stores; $K_1, \ldots, K_m$, $D_1, \ldots, D_n$, $B_1, \ldots, B_p$ some constraints, B a sequence of constraints; a an identified constraint; $\Omega_{\#}$, the goal, a sequence of identified constraints; i, i' some integers).

- The *non focused* subsystem:
  - The *true* axiom:

$$\frac{}{\Gamma \blacktriangleright \mathit{true} \blacktriangleleft S \vdash\ S}\ \mathit{true}$$

  - The *Left-elimination-of-conjunction* inference rule:

$$\frac{\Gamma \blacktriangleright a \blacktriangleleft S_{\uparrow}^a \vdash\ S_{\downarrow}^a \qquad \Gamma \blacktriangleright \Omega_{\#} \blacktriangleleft S_{\downarrow}^a, S_{\uparrow}^{\Omega} \vdash\ S_{\downarrow}^{\Omega}}{\Gamma \blacktriangleright a, \Omega_{\#} \blacktriangleleft S_{\uparrow}^a, S_{\uparrow}^{\Omega} \vdash\ S_{\downarrow}^{\Omega}}\ \otimes_L$$

  - The *Exchange* inference rule:

$$\frac{\Gamma \blacktriangleright \Omega_{\#} \blacktriangleleft A'\#i', A\#i, S_{\uparrow} \vdash\ S_{\downarrow}}{\Gamma \blacktriangleright \Omega_{\#} \blacktriangleleft A\#i, A'\#i', S_{\uparrow} \vdash\ S_{\downarrow}}\ X$$

    with the proviso that $A \neq A'$.
- The *focused* subsystem:
  - The *Inactivate* axiom:

$$\frac{}{\Gamma\ !\ \triangleright a \triangleleft S\ \vdash\ a, S}\ \uparrow$$

  - The *Weakening* inference rule:

$$\frac{\Gamma\ !\ \Delta \triangleright a \triangleleft S_{\uparrow}\ \vdash\ S_{\downarrow}}{\Gamma\ !\ (K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow G), \Delta \triangleright a \triangleleft S_{\uparrow}\ \vdash\ S_{\downarrow}}\ W$$

    with no j, ($1 \leq j \leq n$ such that $D_j = \mathit{const}(a)$ or $1 \leq j \leq m$ such that $K_j = \mathit{const}(a)$), $S^D \subseteq S_{\uparrow}$, $S^K \subseteq S_{\uparrow}$, $S^D \uplus S^K \uplus \{a\} = \{K_1\#i_1, \ldots, K_m\#i_m, D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}\}$[6].
- The *Focusing* inference rule:

$$\frac{\Gamma\ !\ \Gamma \triangleright a \triangleleft S_{\uparrow}\ \vdash\ S_{\downarrow}}{\Gamma \blacktriangleright a \blacktriangleleft S_{\uparrow} \vdash\ S_{\downarrow}}\ F$$

- The *Apply* inference rule:

$$\frac{\Gamma \blacktriangleright B_1\#i', \ldots, B_p\#(i'+p) \blacktriangleleft S^K, S_{\uparrow}^B \vdash\ S^{\subseteq K}, S_{\downarrow}^B \qquad \Gamma \blacktriangleright S^{\subseteq K} \blacktriangleleft S_{\downarrow}^B, S_{\uparrow}^{\subseteq K} \vdash\ S_{\downarrow}}{\Gamma\ !\ (K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow B_1, \ldots, B_p), \Delta \triangleright a \triangleleft S^D, S^K, S_{\uparrow}^B, S_{\uparrow}^{\subseteq K}\ \vdash\ S_{\downarrow}}\ \backslash \Leftrightarrow$$

  with either there exists j, $1 \leq j \leq n$ such that $D_j = \mathit{const}(a)$, $S^K = K_1\#i_1, \ldots, K_m\#i_m$, a inserted in S^D at place j is equal to $D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}$, or there exists j, $1 \leq j \leq m$ such that $K_j = \mathit{const}(a)$, a inserted in S^K at place j is equal to $K_1\#i_1, \ldots, K_m\#i_m$, $S^D = D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}$; i' a new integer, $S^{\subseteq K}$ is a subsequence of $K_1\#i_1, \ldots,$ $K_m\#i_m$.

The ω_l sequent calculus system is less structurally constrained than the $\omega_l^{\otimes}$ system:

▶ **Definition 5** (ω_l sequent calculus system). The ω_l sequent calculus system is the $\omega_l^{\otimes}$ sequent calculus system where the store of identified constraints and the multiple heads of rules are multi-sets instead of sequences and the *Exchange* inference rule is omitted.

[6] When used with multi-set operations, sequences are considered as multi-sets

The *non focused* system splits the current goal and allocates the resources. If the current goal is the *true* goal then no identified constraint is consumed and the *true* axiom is applied. If the current goal is a sequence of identified constraints, the *Left-elimination-of-conjunction* inference rule is applied: The first identified constraint a of the sequence is isolated and a part of the resources $S^a_\uparrow$ are allocated to solve the constraint $const(a)$, the rest of the identified constraints, $S^\Omega_\uparrow$, and those produced by a but unconsumed, $S^a_\downarrow$, are allocated to the sequence of identified constraints $S^{\subseteq K}$[7]. This inference rule realizes in fact a hidden use of the cut-rule of the linear-logic sequent calculus: the $S^a_\downarrow$ is a lemma computed by the left subproof and used in the right subproof. Both operational semantics eliminate those instances of the cut-rule in order to linearize the derivation.

The *focused* system chooses, if any, a CHR rule to be applied on the focused identified constraint a. If no such CHR rule exists, the *Inactivate* axiom stores the identified constraint into the store. The *Weakening* inference eliminates, in the order of the sequence Δ, the first CHR rule $(K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow B)$ that cannot be applied since there are no subset S^K and S^D of $S_\uparrow$ such that $S^K \uplus S^D \uplus \{a\} = \{K_1\#i_1, \ldots, K_m\#i_m, D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}\}$.

The *Focusing* inference rule flips from the non focused $\omega_l^\otimes$ system to the focused $\omega_l^\otimes$ system by focusing on an identified constraint.

