A Complete and Terminating Execution Model for Constraint Handling Rules

Hariolf Betz, Frank Raiser, Thom Frühwirth

Ulmer Informatik-Berichte

Nr. 2010-01
Januar 2010

A Complete and Terminating Execution Model for Constraint Handling Rules

Hariolf Betz, Frank Raiser, and Thom Frühwirth

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

Abstract. We observe that the various formulations of the operational semantics of Constraint Handling Rules proposed over the years fall into a spectrum ranging from the analytical to the pragmatic. While existing analytical formulations facilitate program analysis and formal proofs of program properties, they cannot be implemented as is. We propose a novel operational semantics $\omega_!$, which has a strong analytical foundation, while featuring a terminating execution model. We prove its soundness and completeness with respect to existing analytical formulations and we compare its expressivity to that of various other formulations.

1 Introduction

Constraint Handling Rules [1] (CHR) is a declarative, multiset- and rule-based programming language suitable for concurrent execution and powerful program analysis. While it is known as a language that combines efficiency with declarativity, publications in the field display a tendency to favor one of these aspects over the other. We observe a spectrum of research directions ranging from the *analytical* to the *pragmatic*.

On the analytical end of the spectrum, emphasis is put on CHR as a mathematical formalism, declarativity, and the understanding of its logical foundations and theoretical properties. Several formalizations of the operational semantics, found in [2, 3] and [4], belong to this side of the spectrum. Notable results building on these analytical formalizations include decidable criteria for operational equivalence [5] and confluence [6], a strong foundation of CHR in linear logic [7], as well as weak and strong parallelization, as presented in [8] and further developed toward concurrency in [9, 10].

A recent analytical formalization is the operational semantics ω_e, given in [11]. It consists in a rewriting system of equivalence classes of states based on an axiomatic formulation of equivalence. It has been shown to coincide with the operational semantics ω_{va}, which has been introduced in [1] to set a standard for all other operational semantics to build upon.

On the downside, these operational semantics are detached from practical implementation in that they are oblivious to questions of efficiency and termination. Particularly, the class of rules called *propagation rules* causes trivial non-termination in both of them. Hence, it is safe to say that the existing analytical formalizations of the operational semantics lack a terminating execution model.

This contrasts with most work on the pragmatic side of the spectrum, which emphasizes practical implementation and efficiency over formal reasoning. It originates with [12], where a token-based approach is proposed in order to avoid trivial non-termination: Every propagation rule is applicable only once to a specific combination of constraints. This is realized by keeping a *propagation history* – sometimes called *token store* – in the CHR state. Thus, we gain a terminating execution model for the full segment of CHR.

Building upon [12], a plethora of operational semantics has been brought forth, such as the token-based operational semantics ω_t and its refinement ω_r [13]. The latter reduces non-determinism for a gain in efficiency and sets the current standard for CHR implementations. Another notable exponent is the priority-based operational semantics ω_p [14].

On the downside, token stores break with declarativity: Two states that differ only in their token stores may exhibit different operational behavior while sharing the same logical reading. Therefore, we consider token stores as *non-declarative elements* in CHR states.

Recent work on linear logical algorithms [15] and the close relation of CHR to linear logic [7] suggest a novel approach that emphasizes aspects from both sides of the spectrum to a useful degree: In this work, we introduce the notion of *persistent constraints* to CHR, a concept reminiscent of unrestricted or "banged" propositions in linear logic. Persistent constraints provide a finite representation of the result of any number of propagation rule firings.

We furthermore introduce a state transition system based on persistent constraints, which is explicitly irreflexive. In combination, the two ideas solve the problem of trivial non-termination while retaining declarativity and preserving the potential for effective concurrent execution. This state transition system requires no more than two rules. As every transition step corresponds to a CHR rule application, it facilitates formal reasoning over programs.

In this work, we show that the resulting operational semantics $\omega_!$ is sound and complete with respect to ω_e. We show that $\omega_!$ can be faithfully embedded into the operational semantics ω_p, thus effectively providing an implementation in the form of a source-to-source transformation. All operational semantics developed with an emphasis on pragmatic aspects lack this completeness property. Therefore, this work is the first to show that it is possible to implement CHR soundly and completely with respect to its abstract foundations, whilst featuring a terminating execution model.

Example 1. Consider the following straightforward CHR program for computing the transitive hull of a graph represented by edge constraints $e/2$:

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

This most intuitive formulation of a transitive hull is not a suitable implementation in most existing operational semantics. In fact, for goals containing cyclic graphs it is non-terminating in all aforementioned existing semantics. In this work we show that execution in our proposed semantics $\omega_!$ correctly computes the transitive hull whilst guaranteeing termination.

The remainder of this paper is structured as follows: We summarize the existing operational semantics ω_t, ω_p, and ω_e in Sect. 2. Section 3 then presents our semantics $\omega_!$, which we originally proposed in [16]. In Sect. 4, we prove soundness and completeness of $\omega_!$ with respect to the operational semantics ω_e, thus founding our semantics in the abstract line of CHR research. In Sect. 5, we compare $\omega_!$ to other operational semantics with respect to expressivity and show how to faithfully encode it into ω_p. In Sect. 6, we discuss characteristic properties of $\omega_!$. Related work is investigated in Sect. 6.3, before we conclude in Sect. 7.

2 Preliminaries

We first introduce the syntax of CHR and the equivalence-based operational semantics ω_e, which offers a foundation for all other semantics, although it lacks a terminating execution model. We furthermore present its refinements ω_t and ω_p.

2.1 The Syntax of CHR

Constraint Handling Rules distinguishes two kinds of constraints: *user-defined constraints* (or *CHR constraints*) and *built-in constraints*. Reasoning on built-in constraints is possible through a satisfaction-complete and decidable constraint theory $\mathcal{CT}$.

CHR is a programming language that offers advanced rule-based multiset rewriting. Its eponymous rules are of the form

$$r \;@\; H_1 \backslash H_2 \Leftrightarrow G \mid B_c, B_b$$

where H_1 and H_2 are multisets of user-defined constraints, called the *kept head* and *removed head*, respectively. The *guard* G is a conjunction of built-in constraints and the *body* consists of a conjunction of built-in constraints B_b and a multiset of user-defined constraints B_c. The *rule name* r is optional and may be omitted along with the @ symbol.

In this work, we put special emphasis on the class of rules where $H_2 = \emptyset$, called *propagation rules*. Propagation rules can be written alternatively as

$$r \;@\; H_1 \Rightarrow G \mid B_c, B_b.$$

A *variant* of a rule $(r \;@\; H_1 \backslash H_2 \Leftrightarrow G \mid B_c, B_b)$ with variables $\bar{x}$ is a rule of the form $(r \;@\; H_1 \backslash H_2 \Leftrightarrow G \mid B_c, B_b)[\bar{x}/\bar{y}]$ for any sequence of pairwise distinct variables $\bar{y}$. For any rule $(r \;@\; H_1 \backslash H_2 \Leftrightarrow G \mid B_c, B_b)$, the *local variables* $\bar{l}_r$ are defined as $\bar{l}_r ::= \text{vars}(G, B_c, B_b) \setminus \text{vars}(H_1, H_2)$. A rule where $\bar{l}_r = \emptyset$ is called *range-restricted*.

A CHR program $\mathcal{P}$ is a set of rules. A *range-restricted* CHR program is a set of range-restricted rules.

2.2 Equivalence-based Operational Semantics ω_e

In this section, we recall the *equivalence-based operational semantics* ω_e [11]. It is operationally close to the very abstract semantics ω_{va}, but we prefer it for its concise formulation and the explicit distinction of global variables, CHR-, and built-in constraints.

Definition 1 (ω_e State). *A* ω_e state *is a tuple* $\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$. *The* goal $\mathbb{G}$ *is a multiset of CHR constraints. The* built-in constraint store $\mathbb{B}$ *is a conjunction of built-in constraints.* $\mathbb{V}$ *is a set of variables called the* global variables. *We use* Σ_e *to denote the set of all* ω_e *states.*

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

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

The operational semantics ω_e is founded on equivalence classes of states, based on the following definition of state equivalence.

Definition 3 (ω_e State Equivalence). *Equivalence between* ω_e *states is the smallest equivalence relation* $\equiv_e$ *over* ω_e *states that satisfies the following conditions:*

1. (Equality as Substitution)

$$\langle \mathbb{G}; X \doteq t \wedge \mathbb{B}; \mathbb{V} \rangle \equiv_e \langle \mathbb{G}\,[X/t]\,; X \doteq t \wedge \mathbb{B}; \mathbb{V} \rangle$$

2. (Transformation of the Constraint Store) *If* $\mathcal{CT} \models \exists \bar{s}.\mathbb{B} \leftrightarrow \exists \bar{s}'.\mathbb{B}'$ *where* $\bar{s}, \bar{s}'$ *are the strictly local variables of* $\mathbb{B}, \mathbb{B}'$*, respectively, then:*

$$\langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle \equiv_e \langle \mathbb{G}; \mathbb{B}'; \mathbb{V} \rangle$$

3. (Omission of Non-Occurring Global Variables) *If* X *is a variable that does not occur in* $\mathbb{G}$ *or* $\mathbb{B}$ *then:*

$$\langle \mathbb{G}; \mathbb{B}; \{X\} \cup \mathbb{V} \rangle \equiv_e \langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle$$

4. (Equivalence of Failed States)

$$\langle \mathbb{G}; \bot; \mathbb{V} \rangle \equiv_e \langle \mathbb{G}'; \bot; \mathbb{V} \rangle$$

The following theorem gives a necessary, sufficient, and decidable criterion for equivalence of ω_e states. It has been presented and proven in [11].