The *Apply* inference rule flips from focused $\omega_l^\otimes$ system to non focused $\omega_l^\otimes$ system by applying a CHR rule $(K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow B)$ on the focused identified constraint a since there are two subsequences S^K and S^D of $S_\uparrow$ such that $S^K \uplus S^D \uplus \{a\} = \{K_1\#i_1, \ldots, K_m\#i_m, D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}\}$. The solving of the constraint underlying the identified constraint a is reduced to the solving of the goal of the CHR rule $B = B_1, \ldots, B_p$ and eventually the solving of the constraints underlying $S^{\subseteq K}$ in the case that identified constraints from $S^{\subseteq K} \subseteq S^K$ were not consumed during the process of consumption/production of $B_1\#i', \ldots, B_p\#i'+p$. As for the *Left-elimination-of-conjunction* inference rule a part of the resources $S^K \uplus S^B_\uparrow$ is allocated to solve the goal $B_1\#i', \ldots, B_p\#(i'+p)$, the rest of the identified constraints $S^{\subseteq K}$ and those produced by $B_1\#i', \ldots, B_p\#(i'+p)$ but unconsumed $S^B_\downarrow$ are allocated to a sequence $S^{\subseteq K}$. Since the $\omega_l^\otimes$ system only applies a CHR rule if one of the identified constraints of its head is focused on, the calculus of $(\Gamma \blacktriangleright S^{\subseteq K} \blacktriangleleft S^B_\downarrow, S^{\subseteq K}_\uparrow \vdash S_\downarrow)$ is necessary to the completeness. But $S^{\subseteq K}$ is not necessarily equal to $K_1\#i_1, \ldots, K_m\#i_m$ since some identified constraints may have been consumed during the process of consumption/production of $B_1\#i', \ldots, B_p\#(i'+p)$. Moreover, $S^{\subseteq K}$ may be empty if all the resources have been consumed.

In a classical implementation of CHR, $S^{\subseteq K}_\uparrow$ is captured by the flow $S^B_\uparrow / S^B_\downarrow$. In this configuration $S^B_\uparrow$ is not anymore the necessary resources to prove B and $S^B_\downarrow$ the resources produced but unconsumed by B but respectively the input store and the output store of the derivation of B.

Once again, this *Apply* inference rule realizes in fact a hidden use of the cut-rule of the linear-logic sequent calculus: a lemma is computed by the left subproof and used in the right subproof. Both operational semantics eliminate those instances of the cut-rule in order to linearize the derivation.

When the applied CHR rule is such that $S^{\subseteq K} = \emptyset$ the *Apply* inference rule is simplified to

$$\frac{\Gamma \blacktriangleright B_1\#i', \ldots, B_p\#(i'+p) \blacktriangleleft S^K, S_\uparrow \vdash S_\downarrow}{\Gamma \,!\, (K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow B_1, \ldots, B_p), \Delta \triangleright a \triangleleft S^D, S^K, S_\uparrow \vdash S_\downarrow} \backslash \Leftrightarrow$$

[7] In the case of the ω_l system, the elements of the multi-set $S^{\subseteq K}$ must be ordered in a sequence $\Omega^{\subseteq K}_\#$.

Moreover, when the applied CHR rule is a simplification rule ($S^K = \emptyset$ and $S^{\subseteq K} = \emptyset$) then the *Apply* inference rule is simplified to

$$\dfrac{\Gamma \blacktriangleright B_1\#i', \ldots, B_p\#(i'+p) \blacktriangleleft S_\uparrow \vdash\ S_\downarrow}{\Gamma\ !\ (D_1, \ldots, D_n \Leftrightarrow B_1, \ldots, B_p), \Delta \triangleright a \triangleleft S^D, S_\uparrow\ \vdash\ S_\downarrow}\ \Leftrightarrow$$

▶ **Example 6.** What follows is a proof [8] in the ω_l system for the ω_l sequent

$(\Gamma \blacktriangleright d\#1, a\#2 \blacktriangleleft\ \vdash\ S_\downarrow) = (r_1, r_2, r_3 \blacktriangleright d\#1, a\#2 \blacktriangleleft\ \vdash\ c\#6, f\#5, g\#4, d\#1).$

with $S_\downarrow = \{c\#6, f\#5, g\#4, d\#1\}$, $\Gamma = r1@(d \Rightarrow e), r2@(a\backslash e \Leftrightarrow g), r3@(a \Leftrightarrow f, c)$

$$\dfrac{\nabla_1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \blacktriangleright g\#4 \blacktriangleleft a\#2, d\#1 \vdash\ g\#4, a\#2, d\#1}\,F\uparrow \qquad \dfrac{\nabla_2}{\Gamma \blacktriangleright a\#2 \blacktriangleleft g\#4, d\#1 \vdash\ S_\downarrow}\,F}{\Gamma\ !\ r2, r3 \triangleright a\#2 \triangleleft d\#1, e\#3\ \vdash\ S_\downarrow}\,\backslash\Leftrightarrow}{\Gamma\ !\ r1, r2, r3 \triangleright a\#2 \triangleleft d\#1, e\#3\ \vdash\ S_\downarrow}\,W}{\Gamma \blacktriangleright a\#2 \blacktriangleleft d\#1, e\#3 \vdash\ S_\downarrow}\,F}{\Gamma \blacktriangleright d\#1, a\#2 \blacktriangleleft\ \vdash\ S_\downarrow}\,\otimes_L$$

with ∇_1:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \blacktriangleright e\#3 \blacktriangleleft d\#1 \vdash\ e\#3, d\#1}\,F\uparrow \qquad \dfrac{}{\Gamma \blacktriangleright d\#1 \blacktriangleleft e\#3 \vdash\ d\#1, e\#3}\,F\uparrow}{\Gamma\ !\ r1, r2, r3 \triangleright d\#1 \triangleleft\ \ \vdash\ d\#1, e\#3}\,\backslash\Leftrightarrow}{\Gamma \blacktriangleright d\#1 \blacktriangleleft\ \vdash\ d\#1, e\#3}\,F$$

and ∇_2:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \blacktriangleright f\#5 \blacktriangleleft g\#4, d\#1 \vdash\ f\#5, g\#4, d\#1}\,F\uparrow \qquad \dfrac{}{\Gamma \blacktriangleright c\#6 \blacktriangleleft f\#5, g\#4, d\#1 \vdash\ S_\downarrow}\,F\uparrow}{\Gamma \blacktriangleright f\#5, c\#6 \blacktriangleleft g\#4, d\#1 \vdash\ S_\downarrow}\,\otimes_L}{\Gamma\ !\ r3 \triangleright a\#2 \triangleleft g\#4, d\#1\ \vdash\ S_\downarrow}\,\Leftrightarrow}{\Gamma\ !\ r2, r3 \triangleright a\#2 \triangleleft g\#4, d\#1\ \vdash\ S_\downarrow}\,W}{\Gamma\ !\ r1, r2, r3 \triangleright a\#2 \triangleleft g\#4, d\#1\ \vdash\ S_\downarrow}\,W$$

What follows is a proof in the $\omega_l^{\otimes}$ system:

$$\dfrac{\nabla_1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \blacktriangleright g\#4 \blacktriangleleft a\#2, d\#1 \vdash\ g\#4, a\#2, d\#1}\,F\uparrow \qquad \dfrac{\nabla_2}{\Gamma \blacktriangleright a\#2 \blacktriangleleft g\#4, d\#1 \vdash\ S_\downarrow}\,F}{\Gamma\ !\ r2, r3 \triangleright a\#2 \triangleleft e\#3, d\#1\ \vdash\ S_\downarrow}\,\backslash\Leftrightarrow}{\Gamma\ !\ r1, r2, r3 \triangleright a\#2 \triangleleft e\#3, d\#1\ \vdash\ S_\downarrow}\,W}{\Gamma \blacktriangleright a\#2 \blacktriangleleft e\#3, d\#1 \vdash\ S_\downarrow}\,F}{\Gamma \blacktriangleright a\#2 \blacktriangleleft d\#1, e\#3 \vdash\ S_\downarrow}\,X}{\Gamma \blacktriangleright d\#1, a\#2 \blacktriangleleft\ \vdash\ S_\downarrow}\,\otimes_L$$

Notice the use of the *Exchange* inference rule (X) in order to permute the identified constraints $d\#1$ and $e\#3$.

We give the first contribution of this article: the soundness and completeness theorem of the ω_l system w.r.t. the ω_t semantics:

[8] In this example, we define the $F\uparrow$ axiom: $\dfrac{}{r_1, r_2, \ldots, r_n \blacktriangleright a \blacktriangleleft S \vdash\ a, S}$ as a shorthand for an instance of a *Focusing* inference rule followed by many instances, as needed, of the *Weakening* inference rule and followed by an instance of the *Inactivate* axiom.

▶ **Theorem 7** (Soundness and completeness of the ω_l system w.r.t. the ω_t semantics). *Let Γ be a CHR program and $B_1, \ldots, B_p$ some constraints. The initial goal $B_1, \ldots, B_p$ is solved in the ω_t semantics by Γ with a final store (a multi-set) of identified constraints Σ if and only if there exists an ω_l proof of $(\Gamma \blacktriangleright B_1\#1, \ldots, B_p\#p \blacktriangleleft\ \vdash\ \Sigma)$.*

And as a corollary, we obtain the soundness of the $\omega_l^{\otimes}$ system w.r.t. the ω_t semantics:

▶ **Theorem 8** (Soundness of the $\omega_l^{\otimes}$ system w.r.t. the ω_t semantics). *Let Γ be a CHR program and $B_1, \ldots, B_p$ some constraints and Σ a store (a multi-set) of identified constraints. If there exists an $\omega_l^{\otimes}$ proof of $(\Gamma \blacktriangleright B_1\#1, \ldots, B_p\#p \blacktriangleleft\ \vdash\ S)$, where S is a sequence of Σ then the initial goal $B_1, \ldots, B_p$ is solved in the ω_t semantics by Γ with a final store Σ.*

The $\omega_l^{\otimes}$ system is not complete w.r.t. the ω_t semantics since the *Exchange* inference rule is limited to identified constraints that are based on different constraints.

▶ **Example 9** (Example of the introduction continued). We can prove with the $\omega_l^{\otimes}$ system the sequent $(r \blacktriangleright a(1)\#1, a(2)\#2, a(3)\#3, s\#4 \blacktriangleleft\ \vdash\ a(1)\#1)$:

$$\dfrac{\dfrac{}{r \blacktriangleright a(1)\#1 \blacktriangleleft\ \vdash\ a(1)\#1}\,F\uparrow \qquad \dfrac{\dfrac{}{r \blacktriangleright a(2)\#2 \blacktriangleleft a(1)\#1 \vdash\ a(2)\#2, a(1)\#1}\,F\uparrow \qquad \nabla}{r \blacktriangleright a(2)\#2, a(3)\#3, s\#4 \blacktriangleleft a(1)\#1 \vdash\ a(1)\#1}\,\otimes_L}{r \blacktriangleright a(1)\#1, a(2)\#2, a(3)\#3, s\#4 \blacktriangleleft\ \vdash\ a(1)\#1}\,\otimes_L$$

with ∇ $(S = a(3)\#3, a(2)\#2, a(1)\#1)$:

$$\dfrac{\dfrac{}{r \blacktriangleright a(3)\#3 \blacktriangleleft a(2)\#2, a(1)\#1 \vdash\ S}\,F\uparrow \qquad \dfrac{\dfrac{\dfrac{}{r \blacktriangleright \mathit{true} \blacktriangleleft a(1)\#1 \vdash\ a(1)\#1}\,\mathit{true}}{r\ !\ r \triangleright s\#4 \triangleleft S\ \vdash\ a(1)\#1}\,\Leftrightarrow}{r \blacktriangleright s\#4 \blacktriangleleft S \vdash\ a(1)\#1}\,F}{r \blacktriangleright a(3)\#3, s\#4 \blacktriangleleft a(2)\#2, a(1)\#1 \vdash\ a(1)\#1}\,\otimes_L$$

But not the sequent $(r \blacktriangleright a(3)\#3, a(2)\#2, a(1)\#1, s\#4 \blacktriangleleft\ \vdash\ a(2)\#2)$ of Example 1 nor the sequent $(r \blacktriangleright a(3)\#3, a(2)\#2, a(1)\#1, s\#4 \blacktriangleleft\ \vdash\ a(3)\#3)$ since the store S is a sequence (and not a multi-set) and the *Exchange* inference rule cannot be applied since the identified constraints $a(1)\#1$, $a(2)\#2$ and $a(3)\#3$ are based on the same constraint a.