Theorem 1 (Criterion for $\equiv_e$). *Let* $\sigma = \langle \mathbb{G}; \mathbb{B}; \mathbb{V} \rangle, \sigma' = \langle \mathbb{G}'; \mathbb{B}'; \mathbb{V} \rangle$ *be* ω_e *states with local variables* $\bar{y}, \bar{y}'$ *that have been renamed apart.*

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

Definition 4 (ω_e Transitions). *For a CHR program $\mathcal{P}$, the state transition system $(\Sigma_e/{\equiv_e}, \rightarrowtail_e)$ is defined as follows. The transition is based on a variant of a rule r in $\mathcal{P}$ such that its local variables are disjoint from the variables occurring in the pre-transition state.*

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

When the rule r is clear from the context or not important, we may write $\rightarrowtail_e$ rather than $\rightarrowtail_e^r$. By $\rightarrowtail_e^*$, we denote the reflexive-transitive closure of $\rightarrowtail_e$.

In the following, we freely mix equivalence classes and their representative, i.e. we often write $\sigma \rightarrowtail_e \tau$ instead of $[\sigma] \rightarrowtail_e [\tau]$.

An inherent problem of ω_e is its behavior with respect to propagation rules: If a state can fire a propagation rule once, it can do so again and again, ad infinitum. In the literature, this problem is referred to as *trivial non-termination* of propagation rules.

2.3 Theoretical Operational Semantics

The theoretical operational semantics ω_t [1, 17] uses a so-called *token store* to avoid trivial non-termination. A propagation rule can only be applied once to each combination of constraints matching the head. Hence, the token store keeps a history of fired propagation rules based on constraint identifiers, as defined below.

Definition 5 (Identified CHR Constraints). *An* identified CHR constraint *$c\#i$ is a CHR constraint c associated with a unique integer i, the* constraint identifier. *We introduce the functions* $\mathrm{chr}(c\#i) = c$ *and* $\mathrm{id}(c\#i) = i$, *and extend them to sequences and sets of identified CHR constraints in the obvious manner.*

The definition of a ω_t state is more complicated, because identified constraints are distinguished from unidentified constraints and the token store is added [1].

Definition 6 (ω_t State). *A ω_t* state *is a tuple of the form $\langle \mathbb{G}; \mathbb{S}; \mathbb{B}; \mathbb{T}\rangle_n^{\mathbb{V}}$ where the* goal (store) *$\mathbb{G}$ is a multiset of constraints, the* CHR (constraint) store *$\mathbb{S}$ is a set of identified CHR constraints, the* built-in (constraint) store *$\mathbb{B}$ is a conjunction of built-in constraints. The* token store *(or* propagation history*) $\mathbb{T}$ is a set of tuples (r, I), where r is the name of a propagation rule and I is an ordered sequence of constraint identifiers. $\mathbb{V}$ is a set of variables called the* global variables*. We use Σ_t to denote the set of all ω_t states.*

The corresponding transition system consists of the following three types of transitions.

Definition 7 (ω_t Transitions). *For a CHR program $\mathcal{P}$, the state transition system $(\Sigma_t, \rightarrowtail_t)$ is defined as follows.*

1. Solve. $\langle\{c\} \uplus \mathbb{G}; \mathbb{S}; \mathbb{B}; \mathbb{T}\rangle_n^{\mathbb{V}} \rightarrowtail_t \langle\mathbb{G}; \mathbb{S}; \mathbb{B}'; \mathbb{T}\rangle_n^{\mathbb{V}}$
where c is a built-in constraint and $\mathcal{CT} \models \forall((c \wedge \mathbb{B}) \leftrightarrow \mathbb{B}')$.
2. Introduce. $\langle\{c\} \uplus \mathbb{G}; \mathbb{S}; \mathbb{B}; \mathbb{T}\rangle_n^{\mathbb{V}} \rightarrowtail_t \langle\mathbb{G}; \{c\#n\} \cup \mathbb{S}; \mathbb{B}; \mathbb{T}\rangle_{n+1}^{\mathbb{V}}$
where c is a CHR constraint.
3. Apply. $\langle\mathbb{G}; H_1 \cup H_2 \cup \mathbb{S}; \mathbb{B}; \mathbb{T}\rangle_n^{\mathbb{V}} \rightarrowtail_t \langle B \uplus \mathbb{G}; H_1 \cup \mathbb{S}; \mathrm{chr}(H_1) = H_1' \wedge \mathrm{chr}(H_2) = H_2' \wedge G \wedge \mathbb{B}; \mathbb{T} \cup \{(r, \mathrm{id}(H_1) + \mathrm{id}(H_2))\}\rangle_n^{\mathbb{V}}$
where r @ $H_1' \setminus H_2' \Leftrightarrow G \mid B$ is a fresh variant of a rule in $\mathcal{P}$ with fresh variables $\bar{x}$ such that $\mathcal{CT} \models \exists(\mathbb{B}) \wedge \forall(\mathbb{B} \rightarrow \exists\bar{x}(\mathrm{chr}(H_1) = H_1' \wedge \mathrm{chr}(H_2) = H_2' \wedge G))$ and $(r, \mathrm{id}(H_1) + \mathrm{id}(H_2)) \notin \mathbb{T}$.

When the rule r is clear from the context or not important, we may write $\rightarrowtail_t$ rather than $\rightarrowtail_t^r$. By $\rightarrowtail_t^*$, we denote the reflexive-transitive closure of $\rightarrowtail_t$.

2.4 Operational Semantics with Rule Priorities

The extension of CHR with rule priorities was initially proposed in [14]. It annotates rules with priorities and modifies the operational semantics such that among the applicable rules, we always select one of highest priority for execution. The operational semantics of this extension is denoted as ω_p and the formulation we use in work was given in [18]. Its state definition coincides with that of ω_t.

Definition 8 (ω_p State). *A ω_p state is a ω_t state. We use Σ_p to denote the set of all ω_p states.*

Definition 9 (ω_p Transitions). *For a CHR program $\mathcal{P}$ with rule priorities, the state transition system $(\Sigma_p, \rightarrowtail_p)$ is defined as follows.*

1. Solve. $\langle\{c\} \uplus \mathbb{G}; \mathbb{S}; \mathbb{B}; \mathbb{T}\rangle_n^{\mathbb{V}} \rightarrowtail_p \langle\mathbb{G}; \mathbb{S}; \mathbb{B}'; \mathbb{T}\rangle_n^{\mathbb{V}}$
where c is a built-in constraint and $\mathcal{CT} \models \forall((c \wedge \mathbb{B}) \leftrightarrow \mathbb{B}')$.
2. Introduce. $\langle\{c\} \uplus \mathbb{G}; \mathbb{S}; \mathbb{B}; \mathbb{T}\rangle_n^{\mathbb{V}} \rightarrowtail_p \langle\mathbb{G}; \{c\#n\} \cup \mathbb{S}; \mathbb{B}; \mathbb{T}\rangle_{n+1}^{\mathbb{V}}$
where c is a CHR constraint.
3. Apply. $\langle\emptyset; H_1 \cup H_2 \cup \mathbb{S}; \mathbb{B}; \mathbb{T}\rangle_n^{\mathbb{V}} \rightarrowtail_p \langle B; H_1 \cup \mathbb{S}; \Theta \wedge \mathbb{B}; \mathbb{T} \cup t\rangle_n^{\mathbb{V}}$ *where $\mathcal{P}$ contains a rule of priority p with fresh variables of the form*

$$p :: r \ @ \ H_1' \setminus H_2' \Leftrightarrow G \mid B$$

and a matching substitution Θ such that $\mathrm{chr}(H_1) = \Theta(H_1')$, $\mathrm{chr}(H_2) = \Theta(H_2')$, $\mathcal{CT} \models \exists(\mathbb{B}) \wedge \forall(\mathbb{B} \rightarrow \bar{\exists}_{\mathbb{B}}(\Theta \wedge G))$, $\Theta(p)$ is a ground arithmetic expression and $t = (r, \mathrm{id}(H_1) + \mathrm{id}(H_2)) \notin \mathbb{T}$. Furthermore, no rule of priority p' and substitution Θ' exists with $\Theta'(p') < \Theta(p)$ for which the above conditions hold.

When the rule r is clear from the context or not important, we may write $\rightarrowtail_p$ rather than $\rightarrowtail_p^r$. By $\rightarrowtail_p^*$, we denote the reflexive-transitive closure of $\rightarrowtail_p$.

3 Operational Semantics with Persistent Constraints $\omega_!$

In this section, we present the operational semantics $\omega_!$ with persistent constraints, proposed in [16]. It is based on the following ideas:

1. In ω_e, the body of a propagation rule can be generated any number of times, provided that the corresponding head constraints are present in the store. In order to give consideration to this theoretical behavior, we introduce those body constraints as so-called *persistent constraints.* A persistent constraint is a finite representation of a large, though unspecified number of identical constraints. For a proper distinction, constraints that are not persistent constraints are henceforth called *linear* constraints.
2. As a secondary consequence, arbitrary generation of rule bodies in ω_e affects other types of CHR rules as well. Consider the following program:

$$\begin{aligned} &\text{r1} \ @\ a \Longrightarrow b \\ &\text{r2} \ @\ b \ \Leftrightarrow \ c \end{aligned}$$

   If executed with a goal a, this program can generate an arbitrary number of constraints of the form b. As a consequence of this, it can also generate arbitrarily many constraints c. To take these indirect consequences of propagation rules into account, we introduce a rule's body constraints as persistent whenever its removed head can be matched completely with persistent constraints.
3. As a persistent constraint represents an arbitrary number of identical constraints, we consider multiple occurrences of a persistent constraint as idempotent. Thus, we implicitly apply a set semantics to persistent constraints.
4. We adapt the execution model such that a transition takes place only if the post-transition state is not equivalent to the pre-transition state. This entails two beneficial consequences: Firstly, in combination with the set semantics on persistent constraints, it avoids trivial non-termination of propagation rules. Secondly, as failed states are equivalent, it enforces termination upon failure.

We adapt the definition of $\omega_!$ states with respect to ω_e. The goal store $\mathbb{G}$ of ω_e states is split into a store $\mathbb{L}$ of linear constraints and a store $\mathbb{P}$ of persistent constraints.

Definition 10 ($\omega_!$ State). *A $\omega_!$* state *is a tuple of the form $\langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle$, where $\mathbb{L}$ and $\mathbb{P}$ are multisets of CHR constraints called the* linear (CHR) store *and* persistent (CHR) store*, respectively. $\mathbb{B}$ is a conjunction of built-in constraints and $\mathbb{V}$ is a set of variables called the* global variables*. We use $\Sigma_!$ to denote the set of all $\omega_!$ states.*

Definition 11 is analogous to ω_e, though adapted to comply with Definition 10.

Definition 11 (Variable Types). *For the variables occurring in a $\omega_!$ state $\sigma = \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle$ we distinguish three different types:*

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

The following definition of state equivalence is adapted to comply with Definition 10 and to handle idempotence of persistent constraints.

Definition 12 (Equivalence of $\omega_!$ States). *Equivalence between $\omega_!$ states is the smallest equivalence relation $\equiv_!$ over $\omega_!$ states that satisfies the following conditions:*

1. (Equality as Substitution) *Let X be a variable, t be a term and $\doteq$ the syntactical equality relation.*
$$\langle \mathbb{L}; \mathbb{P}; X \doteq t \wedge \mathbb{B}; \mathbb{V} \rangle \equiv_! \langle \mathbb{L}\,[X/t]\,; \mathbb{P}\,[X/t]\,; X \doteq t \wedge \mathbb{B}; \mathbb{V} \rangle$$
2. (Transformation of the Constraint Store) *If $\mathcal{CT} \models \exists \bar{s}.\mathbb{B} \leftrightarrow \exists \bar{s}'.\mathbb{B}'$ where $\bar{s}, \bar{s}'$ are the strictly local variables of $\mathbb{B}, \mathbb{B}'$, respectively, then:*
$$\langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle \equiv_! \langle \mathbb{L}; \mathbb{P}; \mathbb{B}'; \mathbb{V} \rangle$$
3. (Omission of Non-Occurring Global Variables) *If X is a variable that does not occur in $\mathbb{L}$, $\mathbb{P}$, or $\mathbb{B}$ then:*
$$\langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \{X\} \cup \mathbb{V} \rangle \equiv_! \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle$$
4. (Equivalence of Failed States)
$$\langle \mathbb{L}; \mathbb{P}; \bot; \mathbb{V} \rangle \equiv_! \langle \mathbb{L}'; \mathbb{P}'; \bot; \mathbb{V}' \rangle$$
5. (Contraction)
$$\langle \mathbb{L}; P \uplus P \uplus \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle \equiv_! \langle \mathbb{L}; P \uplus \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle$$

The following definition presents an auxiliary concept that we use to formulate a criterion for $\omega_!$ equivalence.

Definition 13 ($\bowtie$). *For multisets of constraints $\mathbb{G}, \mathbb{G}'$, the relation $\mathbb{G} \bowtie \mathbb{G}'$ holds iff*
$$(\forall c \in \mathbb{G}.\exists c' \in \mathbb{G}'.c = c') \wedge (\forall c' \in \mathbb{G}'.\exists c \in \mathbb{G}.c = c')$$

The following property follows directly from Definition 13. We quote it as a reference in upcoming proofs:

Property 1 (Properties of $\bowtie$). For multisets of constraints $\mathbb{G}, \mathbb{G}'$, all $n \in \mathbb{N}$
$$\mathbb{G} \bowtie \mathbb{G}' \quad \Rightarrow \quad \exists N \in \mathbb{N}.\mathbb{G} \subseteq N \cdot \mathbb{G}'$$

Theorem 2 (Criterion for $\equiv_!$). *For two $\omega_!$ states $\sigma = \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle, \sigma' = \langle \mathbb{L}'; \mathbb{P}'; \mathbb{B}'; \mathbb{V} \rangle \in \Sigma_!$ with local variables $\bar{y}, \bar{y}'$ that have been renamed apart,*
$$\begin{aligned} \sigma \equiv_! \sigma' \quad \Leftrightarrow \quad \mathcal{CT} \models & \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}') \wedge \mathbb{B}')) \wedge \\ & \forall(\mathbb{B}' \rightarrow \exists \bar{y}.((\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}') \wedge \mathbb{B})) \end{aligned}$$

Proof.

'⇐': We consider two $\omega_!$ states $\sigma = \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle, \sigma' = \langle \mathbb{L}'; \mathbb{P}'; \mathbb{B}'; \mathbb{V} \rangle$ with local variables $\bar{y}$ and $\bar{y}'$. We furthermore assume that:

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

If $\mathcal{CT} \models \neg\exists((\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}'))$, we have $\mathcal{CT} \models \mathbb{B} = \mathbb{B}' = \bot$ such that Def. 12.4 proves $\sigma \equiv_! \sigma'$. In the following, we assume that a matching $(\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}')$ exists.

It follows from $\forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}') \wedge \mathbb{B}'))$ by Def. 12:2 that:

$$\sigma \equiv_! \langle \mathbb{L}; \mathbb{P}; (\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}') \wedge \mathbb{B} \wedge \mathbb{B}'; \mathbb{V} \rangle$$

Def. 12:1 gives us:

$$\sigma \equiv_! \langle \mathbb{L}'; \mathbb{P}''; (\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}') \wedge \mathbb{B} \wedge \mathbb{B}'; \mathbb{V} \rangle$$

where $\mathbb{P}''$ equals $\mathbb{P}'$ modulo multiplicities. By Def. 12:5 we thus get:

$$\sigma \equiv_! \langle \mathbb{L}'; \mathbb{P}'; (\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}') \wedge \mathbb{B} \wedge \mathbb{B}'; \mathbb{V} \rangle$$

From $\forall(\mathbb{B}' \rightarrow \exists \bar{y}.((\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}') \wedge \mathbb{B}))$ follows by Def. 12:2 that:

$$\sigma \equiv_! \langle \mathbb{L}'; \mathbb{P}'; \mathbb{B}'; \mathbb{V} \rangle = \sigma'$$

'⇒': To prove the forward direction, we have to show the compliance of the conditions in Def. 12.1 to Def. 12.5 with our criterion. For Def. 12.1 to Def. 12.4, compliance is analogous to Thm. 1 as proven in [11]. We consider Def. 12.5:

Let $\sigma = \langle \mathbb{L}; P \uplus P \uplus \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle, \sigma' = \langle \mathbb{L}; P \uplus \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle \in \Sigma_!$ with local variables $\bar{y}, \bar{y}'$. As, $(P \uplus P \uplus \mathbb{P}) \bowtie (P \uplus \mathbb{P})$, the following is a tautology:

$$\models \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{L} = \mathbb{L}) \wedge ((P \uplus P \uplus \mathbb{P}) \bowtie (P \uplus \mathbb{P})) \wedge \mathbb{B})) \wedge \\ \forall(\mathbb{B} \rightarrow \exists \bar{y}.((\mathbb{L} = \mathbb{L}) \wedge ((P \uplus P \uplus \mathbb{P}) \bowtie (P \uplus \mathbb{P})) \wedge \mathbb{B}))$$

□

Based on the definition of $\equiv_e$, we define the operational semantics $\omega_!$ below. Since body constraints may be introduced either as linear or as persistent constraints, uniform rule application is replaced by two distinct application modes. Note that $\omega_!$ is only defined for *range-restricted* programs (cf. Sect. 6.2 for details).

Definition 14 ($\omega_!$ Transitions). *For a range-restricted CHR program $\mathcal{P}$, the state transition system $(\Sigma_!/{\equiv_!}, \rightarrowtail_!)$ is defined as follows.*
ApplyLinear:

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

ApplyPersistent:

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

When the rule r is clear from the context or not important, we may write $\rightarrowtail_!$ rather than $\rightarrowtail_!^r$. By $\rightarrowtail_!^*$, we denote the reflexive-transitive closure of $\rightarrowtail_!$.

4 Soundness and Completeness

In this section we show soundness and completeness of $\omega_!$ with respect to ω_e. In Sect. 4.1 and Sect. 4.2 we introduce auxiliary concepts required for our theorems given in Sect. 4.3.

4.1 State Inclusion

State inclusion is a partial-order relation on ω_e states that differ only in their goal stores, modulo $\equiv_e$.