3.2 Translation from ω_l and $\omega_l^{\otimes}$ systems into Linear Logic

We define a translation from the ω_l system into the linear-logic sequent calculus and prove that the result of the translation of a ω_l proof is a linear-logic proof in the sense of the definition of Section 2.1. We first give the translation of the CHR rules, then the translation for the ω_l sequents and finally the translation for the ω_l system. The translation from the $\omega_l^{\otimes}$ system into the linear-logic sequent calculus is directly obtained from previous translation by omitting the *Exchange* inference rule (and by considering sequences as multi-sets).

▶ **Definition 10** (Translation of the CHR rules and CHR programs into linear-logic formulas). The CHR rules are translated into linear-logic formulas as follows thanks to the function $(.)_\Gamma$:

- $(K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow \mathit{true})_\Gamma =$
  $\forall x_1 \ldots \forall x_{m+n}$
  $((K_1(x_1) \otimes \ldots \otimes K_m(x_m) \otimes D_{m+1}(x_{m+1}) \otimes \ldots \otimes D_{m+n}(x_{m+n}))$
  $\multimap (K_1(x_1) \otimes \ldots \otimes K_m(x_m) \otimes \mathbf{1}))$

- $(K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow \mathit{false})_\Gamma =$
  $\forall x_1 \ldots \forall x_{m+n}$
  $((K_1(x_1) \otimes \ldots \otimes K_m(x_m) \otimes D_{m+1}(x_{m+1}) \otimes \ldots \otimes D_{m+n}(x_{m+n}))$
  $\multimap (K_1(x_1) \otimes \ldots \otimes K_m(x_m) \otimes \mathbf{0}))$
- $(K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow B_1, \ldots, B_p)_\Gamma =$
  $\forall x_1 \ldots \forall x_{m+n} \exists y_1 \ldots \exists y_p$
  $((K_1(x_1) \otimes \ldots \otimes K_m(x_m) \otimes D_{m+1}(x_{m+1}) \otimes \ldots \otimes D_{m+n}(x_{m+n}))$
  $\multimap (K_1(x_1) \otimes \ldots \otimes K_m(x_m) \otimes B_1(y_1) \otimes \ldots \otimes B_p(y_p)))$
- $(r, \Delta)_\Gamma = (r)_\Gamma \;\&\; (\Delta)_\Gamma$ with r a CHR rule and Δ a non empty sequence of CHR rules.

CHR constant *true* is interpreted to $\mathbf{1}$, the neutral of $\otimes$ the multiplicative conjunction. CHR constant *false* is interpreted to $\mathbf{0}$ which has no elimination rule. Introduction of new identities are interpreted to existential quantifications in order to generate a brand new one each time while transmission of identities of identified constraints are interpreted by universal quantifications. In a CHR rule, symbol "$\Leftrightarrow$" is interpreted to linear implication and symbol "," is interpreted to the multiplicative conjunction $\otimes$. Finally, in CHR program, symbol "," is interpreted to the additive conjunction &, an (ordered) committed choice.

▶ **Example 11** (Example continued)**.**

$$(r_1)_\Gamma = (d \Rightarrow e)_\Gamma = (\forall x\ (\exists y\ (d(x) \multimap d(x) \otimes e(y))))$$
$$(r_2)_\Gamma = (a \backslash e \Leftrightarrow g)_\Gamma = (\forall x, x'\ (\exists y\ (a(x) \otimes e(x') \multimap a(x) \otimes g(y))))$$
$$(r_3)_\Gamma = (a \Leftrightarrow f, c)_\Gamma = (\forall x\ (\exists y, y'\ (a(x) \multimap f(y) \otimes c(y'))))$$

▶ **Definition 12** (Translation of the ω_l sequents into linear sequent)**.** The ω_l system language is translated into Linear Logic as follows thanks to three functions $(.)_\Omega$, $(.)_\uparrow$ and $(.)_\downarrow$ for translating respectively the goal, the up store and the down store of an ω_l sequent.

- $\begin{cases} (\mathit{true})_\Omega = \mathbf{1}, (\mathit{false})_\Omega = \mathbf{0}, \\ (A\#i)_\Omega = A(i) \text{ a token, with } A \text{ a constraint and } i \text{ an identity,} \\ ((a, \Omega^\#))_\Omega = ((a)_\Omega \otimes (\Omega^\#)_\Omega) \\ \text{ with } a \text{ an identified constraint and } \Omega^\# \text{ a sequence of identified constraints} \end{cases}$
- $\begin{cases} (A\#i)_\uparrow = A(i) \text{ a token, with } A \text{ a constraint and } i \text{ an identity,} \\ (S)_\uparrow = \{(a)_\uparrow \mid a \in S\} \\ \text{ with } a \text{ an identified constraint and } S \text{ a store.} \end{cases}$
- $\begin{cases} (A\#i)_\downarrow = A(i) \text{ a token, with } A \text{ a constraint and } i \text{ an identity,} \\ (S)_\downarrow = \bigotimes_{a \in S} (a)_\downarrow \\ \text{ with } a \text{ an identified constraint and } S \text{ a store.} \end{cases}$

For any ω_l sequent is translated into a linear sequent as follows thanks to the function $L(.)$:

- $L(\Gamma \blacktriangleright \Omega^\# \blacktriangleleft S_\uparrow \vdash\ S_\downarrow) = !(\Gamma)_\Gamma, (\Omega^\#)_\Omega, (S_\uparrow)_\uparrow\ \vdash\ (S_\downarrow)_\downarrow$
- $L(\Gamma\ !\ \Delta \triangleright \Omega^\# \triangleleft S_\uparrow\ \vdash\ S_\downarrow) = !(\Gamma)_\Gamma, (\Delta)_\Gamma \& \mathbf{1}, (\Omega^\#)_\Omega, (S_\uparrow)_\uparrow\ \vdash\ (S_\downarrow)_\downarrow$
- $L(\Gamma\ !\ \ \triangleright \Omega^\# \triangleleft S_\uparrow\ \vdash\ S_\downarrow) = !(\Gamma)_\Gamma, (\Omega^\#)_\Omega, (S_\uparrow)_\uparrow\ \vdash\ (S_\downarrow)_\downarrow$

The goal and the down store of identified constraints of the ω_l sequent are interpreted to multiplicative conjunctions of tokens while the up store of identified constraints is interpreted to a sequence of tokens. The multiplicative conjunction of the goal induces a sequence on the identified constraints of the goal. The multiplicative conjunction of the goal allows the introduction of the cut-rule of the *Left-elimination-of-conjunction* inference and *Apply* inference rules.