Definition 15 (State inclusion, $\sqsubseteq$). *State inclusion is the smallest partial-order relation $\sqsubseteq \subseteq (\Sigma_e \times \Sigma_e)$ such that for all states $\sigma_e, \sigma_e', \langle \mathbb{G}; \mathbb{B}; \mathbb{V}\rangle \in \Sigma_e$ and all multisets of user-defined constraints $\mathbb{G}'$,*

1. $\sigma_e \equiv_e \sigma_e' \Rightarrow \sigma_e \sqsubseteq \sigma_e'$
2. $\langle \mathbb{G}; \mathbb{B}; \mathbb{V}\rangle \sqsubseteq \langle \mathbb{G} \uplus \mathbb{G}'; \mathbb{B}; \mathbb{V}\rangle$

The following lemmata give a criterion for deciding the state inclusion $\sqsubseteq$ and define its relationship with $\equiv_e$ and $\rightarrowtail_e$.

Lemma 1 (Criterion for State Inclusion). *For ω_e states $\sigma_e = \langle \mathbb{G}; \mathbb{B}; \mathbb{V}\rangle$, $\sigma_e' = \langle \mathbb{G}'; \mathbb{B}'; \mathbb{V}\rangle$ such that their respective local variables $\bar{y}, \bar{y}'$ are disjoint,*

$$\sigma \sqsubseteq \sigma' \quad \Leftrightarrow \quad \mathcal{CT} \models \forall(\mathbb{B}' \rightarrow \exists \bar{y}.((\mathbb{G} \subseteq \mathbb{G}') \wedge \mathbb{B})) \wedge \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{G} \subseteq \mathbb{G}') \wedge \mathbb{B}'))$$

Proof.

'$\Rightarrow$': We can easily show that both criteria given in Def. 15 comply with the criterion.

Def. 15.1 : By Thm. 1, we have that from $\sigma_e \equiv_e \sigma_e'$ follows

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

which in turn implies

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

Def. 15.2 : Let $\sigma_e = \langle \mathbb{G}; \mathbb{B}; \mathbb{V}\rangle, \sigma_e' = \langle \mathbb{G} \uplus \mathbb{G}'; \mathbb{B}; \mathbb{V}\rangle$ be states with local variables $\bar{y}, \bar{y}'$. As $\mathbb{G} \subseteq \mathbb{G} \uplus \mathbb{G}'$ is trivially true, the criterion is reduced to a tautology:

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

'⇐': Let $\sigma_e = \langle \mathbb{G}; \mathbb{B}; \mathbb{V}\rangle, \sigma'_e = \langle \mathbb{G}'; \mathbb{B}'; \mathbb{V}\rangle \in \Sigma_e$ s.t. their respective local variables $\bar{y}, \bar{y}'$ are disjoint and

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

Since $\mathcal{CT} \models \forall(\mathbb{B}' \rightarrow \exists \bar{y}.((\mathbb{G} \subseteq \mathbb{G}') \wedge \mathbb{B}))$, we apply Def. 3.2 to get

$$\sigma'_e \equiv_e \langle \mathbb{G}'; \mathbb{B}' \wedge (\mathbb{G} \subseteq \mathbb{G}') \wedge \mathbb{B}); \mathbb{V}\rangle$$

By Def. 3.1, we get that for some multiset of user-defined constraints $\mathbb{G}''$, we have

$$\sigma'_e \equiv_e \langle \mathbb{G} \uplus \mathbb{G}''; \mathbb{B}' \wedge (\mathbb{G} \subseteq \mathbb{G}') \wedge \mathbb{B}); \mathbb{V}\rangle$$

As $\mathcal{CT} \models \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{G} \subseteq \mathbb{G}') \wedge \mathbb{B}'))$, we can apply Def. 3.2 to show

$$\sigma_e \equiv_e \langle \mathbb{G}; \mathbb{B} \wedge (\mathbb{G} \subseteq \mathbb{G}') \wedge \mathbb{B}'; \mathbb{V}\rangle$$

Finally, we apply Def. 15.2 to obtain

$$\sigma_e \equiv_e \langle \mathbb{G}; \mathbb{B} \wedge (\mathbb{G} \subseteq \mathbb{G}') \wedge \mathbb{B}'; \mathbb{V}\rangle \sqsubseteq \langle \mathbb{G} \uplus \mathbb{G}''; \mathbb{B}' \wedge (\mathbb{G} \subseteq \mathbb{G}') \wedge \mathbb{B}); \mathbb{V}\rangle \equiv_e \sigma'_e$$

□

Lemma 2 (State Inclusion and Equivalence). *For $\sigma_e, \sigma'_e \in \Sigma_e$,*

$$\sigma_e \equiv_e \sigma'_e \quad \textit{iff} \quad \sigma_e \sqsubseteq \sigma'_e \textit{ and } \sigma'_e \sqsubseteq \sigma_e$$

Proof. As the $\equiv_!$ relation is symmetric, the '⇒' direction follows directly from Def. 15. As for the '⇐' direction, assume that $\sigma_e = \langle \mathbb{G}; \mathbb{B}; \mathbb{V}\rangle$ and $\sigma_e = \langle \mathbb{G}'; \mathbb{B}'; \mathbb{V}\rangle$. From the mutual inclusions $\sigma_e \sqsubseteq \sigma'_e$ and $\sigma'_e \sqsubseteq \sigma_e$ follows by Lemma 1 that

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

According to Thm. 1, this implies $\sigma_e = \sigma'_e$. □

Lemma 3 (State Inclusion and Derivation). *For states $\sigma_e, \sigma'_e, \tau_e \in \Sigma_e$ such that $\sigma_e \rightarrowtail_e \tau_e$ and $\sigma_e \sqsubseteq \sigma'_e$, there exists some τ'_e such that $\sigma'_e \rightarrowtail^r_e \tau'_e$ and $\tau_e \sqsubseteq \tau'_e$*

Proof. Let r be of the form r @ $H_1 \setminus H_2 \Leftrightarrow G \mid B_c \uplus B_b$. The derivation $\sigma_e \rightarrowtail^r_e \tau_e$ implies that $\sigma_e \equiv_e \langle H_1 \uplus H_2 \uplus \mathbb{G}; G \uplus \mathbb{B}; \mathbb{V}\rangle$ and $\tau_e \equiv_e \langle H_2 \uplus B_c \uplus \mathbb{G}; G \uplus B_b \uplus \mathbb{B}; \mathbb{V}\rangle$ for some $\mathbb{G}, \mathbb{B}, \mathbb{V}$. As $\sigma_e \sqsubseteq \sigma'_e$, there exists some $\mathbb{G}'$ such that $\sigma_e \equiv_e \langle H_1 \uplus H_2 \uplus \mathbb{G} \uplus \mathbb{G}'; G \uplus \mathbb{B}; \mathbb{V}\rangle$. We choose $\tau'_e = \langle H_2 \uplus B_c \uplus \mathbb{G} \uplus \mathbb{G}'; G \uplus B_b \uplus \mathbb{B}; \mathbb{V}\rangle$. Applying Def. 15 shows that $\tau_e \sqsubseteq \tau'_e$ and $\sigma'_e \rightarrowtail_e \tau'_e$. □

4.2 State Projection

We use the state projection function defined below to relate $\omega_!$ states and ω_e states in the soundness and completeness theorems.

Definition 16 (State Projection). *proj is a function mapping from* $\mathbb{N} \times \Sigma_!$ *to* Σ_e *such that* $proj(N, \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle) = \langle \mathbb{L} \uplus N \cdot \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle$. *We call* $proj(N, \sigma)$ *the* N'th projection *of* σ.

Lemma 4 (Preservation of Equivalence Upon Projection). *For states* $\sigma, \sigma' \in \Sigma_!$,

$$\sigma \equiv_! \sigma' \quad \Rightarrow \quad \forall n \in \mathbb{N}.\exists N \in \mathbb{N}.proj(n, \sigma) \sqsubseteq proj(N, \sigma')$$

Proof. *Let* $\sigma = \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V} \rangle, \sigma' = \langle \mathbb{L}'; \mathbb{P}'; \mathbb{B}'; \mathbb{V}' \rangle$, *it follows that*

$$\begin{aligned} \mathcal{CT} \models & \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}') \wedge \mathbb{B}')) \wedge \\ & \forall(\mathbb{B}' \rightarrow \exists \bar{y}.((\mathbb{L} = \mathbb{L}') \wedge (\mathbb{P} \bowtie \mathbb{P}') \wedge \mathbb{B})) \end{aligned}$$

Due to Prop. 1, this implies

$$\begin{aligned} \mathcal{CT} \models \ & \forall n \in \mathbb{N}.\, \exists N \in \mathbb{N}.\, \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{L} = \mathbb{L}') \wedge (n \cdot \mathbb{P} \subseteq N \cdot \mathbb{P}') \wedge \mathbb{B}')) \wedge \\ & \forall(\mathbb{B}' \rightarrow \exists \bar{y}.((\mathbb{L} = \mathbb{L}') \wedge (n \cdot \mathbb{P} \subseteq N \cdot \mathbb{P}') \wedge \mathbb{B})) \end{aligned}$$

Therefore,

$$\begin{aligned} \mathcal{CT} \models \ & \forall n \in \mathbb{N}.\, \exists N \in \mathbb{N}.\, \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{L} \uplus n \cdot \mathbb{P} \subseteq \mathbb{L}' \uplus N \cdot \mathbb{P}') \wedge \mathbb{B}')) \wedge \\ & \forall(\mathbb{B}' \rightarrow \exists \bar{y}.((\mathbb{L} \uplus n \cdot \mathbb{P} \subseteq \mathbb{L}' \uplus N \cdot \mathbb{P}') \wedge \mathbb{B})) \end{aligned}$$

By Lemma 1, this finally proves

$$\forall n \in \mathbb{N}.\exists N \in \mathbb{N}.proj(n, \sigma) \sqsubseteq proj(N, \sigma')$$

□

Lemma 5 (Analogy between $\equiv_e$ and $\equiv_!$ under Projection). *For* $\tau_!' \in \Sigma_!, \tau_e \in \Sigma_e, n \in \mathbb{N}$,

$$proj(n, \tau_!') \equiv_e \tau_e \quad \Rightarrow \quad \exists \tau_! \in \Sigma_!.\tau_!' \equiv_! \tau_! \wedge proj(n, \tau_!) = \tau_e$$

Proof (sketch). For every axiom α *in Def. 3, there exists a corresponding axiom* α' *in Def. 12, such that if* $proj(n, \sigma_!) \equiv_e proj(n, \tau_!)$ *by axiom* α *then* $\sigma_! \equiv_! \tau_!$ *by axiom* α' *for any* $\sigma_!, \tau_! \in \Sigma_!$. *As* $\equiv_e$ *is the smallest reflexive-transitive relation satisfying its axioms, this proves our lemma.* □

4.3 Soundness and Completeness

The following lemma defines soundness of single-step derivations and directly leads to the soundness theorem given below.

Lemma 6 (Single-Step Soundness). *Let* $\sigma_!, \tau_! \in \Sigma_!$ *and* $n \in \mathbb{N}$. *If* $\sigma_! \rightarrowtail_!^r \tau_!$ *then there exists some* $N \in \mathbb{N}$ *and some* $\tau_e \in \Sigma_e$ *such that* $proj(N, \sigma_!) \rightarrowtail_e^* \tau_e$ *and* $proj(n, \tau_!) \sqsubseteq \tau_e$.

Proof. $\sigma_! \rightarrowtail_!^r \tau_!$ implies a singular application of either **ApplyLinear** or **ApplyPersistent**. We distinguish two cases:

ApplyLinear: We assume the existence of states

$$\sigma_!' = \langle H_1^l \uplus H_2^l \uplus \mathbb{L}; H_1^p \uplus H_2^p \uplus \mathbb{P}; G \wedge \mathbb{B}; \mathbb{V}\rangle$$
$$\tau_!' = \langle H_1^l \uplus B_c \uplus \mathbb{L}; H_1^p \uplus H_2^p \uplus \mathbb{P}; G \wedge \mathbb{B} \wedge B_b; \mathbb{V}\rangle$$

and a fresh variant of a rule r @ $(H_1^l \uplus H_1^p)\backslash(H_2^l \uplus H_2^p) \Leftrightarrow G \mid B_c, B_b$ such that $\sigma_! \equiv_! \sigma_!'$ and $\tau_! \equiv_! \tau_!'$.

By Lemma 4, we have that $\tau_! \equiv_! \tau_!'$ implies the existence of some $k \in \mathbb{N}$ such that $proj(n, \tau_!) \sqsubseteq proj(k, \tau_!')$. We observe that

$$proj(k+1, \sigma_!') = \langle H_1^l \uplus H_2^l \uplus \mathbb{L} \uplus ((k+1) \cdot H_1^p) \uplus ((k+1) \cdot H_2^p) \uplus ((k+1) \cdot \mathbb{P}); G \wedge \mathbb{B}; \mathbb{V}\rangle$$

and $proj(k, \tau_!') = \langle H_1^l \uplus B_c \uplus \mathbb{L} \uplus (m \cdot H_1^p) \uplus (k \cdot H_2^p) \uplus (k \cdot \mathbb{P})); G \wedge \mathbb{B} \wedge B_b; \mathbb{V}\rangle$.

According to the definition of ω_e, we have $proj(k+1, \sigma_!') \rightarrowtail_e \tau_e'$ for

$$\tau_e' = \langle H_1^l \uplus B_c \uplus \mathbb{L} \uplus ((k+1) \cdot H_1^p) \uplus (k \cdot H_2^p) \uplus ((k+1) \cdot \mathbb{P})); G \wedge \mathbb{B} \wedge B_b; \mathbb{V}\rangle.$$

We observe that $proj(k, \tau_!') \sqsubseteq \tau_e'$.

Lemma 4 furthermore implies that for some $N \in \mathbb{N}$, $proj(k+1, \sigma_!') \sqsubseteq proj(N, \sigma_!)$. According to Lemma 3, a $\tau_e \in \Sigma_e$ exists such that $proj(N, \sigma_!) \rightarrowtail_e \tau_e$ with $\tau_e' \sqsubseteq \tau_e$. Transitivity of the $\sqsubseteq$ relation finally proves $proj(n, \tau_!) \sqsubseteq \tau_e$.

ApplyPersistent: We assume the existence of states

$$\sigma_!' = \langle H_1^l \uplus \mathbb{L}; H_1^p \uplus H_2^p \uplus \mathbb{P}; G \wedge \mathbb{B}; \mathbb{V}\rangle$$
$$\tau_!' = \langle H_1^l \uplus \mathbb{L}; H_1^p \uplus H_2^p \uplus B_c \uplus \mathbb{P}; G \wedge \mathbb{B} \wedge B_b; \mathbb{V}\rangle$$

and a fresh variant of a rule r @ $(H_1^l \uplus H_1^p)\backslash H_2^p \Leftrightarrow G \mid B_c, B_b$ such that $\sigma_! \equiv_! \sigma_!'$ and $\tau_! \equiv_! \tau_!'$.

According to Lemma 4, for some $k \in \mathbb{N}$ we have $proj(n, \tau_!) \sqsubseteq proj(k, \tau_!')$. We observe that

$$proj(2k, \sigma_!') = \langle H_1^l \uplus \mathbb{L} \uplus (2k \cdot H_1^p) \uplus (2k \cdot H_2^p) \uplus (2k \cdot \mathbb{P}); G \wedge \mathbb{B}; \mathbb{V}\rangle \text{ and}$$
$$proj(k, \tau_!') = \langle H_1^l \uplus \mathbb{L} \uplus (k \cdot H_1^p) \uplus (k \cdot H_2^p) \uplus (k \cdot B_c) \uplus (k \cdot \mathbb{P})); G \wedge \mathbb{B} \wedge B_b; \mathbb{V}\rangle.$$

For every $i \in \mathbb{N}$, let $\tau_e^i = \langle H_1^l \uplus \mathbb{L} \uplus (2k \cdot H_1^p) \uplus ((2k - i) \cdot H_2^p) \uplus (i \cdot B_c) \uplus (2k \cdot \mathbb{P}); G \wedge \mathbb{B}; \mathbb{V}\rangle$. According to the definition of ω_e, we have $proj(2k, \sigma_!') \rightarrowtail_e \tau_e^1 \rightarrowtail_e \ldots \rightarrowtail_e \tau_e^k$.

We observe that $\tau_e^k = \langle H_1^l \uplus \mathbb{L} \uplus (2k \cdot H_1^p) \uplus (k \cdot H_2^p) \uplus (k \cdot B_c) \uplus (2k \cdot \mathbb{P}); G \wedge \mathbb{B}; \mathbb{V}\rangle$, and therefore $proj(k, \tau_!') \sqsubseteq \tau_e^k$. By Lemma 4 and Lemma 3, we have that for some $N \in \mathbb{N}$ and some $\tau_e \in \Sigma_e$, we have $proj(2k, \sigma_!') \sqsubseteq proj(N, \sigma_!)$ and $proj(N, \sigma_!) \rightarrowtail_e \tau_e$ such that $\tau_e^k \sqsubseteq \tau_e$. Transitivity of $\sqsubseteq$ proves the hypothesis. □

Theorem 3 (Soundness). *Let $\sigma_! = \langle \mathbb{G}; \emptyset; \mathbb{B}; \mathbb{V}\rangle, \tau_! \in \Sigma_!$ be $\omega_!$ states. If $\sigma_! \rightarrowtail_!^* \tau_!$, then for every $N \in \mathbb{N}$ there exists a τ_e such that $proj(0, \sigma_!) \rightarrowtail_e^* \tau_e$ and $proj(N, \tau_!) \sqsubseteq \tau_e$.*

Proof. *Transitive application of Lemma 6 proves that there exists some $n \in \mathbb{N}$ such that $proj(n, \sigma_!) \rightarrowtail_e^* \tau_e$ and $proj(N, \tau_!) \sqsubseteq \tau_e$. We furthermore observe that the empty persistent store of $\sigma_!$ implies for any $n \in \mathbb{N}$, we have $proj(n, \sigma_!) = proj(0, \sigma_!)$.* □

Similar to the soundness case, we first give a lemma stating completeness of single-step derivations that immediately entails our completeness theorem given below.