The CHR program is interpreted as a large additive conjunction of linear implications ended with the **1** constant in order to allow the move in the *Inactivate* inference rule of the identified constraint from the goal to the down store when no CHR rule is found to be applied.

▶ **Definition 13** (Translation of the ω_l system into the linear-logic sequent calculus)**.** The *non focused* ω_l system:

- *true* axiom

$$\dfrac{}{\Gamma \blacktriangleright \mathit{true} \blacktriangleleft S \vdash\ S}\ \mathit{true}$$

is translated into [9]

$$\dfrac{\dfrac{\dfrac{}{(S)_\uparrow\ \vdash\ (S)_\downarrow}\ I\otimes}{\mathbf{1}, (S)_\uparrow\ \vdash\ (S)_\downarrow}\ \mathbf{1}L}{L(\Gamma \blacktriangleright \mathbf{1} \blacktriangleleft S \vdash\ S)}\ !W$$

- *Left-elimination-of-conjunction* inference rule:

$$\dfrac{\Gamma \blacktriangleright a \blacktriangleleft S^a_\uparrow \vdash\ S^a_\downarrow \qquad \Gamma \blacktriangleright \Omega^\# \blacktriangleleft S^a_\downarrow, S^\Omega_\uparrow \vdash\ S^\Omega_\downarrow}{\Gamma \blacktriangleright a, \Omega^\# \blacktriangleleft S^a_\uparrow, S^\Omega_\uparrow \vdash\ S^\Omega_\downarrow}\ \otimes_L$$

is translated into

$$\dfrac{\dfrac{\dfrac{L(\Gamma \blacktriangleright a \blacktriangleleft S^a_\uparrow \vdash\ S^a_\downarrow) \qquad \dfrac{L(\Gamma \blacktriangleright \Omega^\# \blacktriangleleft S^a_\downarrow, S^\Omega_\uparrow \vdash\ S^\Omega_\downarrow)}{!(\Gamma)_\Gamma, (\Omega^\#)_\Omega, (S^a_\downarrow)_\downarrow, (S^\Omega_\uparrow)_\uparrow\ \vdash\ (S^\Omega_\downarrow)_\downarrow}\ \otimes L*}{!(\Gamma)_\Gamma, !(\Gamma)_\Gamma, (a)_\Omega, (\Omega^\#)_\Omega, (S^a_\uparrow)_\uparrow, (S^\Omega_\uparrow)_\uparrow\ \vdash\ (S^\Omega_\downarrow)_\downarrow}\ \mathit{Cut}}{!(\Gamma)_\Gamma, (a)_\Omega, (\Omega^\#)_\Omega, (S^a_\uparrow)_\uparrow, (S^\Omega_\uparrow)_\uparrow\ \vdash\ (S^\Omega_\downarrow)_\downarrow}\ !C}{L(\Gamma \blacktriangleright (a, \Omega^\#) \blacktriangleleft S^a_\uparrow, S^\Omega_\uparrow \vdash\ S^\Omega_\downarrow)}\ \otimes L$$

- The *focused* ω_l system:
  - Weakening rule:

$$\dfrac{\Gamma\ !\ \Delta \triangleright a \triangleleft S_\uparrow\ \vdash\ S_\downarrow}{\Gamma\ !\ (K\backslash D \Leftrightarrow B), \Delta \triangleright a \triangleleft S_\uparrow\ \vdash\ S_\downarrow}\ W$$

  is translated into

$$\dfrac{L(\Gamma\ !\ \Delta \triangleright a \triangleleft S_\uparrow\ \vdash\ S_\downarrow)}{L(\Gamma\ !\ (K\backslash D \Leftrightarrow B), \Delta \triangleright a \triangleleft S_\uparrow\ \vdash\ S_\downarrow)}\ \&L_2$$

  - Inactivate rule:

$$\dfrac{}{\Gamma\ !\ \triangleright A\#i \triangleleft S\ \vdash\ A\#i, S}\ \uparrow$$

  is translated into

---

[9] The following axiom $I\otimes$: $\dfrac{}{B_1, B_2, \ldots, B_n\ \vdash\ B_1 \otimes B_2 \otimes \ldots \otimes B_n}$ is a shorthand for the following linear proof

$$\dfrac{\dfrac{}{B_1\ \vdash\ B_1}\ I \qquad \dfrac{\vdots \quad \dfrac{\dfrac{}{B_{n-1}\ \vdash\ B_{n-1}}\ I \qquad \dfrac{}{B_n\ \vdash\ B_n}\ I}{B_{n-1}, B_n\ \vdash\ B_{n-1} \otimes B_n}\ \otimes R}{B_2, \ldots, B_n\ \vdash\ B_2 \otimes \ldots \otimes B_n}}{B_1, B_2, \ldots, B_n\ \vdash\ B_1 \otimes B_2 \otimes \ldots \otimes B_n}\ \otimes R$$

$$\dfrac{\dfrac{}{A(i),(S)_\uparrow \;\vdash\; (A(i)\otimes(S)_\downarrow)}\,I\otimes}{L(\Gamma\;!\;\;\triangleright A\#i \triangleleft S \;\vdash\; A\#i, S)}\,!W$$

- The *Focusing rule*:

$$\dfrac{\Gamma\;!\;\Gamma \triangleright a \triangleleft S_\uparrow \;\vdash\; S_\downarrow}{\Gamma \blacktriangleright a \blacktriangleleft S_\uparrow \vdash\; S_\downarrow}\,F$$

is translated into

$$\dfrac{\dfrac{L(\Gamma\;!\;\Gamma \triangleright a \triangleleft S_\uparrow \;\vdash\; S_\downarrow)}{!(\Gamma)_\Gamma, !(\Gamma)_\Gamma, (a)_\Omega, (S_\uparrow)_\uparrow \;\vdash\; (S_\downarrow)_\downarrow}\,!D}{L(\Gamma \blacktriangleright a \blacktriangleleft S_\uparrow \vdash\; S_\downarrow)}\,!C$$