Lemma 7 (Single-Step Completeness). *Let $\sigma_! \in \Sigma_!, \sigma_e, \tau_e \in \Sigma_e$ such that $\sigma_e \rightarrowtail_e^r \tau_e$.*

1. *If $\sigma_e \equiv_e proj(1, \sigma_!)$, there exists some state $\tau_! \in \Sigma_!$ such that $\sigma_! \rightarrowtail_!^r \tau_!$ or $\sigma_! \equiv_! \tau_!$ and $\tau_e \sqsubseteq proj(1, \tau_!)$.*
2. *If $\sigma_e \equiv_e proj(0, \sigma_!)$, there exists some state $\tau_! \in \Sigma_!$ such that $\sigma_! \rightarrowtail_!^r \tau_!$ or $\sigma_! \equiv_! \tau_!$ and $proj(0, \tau_!) \sqsubseteq \tau_e$.*
3. *If $proj(0, \sigma_!) \sqsubseteq \sigma_e \sqsubseteq proj(1, \sigma_!)$, then there exists some state $\tau_! \in \Sigma_!$ such that $\sigma_! \rightarrowtail_!^r \tau_!$ or $\sigma_! \equiv_! \tau_!$ and $proj(0, \tau_!) \sqsubseteq \tau_e \sqsubseteq proj(1, \tau_!)$.*

Proof. *By Def. 4, $\sigma_e \rightarrowtail_e^r \tau_e$ implies that r is of the form $r \ @ \ H_1 \backslash H_2 \Leftrightarrow G \mid B_c \uplus B_b$, such that $\sigma_e \equiv_e \langle H_1 \uplus H_2 \uplus \mathbb{G}; G \wedge \mathbb{B}; \mathbb{V}\rangle$ and $\tau_e \equiv_e \langle H_1 \uplus B_c \uplus \mathbb{G}; G \wedge B_b \wedge \mathbb{B}; \mathbb{V}\rangle$.*

1. *By Lemma 5, we have that $\sigma_! \equiv_! \sigma_!'$ for some $\sigma_!' \in \Sigma_!$ such that*

$$proj(1, \sigma_!') = \langle H_1 \uplus H_2 \uplus \mathbb{G}; G \wedge \mathbb{B}; \mathbb{V}\rangle.$$

According to Def. 16, $\sigma_!'$ is thus of the form

$$\sigma_!' = \langle H_1^l \uplus H_2^l \uplus \mathbb{L}; H_1^p \uplus H_2^p \uplus \mathbb{P}; G \wedge \mathbb{B}; \mathbb{V}\rangle$$

where $H_1^l \uplus H_1^p = H_1$ and $H_2^l \uplus H_2^p = H_2$ and $\mathbb{L} \uplus \mathbb{P} = \mathbb{G}$. If $H_2^p \neq \emptyset$, we choose

$$\tau_! = \langle H_1^l \uplus B_c \uplus \mathbb{L}; H_1^p \uplus H_2^p \uplus \mathbb{P}; G \wedge \mathbb{B} \wedge B_b; \mathbb{V}\rangle,$$

otherwise we choose

$$\tau_! = \langle H_1^l \uplus \mathbb{L}; H_1^p \uplus H_2^p \uplus B_c \uplus \mathbb{P}; G \wedge \mathbb{B} \wedge B_b; \mathbb{V}\rangle.$$

In both cases, we have $proj(1, \tau_!) = \langle H_1 \uplus H_2^p \uplus B_c \uplus \mathbb{G}; G \wedge B_b \wedge \mathbb{B}; \mathbb{V}\rangle$ and therefore $\tau_e \sqsubseteq proj(1, \tau_!)$. By Def. 14 either $\sigma_!' \rightarrowtail_!^r \tau_!$ or $\sigma_!' \equiv_! \tau_!$.

2. *By Lemma 5, we have that $\sigma_! \equiv_! \sigma_!'$ for some $\sigma_!' \in \Sigma_!$ such that*

$$proj(0, \sigma_!') = \langle H_1 \uplus H_2 \uplus \mathbb{G}; G \wedge \mathbb{B}; \mathbb{V}\rangle.$$

According to Def. 16, $\sigma_!'$ is thus of the form

$$\sigma_!' = \langle H_1 \uplus H_2 \uplus \mathbb{G}; \mathbb{P}; G \wedge \mathbb{B}; \mathbb{V}\rangle$$

for some $\mathbb{P}$. If $H_2 \neq \emptyset$, we choose

$$\tau_! = \langle H_1 \uplus B_c \uplus \mathbb{G}; \mathbb{P}; G \wedge \mathbb{B} \wedge B_b; \mathbb{V}\rangle,$$

otherwise we choose

$$\tau_! = \langle H_1 \uplus \mathbb{G}; B_c \uplus \mathbb{P}; G \wedge \mathbb{B} \wedge B_b; \mathbb{V}\rangle.$$

In both cases, we have $proj(0, \tau_!) \sqsubseteq \tau_e$. *By Def. 14 either* $\sigma'_! \rightarrowtail^r_! \tau_!$ *or* $\sigma'_! \equiv_! \tau_!$.

3. *This follows from Lemma 7.1 and Lemma 7.2 in combination with Lemma 3.* □

Theorem 4 (Completeness). *Let* $\langle \mathbb{G}; \mathbb{B}; \mathbb{V}\rangle, \langle \mathbb{G}'; \mathbb{B}'; \mathbb{V}\rangle \in \Sigma_e$. *If* $\langle \mathbb{G}; \mathbb{B}; \mathbb{V}\rangle \rightarrowtail^*_e \langle \mathbb{G}'; \mathbb{B}'; \mathbb{V}\rangle$, *then there exists some state* $\sigma_! \in \Sigma_!$ *such that* $\langle \mathbb{G}; \emptyset; \mathbb{B}; \mathbb{V}\rangle \rightarrowtail^*_! \sigma_!$ *and* $proj(0, \sigma_!) \sqsubseteq \langle \mathbb{G}'; \mathbb{B}'; \mathbb{V}\rangle \sqsubseteq proj(1, \sigma_!)$.

Proof. *We observe that* $proj(0, \langle \mathbb{G}; \emptyset; \mathbb{B}; \mathbb{V}\rangle) = \langle \mathbb{G}; \mathbb{B}; \mathbb{V}\rangle = proj(1, \langle \mathbb{G}; \emptyset; \mathbb{B}; \mathbb{V}\rangle)$ *and therefore* $proj(0, \langle \mathbb{G}; \emptyset; \mathbb{B}; \mathbb{V}\rangle) \sqsubseteq \langle \mathbb{G}; \mathbb{B}; \mathbb{V}\rangle \sqsubseteq proj(1, \langle \mathbb{G}; \emptyset; \mathbb{B}; \mathbb{V}\rangle)$. *Thus the theorem is a consequence of Lemma 7.3.* □

5 Expressivity

In this section we compare expressivity of the operational semantics $\omega_e, \omega_t, \omega_p$, and $\omega_!$. As all these are Turing-complete [19], expressivity is compared in the literature via the concept acceptable encoding. This concept originates from Shapiro [20] and was first applied to CHR in [21]. It relies on the notion of *answer* defined below.

In order to distinguish linear and persistent constraints when considering goals, we introduce for each CHR constraint symbol c/n, denoting a linear constraint, a corresponding fresh symbol $!c/n$, denoting a persistent constraint. For a multiset $M = \{c_1(\bar{t}_1), \dots, c_n(\bar{t}_n)\}$ let $!M = \{!c_1(\bar{t}_1), \dots, !c_n(\bar{t}_n)\}$.

In the literature answers are usually defined as logical formulas, expressing the declarative reading of a final state. We found it more suitable to define them as ω_e states for two reasons: Firstly, unlike logical formulas, ω_e states are aware of multiplicities of constraints. Secondly, ω_e states enable us to exploit $\equiv_e$ when comparing answers.

Definition 17 (Answers). *Let* $G \wedge B$ *be a goal with CHR constraints* G *and built-in constraints* B. *Then the set of equivalence classes of* ω_e *states* $\mathcal{A}_\mathcal{P}(G \wedge B)$ *for a program* $\mathcal{P}$ *is called the (set of)* answers *and is defined as follows:*

- *for* ω_e: $\mathcal{A}^e_\mathcal{P}(G \wedge B) = \{\tau \mid \langle G; B; \mathrm{vars}(G \wedge B)\rangle \rightarrowtail^*_e \tau \not\rightarrowtail_e\}/{\equiv_e}$
- *for* ω_t: $\mathcal{A}^t_\mathcal{P}(G \wedge B) = \{\langle \mathrm{chr}(\mathbb{G}); \mathbb{B}; \mathrm{vars}(G \wedge B)\rangle \mid \langle G, B; \emptyset; \top; \emptyset\rangle_0^{\mathrm{vars}(G \wedge B)} \rightarrowtail^*_t \langle \emptyset; \mathbb{G}; \mathbb{B}; T\rangle_n^{\mathrm{vars}(G \wedge B)} \not\rightarrowtail_t\}/{\equiv_e}$
- *for* ω_p: $\mathcal{A}^p$ *is defined analogously to* $\mathcal{A}^t$.
- *for* $\omega_!$: $\mathcal{A}^!_\mathcal{P}(G \wedge B) = \{\langle \mathbb{L} \wedge !\mathbb{P}; \mathbb{B}; \mathrm{vars}(G \wedge B)\rangle \mid G = L \uplus !P, \langle L; P; B; \mathrm{vars}(G \wedge B)\rangle \rightarrowtail^*_! \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathrm{vars}(G \wedge B)\rangle \not\rightarrowtail_!\}/{\equiv_e}$

The following definition is based on Gabbrielli's definition of acceptable encoding [21] for CHR operational semantics.

Definition 18 (Acceptable Encoding).
Let ω_1, ω_2 be two operational semantics, $\mathcal{P}_i$ the set of all ω_i programs, and $\mathcal{G}_i$ the set of all ω_i goals for $i = 1, 2$. An acceptable encoding *of ω_1 into ω_2 is a pair of mappings $[\![\ ]\!] : \mathcal{P}_1 \to \mathcal{P}_2$ and $[\![\ ]\!]_g : \mathcal{G}_1 \to \mathcal{G}_2$ which satisfy the following conditions:*

- *$\mathcal{P}_1$ and $\mathcal{P}_2$ share the same constraint theory $\mathcal{CT}$;*
- *for any goal $(A \wedge B) \in \mathcal{G}_1$, $[\![A \wedge B]\!]_g = [\![A]\!]_g \wedge [\![B]\!]_g$. We also assume that the built-ins present in the goal are left unchanged;*
- *Answers are preserved, that is, for all $G \in \mathcal{G}_1$ and $\mathcal{P} \in \mathcal{P}_1$, $\mathcal{A}^2_{[\![\mathcal{P}]\!]}([\![G]\!]_g) = [\![\mathcal{A}^1_{\mathcal{P}}(G)]\!]_g$ holds.*

Figure 1 orders the different operational semantics by expressivity. As shown in [21], there exists an acceptable encoding to embed ω_t into ω_p, but not vice versa. Thus, ω_p is strictly more expressive than ω_t, denoted by the corresponding arrow in Fig. 1. In this work we furthermore show that ω_p is strictly more expressive than $\omega_!$ and that ω_e is strictly less expressive than both ω_t and $\omega_!$.

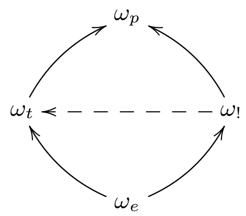

Fig. 1. Acceptable encodings between different operational semantics

Concerning the embedding of ω_e into $\omega_!$, we assume range-restricted programs only. Concerning the acceptable encodings of $\omega_!$ into ω_t and ω_p, we require that the respective programs do not contain pathological rules, according to the following definition.

Definition 19 (Pathological Rules). *A CHR rule*

$$r \ @ \ H_1 \backslash H_2 \Leftrightarrow G \mid B_c, B_b$$

is called pathological *if and only if*

$$\exists \mathbb{B}. \langle H_2; \mathbb{B} \wedge G; \emptyset \rangle \equiv_e \langle B_c; B_b; \emptyset \rangle$$

It is called trivially pathological *iff $\mathbb{B} = \top$. A CHR program $\mathcal{P}$ is called pathological if it contains at least one pathological rule.*

The range-restriction requirement on ω_e programs is due to the fact that Definition 14 for $\omega_!$ is only defined on range-restricted programs. The restriction to non-pathological programs for embeddings of $\omega_!$ into ω_t and ω_p ensures **ApplyLinear** transitions never fail due to irreflexivity, according to the following Lemma.

Concerning the relationship of ω_t and $\omega_!$, we found that no acceptable encoding of ω_t into $\omega_!$ exists. We did find an acceptable encoding of $\omega_!$ into ω_t. However, a thus encoded program might exhibit a different termination behavior from the original $\omega_!$ program (cf. Example 2), as visualized by the dashed arrow in Fig. 1. We currently do not know whether an acceptable encoding without that limitation exists.

The definition of pathological rules is chosen such as to coincide with those rules that cause redundant rule applications – modulo state equivalence – in ω_e.

Lemma 8. *Let $\mathcal{P}$ be a non-pathological CHR program. Then for all ω_e states $\sigma, \tau \in \Sigma_e$ where $\sigma \rightarrowtail_e \tau$, we have $\sigma \not\equiv_e \tau$.*

Proof. We first show a property of Def. 19: Let $\langle H_2; \mathbb{B} \wedge G; \emptyset\rangle \equiv_e \langle B_c; B_b; \emptyset\rangle$, w.l.o.g. let the respective local variables $\bar{y}, \bar{y}'$ be renamed apart. Then by Thm. 1:

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

This is logically equivalent to

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

Therefore, again by Thm. 1, we have that

$$\langle H_2; G \wedge \mathbb{B}; \emptyset\rangle \equiv_e \langle B_c; B_b; \emptyset\rangle \equiv_e \langle B_c; B_b \wedge G \wedge \mathbb{B}; \emptyset\rangle$$

Now let r be a rule $r \ @\ H_1 \setminus H_2 \Leftrightarrow G \mid B_c, B_b$ such that $\sigma \rightarrowtail_e^r \tau$. It follows that $\sigma \equiv_e \langle H_1 \uplus H_2 \uplus \mathbb{C}; G \wedge \mathbb{B}; \mathbb{V}\rangle$ and $\tau \equiv_e \langle B_c \uplus H_1 \uplus \mathbb{C}; B_b \wedge G \wedge \mathbb{B}; \mathbb{V}\rangle$.

Assume that $\sigma \equiv_e \tau$. As H_1 and $\mathbb{C}$ occur in both states, the corresponding states with those multisets removed are also equivalent. Similarly, the same states with $\emptyset$ instead of $\mathbb{V}$ for global variables are equivalent. Therefore,

$$\langle H_2; G \wedge \mathbb{B}; \emptyset\rangle \equiv_e \langle B_c; B_b \wedge G \wedge \mathbb{B}; \emptyset\rangle$$

This implies that there exists a $\mathbb{B}$ according to Def. 19, which is a contradiction to the program being non-pathological. Hence, $\sigma \not\equiv_e \tau$. □

Lemma 9 **($\omega_t \rightarrow \omega_p$).** *There exists an acceptable encoding of ω_t into ω_p.*

Proof (Sketch). All rules have priority 1. □

Lemma 10 **($\omega_p \not\rightarrow \omega_t$).** *There exists no acceptable encoding of ω_p into ω_t.*

Proof. Follows directly from [21]. □

Lemma 11 **($\omega_e \rightarrow \omega_t$).** *There exists an acceptable encoding of ω_e into ω_t.*

Proof (Sketch). *Replace propagation rules with simplification rules that contain a copy of the head in their bodies.* □

Lemma 12 **($\omega_t \not\to \omega_e$).** *There exists no acceptable encoding of ω_t into ω_e.*

Proof. *For any program $\mathcal{P}'$ if $(G' \wedge B') \in \mathcal{A}^m_{\mathcal{P}'}(G)$, no rule in $\mathcal{P}'$ is applicable to $\langle G'; B'; var(G)\rangle$. As global variables do not affect rule application, we have $\mathcal{A}^m_{\mathcal{P}'}(G' \wedge B') \ni (G' \wedge B')$. Consider the ω_t program $\mathcal{P} = (a \Rightarrow b)$. Since $\mathcal{A}^t_{\mathcal{P}}(a) = \{a \wedge b\}$ and $\mathcal{A}^t_{\mathcal{P}}(a \wedge b) = \{a \wedge b \wedge b\}$, an acceptable encoding has to satisfy $\mathcal{A}^m_{[\![\mathcal{P}]\!]}([\![a]\!]_g) = \{[\![a \wedge b]\!]_g\}$ and $\mathcal{A}^m_{[\![\mathcal{P}]\!]}([\![a \wedge b]\!]_g) = \{[\![a \wedge b \wedge b]\!]_g\} = \{[\![a]\!]_g \wedge [\![b]\!]_g \wedge [\![b]\!]_g\} \neq \{[\![a]\!]_g \wedge [\![b]\!]_g\} = \{[\![a \wedge b]\!]_g\}$ which contradicts our earlier observation.* □

Lemma 13 **($\omega_t \not\to \omega_!$).** *There exists no acceptable encoding of ω_t into $\omega_!$.*

Proof. *For any program $\mathcal{P}'$ if $(L' \wedge G' \wedge B') \in \mathcal{A}^!_{\mathcal{P}'}(G)$, no rule in $\mathcal{P}'$ is applicable to $\langle L'; G'; B'; var(G)\rangle$. As global variables do not affect rule application, we also have $\mathcal{A}^!_{\mathcal{P}'}(L' \wedge G' \wedge B') \ni (L' \wedge G' \wedge B')$. Consider the ω_t program $\mathcal{P} = (a \Rightarrow b)$. Since $\mathcal{A}^t_{\mathcal{P}}(a) = \{a \wedge b\}$ and $\mathcal{A}^t_{\mathcal{P}}(a \wedge b) = \{a \wedge b \wedge b\}$, an acceptable encoding has to satisfy $\mathcal{A}^!_{[\![\mathcal{P}]\!]}([\![a]\!]_g) = \{[\![a \wedge b]\!]_g\}$ and $\mathcal{A}^!_{[\![\mathcal{P}]\!]}([\![a \wedge b]\!]_g) = \{[\![a \wedge b \wedge b]\!]_g\} = \{[\![a]\!]_g \wedge [\![b]\!]_g \wedge [\![b]\!]_g\} \neq \{[\![a]\!]_g \wedge [\![b]\!]_g\} = \{[\![a \wedge b]\!]_g\}$ which contradicts our earlier observation.* □

Lemma 14 **($\omega_! \to \omega_t$).** *There exists an acceptable encoding of $\omega_!$ into ω_t.*

Proof. *We show how to encode any $\omega_!$ program $\mathcal{P}$ in ω_t. For every n-ary constraint c/n in $\mathcal{P}$, there exists an $(n+1)$-ary constraint $c/n+1$ in the encoding. In the following, for a multiset $M = \{c_1(\bar{t}_1),\ldots,c_n(\bar{t}_n)\}$ of user-defined $\omega_!$-constraints let*

$$l(M) = \{c_1(l, \bar{t}_1), \ldots, c_n(l, \bar{t}_n)\} \text{ and } p(M) = \{c_1(p, \bar{t}_1), \ldots, c_n(p, \bar{t}_n)\}.$$