- The *Apply* rule with

$(K_1,\ldots,K_m\backslash D_1,\ldots,D_n \Leftrightarrow B)_\Gamma =$
$\forall x_1\ldots\forall x_{m+n}\exists y_1\ldots\exists y_p((K_1(x_1)\otimes\ldots\otimes K_m(x_m)\otimes D_{m+1}(x_{m+1})\otimes\ldots\otimes D_{m+n}(x_{m+n}))$
$\multimap (K_1(x_1)\otimes\ldots\otimes K_m(x_m)\otimes B_1(y_1)\otimes\ldots\otimes B_p(y_p)))$

such that $K_\otimes = K_1(i)\otimes\ldots\otimes K_m(i+m)$, $D_\otimes = D_1(i+m+1)\otimes\ldots\otimes D_n(i+m+n)$, $B = B_1,\ldots,B_p$, $B_\# = B_1\#i',\ldots,B_p\#(i'+p)$, $i' = i+m+n+1$ and $p > 0$, and $B_\otimes = B_1(i')\otimes\ldots\otimes B_p(i'+p) = (B_1\#i',\ldots,B_p\#(i'+p))_\Omega$. The *Apply* rule

$$\dfrac{\Gamma \blacktriangleright B_\# \blacktriangleleft S^K, S^B_\uparrow \vdash\; S^{\subseteq K}, S^B_\downarrow \qquad \Gamma \blacktriangleright \Omega^{\subseteq K}_\# \blacktriangleleft S^B_\downarrow, S^{\subseteq K}_\uparrow \vdash\; S_\downarrow}{\Gamma\;!\;(K_1,\ldots,K_m\backslash D_1,\ldots,D_n \Leftrightarrow B),\Delta \triangleright a \triangleleft S^D, S^K, S^B_\uparrow, S^{\subseteq K}_\uparrow \;\vdash\; S_\downarrow}\,\backslash\Leftrightarrow$$

is translated into

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(a)_\Omega,(S^D)_\uparrow,(S^K)_\uparrow \;\vdash\; K_\otimes\otimes D_\otimes}\,I\otimes \qquad \nabla}{!(\Gamma)_\Gamma,(K_\otimes\otimes D_\otimes \multimap K_\otimes\otimes B_\otimes),(a)_\Omega,(S^D)_\uparrow,(S^K)_\uparrow,(S^B_\uparrow)_\uparrow,(S^{\subseteq K}_\uparrow)_\uparrow \;\vdash\; (S_\downarrow)_\downarrow}\,\multimap L}{!(\Gamma)_\Gamma,(K_1,\ldots,K_m\backslash D_1,\ldots,D_n \Leftrightarrow B)_\Gamma,(a)_\Omega,(S^D)_\uparrow,(S^K)_\uparrow,(S^B_\uparrow)_\uparrow,(S^{\subseteq K}_\uparrow)_\uparrow \;\vdash\; (S_\downarrow)_\downarrow}\,\exists L^*\forall L^*}{L(\Gamma\;!\;(K_1,\ldots,K_m\backslash D_1,\ldots,D_n \Leftrightarrow B),\Delta \triangleright a \triangleleft S^D, S^K, S^B_\uparrow, S^{\subseteq K}_\uparrow \;\vdash\; S_\downarrow)}\,\&L_1$$

with $\nabla =$

$$\dfrac{\dfrac{\dfrac{L(\Gamma \blacktriangleright B_\# \blacktriangleleft S^K, S^B_\uparrow \vdash\; S^{\subseteq K}, S^B_\downarrow) \qquad \dfrac{L(\Gamma \blacktriangleright \Omega^{\subseteq K}_\# \blacktriangleleft S^B_\downarrow, S^{\subseteq K} \vdash\; S_\downarrow)}{!(\Gamma)_\Gamma,(\Omega^{\subseteq K}_\#)_\Omega\otimes(S^B_\downarrow)_\downarrow,(S^{\subseteq K})_\uparrow \;\vdash\; (S_\downarrow)_\downarrow}\,\otimes L*}{!(\Gamma)_\Gamma,!(\Gamma)_\Gamma,(S^K)_\uparrow,B_\otimes,(S^B_\uparrow)_\uparrow,(S^{\subseteq K}_\uparrow)_\uparrow \;\vdash\; (S_\downarrow)_\downarrow}\,Cut}{!(\Gamma)_\Gamma,(S^K)_\uparrow,B_\otimes,(S^B_\uparrow)_\uparrow,(S^{\subseteq K}_\uparrow)_\uparrow \;\vdash\; (S_\downarrow)_\downarrow}\,!C}{!(\Gamma)_\Gamma,K_\otimes\otimes B_\otimes,(S^B_\uparrow)_\uparrow,(S^{\subseteq K}_\uparrow)_\uparrow \;\vdash\; (S_\downarrow)_\downarrow}\,\otimes L*$$

Note that since $\Omega^{\subseteq K}_\#$ is a sequence over $S^{\subseteq K}$, it may be chosen such that $(\Omega^{\subseteq K}_\#)_\Omega = (S^{\subseteq K})_\downarrow$. If $S^{\subseteq K} = \emptyset$ or $B = \mathit{true}$ or $B = \mathit{false}$ the above translation is simplified in a straightforward manner.

The linear cut-rule is used in the translation of the *Left-elimination-of-conjunction* inference rule in order to transmit the down store of the left subproof to the right subproof. This down store which is a multiplicative conjunction is then split into a sequence of identified constraints thanks to linear-logic $\otimes$ left elimination $\otimes L$-rule.

Weakening inference rule tries the CHR rules in the order of the CHR program thanks to the linear-logic $\& L_2$ rule.

The linear cut-rule is also used in the translation of the *Apply* inference rule in order to transmit the down store of the left subproof to the the right subproof if S^K has not been completely consumed by the subproof (ie. $S^{\subseteq K} \neq \emptyset$). This down store which is a multiplicative conjunction is then split into a sequence of identified constraints thanks to the linear-logic $\otimes$ left elimination $\otimes L$-rule.

We now establish the second contribution of this article, the soundness of the translation from the ω_l system to the linear-logic sequent calculus:

▶ **Theorem 14.** *The result of the translation by Definitions 10, 12 and 13 of an ω_l proof is a linear proof.*

As a direct corollary, the soundness of the translation from the $\omega_l^{\otimes}$ system to the linear-logic sequent calculus with the same translation that for the ω_l system (instances of the *Exchange* inference rule are simply ignored):

▶ **Theorem 15.** *The result of the translation by Definitions 10, 12 and 13 of an $\omega_l^{\otimes}$ proof is a linear proof.*

4 Discussion

[3] proposes a normalization process of the Linear Logic proofs to a subclass of proofs, called the "focusing" proofs, which is complete (any derivable formula in Linear Logic has a focusing proof). Focusing proofs are expressed in a Triadic system, which respects the symmetry of Linear Logic. This process of normalization informally interleaves a *don't care* nondeterministic phase on *asynchronous* formulae and a phase applied on a *synchronous focused* formula. This last phase is a critical section and *don't know* nondeterminism can only appear during this phase. Since our $\omega_l^{\otimes}$ system is completely deterministic, the two phases of the $\omega_l^{\otimes}$ system are not based on the same principles as the two phases of the Triadic system. But, since the Triadic system is complete w.r.t. Linear Logic, it would be interesting to translate the ω_l and $\omega_l^{\otimes}$ proofs in focusing proofs to understand the semantics of CHR in terms of synchronous and asynchronous connectors.

5 Conclusion

We have proposed in this article two new proof-theoretical linear sequent systems for the semantics of CHR. The $\omega_l^{\otimes}$ system makes the semantics of the language completely deterministic. This semantics overcomes the hidden nondeterminism due to the management of the store of identified constraints and the multiple head of rules as multi-sets. But we can reintroduce the *don't care* nondeterminism of the committed choice principle if we allow the weakening inference rule even if the CHR rule is applicable (and of course also the *don't know* nondeterminism). Due to the lack of space, we cannot present a restricted version of the *Apply* inference rule (with $S^{\subseteq K}$ replaces only by K) which corresponds more faithfully to the ω_r semantics.

References

1. S. Abdennadher. Operational semantics and confluence of constraint propagation rules. In *Proceedings of the 3rd International Conference on Principles and Practice of Constraint Programming (CP'97)*, pages 252–266, 1997.
2. S. Abdennadher and H. Schütz. CHR$^{\forall}$: A Flexible Query Language. In *Proceedings of the 3^{rd} International Conference on Flexible Query Answering Systems*, pages 1–14, 1998.
3. J.M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of logic and computation*, 2(3):297–347, 1992.
4. H. Betz. A linear-logic semantics for constraint handling rules with disjunction. In *Proceedings of the 4th Workshop on Constraint Handling Rules (CP'07)*, pages 17–31, 2007.
5. H. Betz. A Unified Analytical Foundation for Constraint Handling Rules, PhD thesis, Ulm University, 2014.
6. H. Betz and T.W. Frühwirth. A linear-logic semantics for constraint handling rules. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP'05)*, pages 137–151, 2005.
7. H. Betz and T.W. Frühwirth. Linear-logic based analysis of constraint handling rules with disjunction. *ACM Transactions on Computational Logic*, 14(1), 2013.
8. G.J. Duck, P.J. Stuckey, M.G. de la Banda, and C. Holzbaur. The refined operational semantics of constraint handling rules. In *Proceedings of the 20th International Conference on Logic Programming (ICLP'04)*, pages 90–104, 2004.
9. T.W. Frühwirth. Constraint Handling Rules. Technical report, ECRC, 1992.
10. T.W. Frühwirth. Constraint Handling Rules. In *Constraint Programming: Basics and Trends*, pages 90–107, 1994.
11. T.W. Frühwirth. Theory and practice of constraint handling rules. *Journal of Logic Programming*, 37(1-3):95–138, 1998.
12. T.W. Frühwirth. *Constraint Handling Rules.* Cambridge University Press, 2009.
13. T.W. Frühwirth and S. Abdennadher. *Essentials of Constraint Programming.* Springer-Verlag, 2003.
14. T.W. Frühwirth and F. Raiser, editors. *Constraint Handling Rules: Compilation, Execution, and Analysis.* March 2011.
15. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
16. P. Van Hentenryck. Constraint logic programming. *Knowledge Engineering Review*, 6(3):151–194, 1991.
17. J. Jaffar and J.-L. Lassez. Constraint logic programming. In *Proceedings of the 14th Annual ACM Symposium on Principles of Programming Languages*, pages 111–119, 1987.
18. J. Jaffar and M.J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.
19. F. Raiser, H. Betz, and Thom Frühwirth. Equivalence of CHR states revisited. In *Proceedings of the* 6^{th} *International Workshop on Constraint Handling Rules*, pages 34–48, 2009.

6 Appendix

In order to prove the soundness and completeness of the ω_l system w.r.t. the ω_t semantics, we first introduce the ω_t sequent calculus system that imitates faithfully the ω_t semantics. Hence we prove the soundness and completeness of this ω_t system w.r.t. the ω_t semantics and then prove the soundness and completeness of ω_l system w.r.t. ω_t system.

We first define what is a ω_t sequent.

▶ **Definition 16** (ω_t sequent)**.** An ω_t sequent is a triplet $(\Gamma \blacktriangleright \Omega \blacktriangleleft S \vdash \ )$ where S, the store of identified constraints, is a multi-set of identified constraints, Ω, the current goal, is a multi-set of constraints and Γ, the program, is a sequence of CHR rules.

Notice that in a ω_t sequent, compare to ω_l or $\omega_l^{\otimes}$ sequents, the final store is empty. It will be only known at the (unique) leaf of the ω_t proof.

Now we are able to define our ω_t system.

▶ **Definition 17** (ω_t system)**.** The symbol Γ denotes a program, Ω a multi-set of constraints, S, S^K, S^D some sets of identified constraints, A, $K_1, \ldots, K_m$, $D_1, \ldots, D_n$ some constraints, i, $i_1 \ldots, i_{m+n}$ some distinct integers, B a sequence of constraints. The ω_t system is the set of the following ω_t *inference* rules:

- ω_t axiom:

$$\frac{}{\Gamma \blacktriangleright \ \blacktriangleleft S \vdash} \omega_t$$

  with no simpagation rule $(K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow B) \in \Gamma$ such that $S^K = \{K_1\#i_1, \ldots, K_m\#i_m\}$ and $S^D = \{D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}\}$ and $S^K \cup S^D \subseteq S$.
- ω_t-*Tokenize* inference rule:

$$\frac{\Gamma \blacktriangleright \Omega \blacktriangleleft A\#i, S \vdash}{\Gamma \blacktriangleright A, \Omega \blacktriangleleft S \vdash} \#$$

  A usual proviso for quantifier elimination is assumed: i must be a brand new integer.
- ω_t-*Apply* inference rule[10]:

$$\frac{\Gamma \blacktriangleright B, \Omega \blacktriangleleft S^K, S \vdash}{\Gamma \blacktriangleright \Omega \blacktriangleleft S^K, S^D, S \vdash} \backslash \Leftrightarrow$$

  with $(K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow B)$ in Γ and $S^K = \{K_1\#i_1, \ldots, K_m\#i_m\}$ and $S^D = \{D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}\}$.
- ω_t-*true Apply* inference rule:

$$\frac{\Gamma \blacktriangleright \Omega \blacktriangleleft S^K, S \vdash}{\Gamma \blacktriangleright \Omega \blacktriangleleft S^K, S^D, S \vdash} \backslash \Leftrightarrow$$

  with $(K_1, \ldots, K_m \backslash D_1, \ldots, D_n \Leftrightarrow \mathit{true})$ in Γ and $S^K = \{K_1\#i_1, \ldots, K_m\#i_m\}$ and $S^D = \{D_1\#i_{m+1}, \ldots, D_n\#i_{m+n}\}$.

We define also what are a ω_t proof tree and an ω_t proof.

[10] If B is the sequence $B_1, \ldots, B_p$, $p > 0$ then B, Ω means $\{B_1, \ldots, B_p\} \uplus \Omega$.

▶ **Definition 18** (ω_t proof tree and ω_t proof)**.** The set of ω_t *proof trees* is the least set of trees containing all one-node trees labeled with an ω_t sequent, and closed under the rules of Definition 17 in the following sense: For any ω_t proof tree ∇ whose root is labeled with sequent ω_t, s (and whose unique leaf is labeled with sequent s'') and for any instance of an inference rule $\frac{s}{s'}$ of Definition 17, the tree $\frac{\nabla}{s'}$ is an ω_t proof tree whose root is labeled with s' (and whose unique leaf is labeled with s'').

An ω_t *proof* of a sequent s is any ω_t proof tree whose root is labeled with s and whose unique leaf is labeled with an ω_t axiom.

The following lemma expressing the completeness of the ω_t system w.r.t. the ω_t semantics is straightforward.

▶ **Lemma 19** (Completeness of the ω_t system w.r.t. ω_t semantics)**.** *Let Γ be a program, Ω and Ω' two goals, S and S' two stores, c and c' integers such that $c \leq c'$, H and H' two propagation histories such that $H \subseteq H'$.*

If $\langle \Omega, S, H\rangle_c \rightsquigarrow_t^* \langle \Omega', S', H'\rangle_{c'}$ is an ω_t derivation then there exists an ω_t proof tree whose root is $(\Gamma \blacktriangleright \Omega \blacktriangleleft S \vdash\ )$ and such that there is only one sequent leaf $(\Gamma \blacktriangleright \Omega' \blacktriangleleft S' \vdash\ )$.

The following lemma expressing the soundness of the ω_t system w.r.t. the ω_t semantics is a little more difficult since the policy applied to avoid trivial loops has to be maintained.

▶ **Lemma 20** (Soundness of ω_t system w.r.t. ω_t semantics)**.** *Let Γ be a program, Ω and Ω' two multi-sets of constraints, $\Omega_\#$ and $\Omega'_\#$ two multi-sets of identified constraints and H a set of identities of instantiated rules.*

*If $(\Gamma \blacktriangleright \Omega_\# \blacktriangleleft S \vdash\ )$ admits an ω_t proof tree such that there is only one sequent leaf $(\Gamma \blacktriangleright \Omega'_\# \blacktriangleleft S' \vdash\ )$ with no identity of an instantiated rule in the ω_t proof tree appearing twice nor in H, then there exists an ω_t derivation $\langle \Omega, S, H\rangle_i \rightsquigarrow_t^* \langle \Omega', S', H'\rangle_{i'+1}$ with i (resp. i') the integer introduced by the first (last) instance of the ω_t-*Tokenize *inference rule in the ω_t proof tree and H' is the union of H and all the identities of the instantiated rules of the ω_t proof tree.*

The following theorem of completeness and soundness of the ω_t system w.r.t. the ω_t semantics is a direct corollary of the two previous lemmas.

▶ **Theorem 21** (Soundess and completeness of ω_t system w.r.t. ω_t semantics)**.** *Let Γ be a program and Ω an initial goal.*

$\langle \Omega, \emptyset, \emptyset\rangle_1$ admits a successful ω_t derivation if and only if $(\Gamma \blacktriangleright \Omega \blacktriangleleft\ \vdash\ )$ admits an ω_t proof with no identity of instantiated rule appearing twice.

▶ **Lemma 22** (Completeness of ω_l system w.r.t. ω_t system)**.** *Let Γ be a CHR program and $B_1, \ldots, B_p$ some constraints.*

If the ω_t sequent $(\Gamma \blacktriangleright B_1, \ldots, B_p \blacktriangleleft\ \vdash\ )$ admits an ω_t proof with a last sequent $(\Gamma \blacktriangleright\ \blacktriangleleft S \vdash\ )$ then the ω_l sequent $(\Gamma \blacktriangleright B_1\#1, \ldots, B_p\#p \blacktriangleleft\ \vdash\ S)$ admits an ω_l proof.

▶ **Lemma 23** (Soundness of ω_l system w.r.t. ω_t system)**.** *Let Γ be a CHR program and $B_1, \ldots, B_p$ some constraints.*

If the ω_l sequent $(\Gamma \blacktriangleright B_1\#1, \ldots, B_p\#p \blacktriangleleft\ \vdash\ S)$ admits an ω_l proof then the ω_t sequent $(\Gamma \blacktriangleright B_1, \ldots, B_p \blacktriangleleft\ \vdash\ )$ admits an ω_t proof with a last sequent $(\Gamma \blacktriangleright\ \blacktriangleleft S \vdash\ )$.

▶ **Theorem 24** (Completeness of ω_l system w.r.t. ω_t system)**.** *Let Γ be a CHR program and $B_1, \ldots, B_p$ some constraints.*

The ω_t sequent $(\Gamma \blacktriangleright B_1, \ldots, B_p \blacktriangleleft\ \vdash\ )$ admits an ω_t proof with a last sequent $(\Gamma \blacktriangleright\ \blacktriangleleft S \vdash\ )$ if and only if the ω_l sequent $(\Gamma \blacktriangleright B_1\#1, \ldots, B_p\#p \blacktriangleleft\ \vdash\ S)$ admits an ω_l proof.

Proof of Theorem 24. Direct consequence of Lemmas 22 and 23. ◀

Proof of Theorem 7. Direct consequence of Theorems 21 and 24. ◀

Proof of Theorem 8. The soundness is a direct consequence of Theorem 7. ◀