The encoded program $[\![\mathcal{P}]\!]$ is constructed as follows:

1. *For every rule $r\ @\ H_1 \setminus H_2 \Leftrightarrow G \mid B$ in $\mathcal{P}$, and all multisets $H_1^l, H_1^p, H_2^l, H_2^p$ s.t. $H_1^l \uplus H_1^p = H_1$ and $H_2^l \uplus H_2^p = H_2$ and $H_2^l \neq \emptyset$, the following rule is in $[\![\mathcal{P}]\!]$:*
$$l(H_1^l) \uplus p(H_1^p) \uplus p(H_2^p) \setminus l(H_2^l) \Leftrightarrow G \mid l(B_c), B_b$$
2. *For every rule $r\ @\ H_1 \setminus H_2 \Leftrightarrow G \mid B_c, B_b$ in $\mathcal{P}$, and all multisets H_1^l, H_1^p s.t. $H_1^l \uplus H_1^p = H_1$, the following rule is in $[\![\mathcal{P}]\!]$:*
$$l(H_1^l) \uplus p(H_1^p) \uplus p(H_2) \Rightarrow G \mid p(B_c), B_b$$
3. *For every rule $\{c(p, \bar{t}), c(p, \bar{t}')\} \uplus H_1 \setminus H_2 \Leftrightarrow G \mid B$ in $[\![\mathcal{P}]\!]$, add also the following rule:*
$$\{c(p, \bar{t})\} \uplus H_1 \setminus H_2 \Leftrightarrow \bar{t} = \bar{t}' \wedge G \mid B$$
4. *For every user-defined constraint declaration c_n in $\mathcal{P}$, there is a rule*
$$c(p, \bar{t}) \setminus c(p, \bar{t}) \Leftrightarrow \top$$

The translation of goals is defined as:

$$[\![\mathbb{L} \wedge !\mathbb{P}]\!]_g ::= l(\mathbb{L}) \wedge p(\mathbb{P})$$

Soundness: *Let $\mathcal{S}_!$ be a function mapping from ω_t states to $\Sigma_!$ such that for: $\sigma_t = \langle l(\mathbb{L}) \uplus p(\mathbb{P}) \uplus \mathbb{B}'; \mathbb{C}; \mathbb{B}; \mathbb{T}\rangle_k^{\mathbb{V}}$ where $chr(\mathbb{C}) = l(\mathbb{L}') \uplus p(\mathbb{P}')$ for some $\mathbb{L}', \mathbb{P}'$,*

$$\mathcal{S}_!(\sigma_t) ::= \langle \mathbb{L} \uplus \mathbb{L}'; \mathbb{P} \uplus \mathbb{P}'; \mathbb{B} \wedge \mathbb{B}'; \mathbb{V}\rangle$$

In the following, we will show that for all $\sigma_t, \tau_t \in \Sigma_t$, $\sigma_t \rightarrowtail_t^* \tau_t$ implies $\mathcal{S}_!(\sigma_t) \rightarrowtail_!^* \mathcal{S}_!(\tau_t)$.

*It is clear from the definition that both the **Introduce** and **Solve** transitions of ω_t are invariant to the $\mathcal{S}_!$ function. Concerning **Apply**, we proceed stepwise w.r.t. the application of the four types of rules present in the encoding $[\![\mathcal{P}]\!]$.*

*1. The rules introduced in construction step 1 represent **ApplyLinear** transitions in $\mathcal{P}$.*

Let r be a variant of a rule $l(H_1^l) \uplus p(H_1^p) \uplus p(H_2^p) \setminus l(H_2^l) \Leftrightarrow G \mid l(B_c) \uplus B_b$ in $[\![\mathcal{P}]\!]$ with fresh variables $\bar{y}$. By definition of the encoding, r has a corresponding rule r' @ $H_1^l \uplus H_1^p \setminus H_2^l \uplus H_2^p \Leftrightarrow G \mid B_c \uplus B_b$ in $\mathcal{P}$. We assume w.l.o.g. that the goal store of σ_t is empty. Hence let $\sigma_t = \langle \emptyset; l(\mathbb{L}) \uplus p(\mathbb{P}); \mathbb{B}; \mathbb{T}\rangle_k^{\mathbb{V}}$ and assume that $\sigma_t \rightarrowtail_t^r \tau_t$. From Def. 7 follows that $\mathcal{CT} \models \forall(\mathbb{B} \rightarrow \exists \bar{y}.(l(H_1^l) \uplus l(H_2^l) \uplus l(\mathbb{L}') = l(\mathbb{L}) \wedge p(H_1^p) \uplus p(H_2^p) \uplus p(\mathbb{P}') = \mathbb{P} \wedge G))$ for some $\mathbb{L}', \mathbb{P}'$. Hence we can show that $\mathcal{S}_!(\sigma_t) \equiv_! \langle H_1^l \uplus H_2^l \uplus \mathbb{L}'; H_1^p \uplus H_2^p \uplus \mathbb{P}'; (H_1^l \uplus H_2^l \uplus \mathbb{L}' = \mathbb{L}) \wedge (H_1^p \uplus H_2^p \uplus \mathbb{P}' = \mathbb{P}) \wedge G \wedge \mathbb{B}; \mathbb{V}\rangle$. Using Def. 7 and Def. 14, we can now show that $\mathcal{S}_!(\sigma_t) \rightarrowtail_!^{r'} \mathcal{S}_!(\tau_t)$ or $\mathcal{S}_!(\sigma_t) \equiv_! \mathcal{S}_!(\tau_t)$.

*2. The rules introduced in step 2 represent **ApplyPersistent** transitions. Analogously to step 1, we show that $\sigma_t \rightarrowtail_t^r \tau_t$ implies $\mathcal{S}_!(\sigma_t) \rightarrowtail_!^{r'} \mathcal{S}_!(\tau_t)$ for some rule $r' \in \mathcal{P}$ or $\mathcal{S}_!(\sigma_t) \equiv_! \mathcal{S}_!(\tau_t)$.*

*3. Step 3 introduces further rules for both **ApplyLinear** and **ApplyPersistent** transitions where a single persistent constraint in the store matches with several head constraints.*

For example, the state $\langle \emptyset; c(0); \top; \emptyset\rangle$ applies to the rule $c(X), c(Y) \Leftrightarrow d(X,Y)$ in $\omega_!$, since $\langle \emptyset; c(0); \top; \emptyset\rangle \equiv_! \langle \emptyset; c(0), c(0); \top; \emptyset\rangle$. Step 2 of the embedding introduces the rule $c(p,X), c(p,Y) \Leftrightarrow d(p,X,Y)$ and step 3 furthermore introduces $c(p,X) \Leftrightarrow X = Y | d(p,X,Y)$ which matches with the ω_t state $\langle \emptyset; c(p,0); \top; \emptyset\rangle_k^{\mathbb{V}}$. Note that the strengthening of the guard might result in a redundant rule: For the rule $c(X), c(Y) \Leftrightarrow X > Y \mid d(X,Y)$, the rule $c(p,X) \Leftrightarrow \bot | d(p,X,Y)$ is introduced which cannot be fired by definition.

To proof soundness, let $\sigma = \langle \mathbb{L}; \{c(p,\bar{t}), c(p,\bar{t}')\} \uplus \mathbb{P}; \mathbb{B}; \mathbb{V}\rangle$ and let $\sigma' = \langle \mathbb{L}; \{c(p,\bar{t})\} \uplus \mathbb{P}; \mathbb{B}; \mathbb{V}\rangle$ such that $\sigma \rightarrowtail_!^r \tau$ for some τ. If $\mathcal{CT} \models \forall(\mathbb{B} \rightarrow \bar{t} = \bar{t}')$, we have $\sigma \equiv_! \sigma'$ and hence $\sigma' \rightarrowtail_!^r \tau$. The soundness of the rules introduced in step 3 is thus reduced to the soundness of those from step 1 and step 2.

4. The rules introduced in step 4 enforce a minimal representation of the persistent store. As $\mathcal{S}_!(\langle \mathbb{G}; \{c(\bar{t}), c(\bar{t})\} \uplus \mathbb{P}; \mathbb{B}; \mathbb{T}\rangle_k^{\mathbb{V}}) \equiv_! \mathcal{S}_!(\langle \mathbb{G}; \{c(\bar{t})\} \uplus \mathbb{P}; \mathbb{B}; \mathbb{T}\rangle_k^{\mathbb{V}})$, they are invariant to soundness.

From 1-4 follows that $\sigma_t \rightarrowtail_t^* \tau_t$ implies $\mathcal{S}_!(\sigma_t) \rightarrowtail_!^* \mathcal{S}_!(\tau_t)$.

Now assume that τ_t *is a* ***fixed point*** *w.r.t.* $[\![\mathcal{P}]\!]$. *This implies that for every possible match (if any) between sequences of constraints* H_1, H_2 *in* τ_t *and a rule* r *in* $[\![\mathcal{P}]\!]$, *there is a token* $(r, id(H_1) + id(H_2))$ *in the propagation history* $\mathbb{T}_\tau$ *inhibiting the firing of* r. *It follows that* r *is of the form* $r \ @ \ l(H_1^l) \uplus p(H^p) \Rightarrow G \mid p(B_c) \uplus B_b$ *and that* $p(B_c)$ *and* B_b *are already contained in* τ_t *from an earlier firing of* r. *Hence, for every possible match (if any) between constraints in* $\mathcal{S}_!(\tau_t)$ *and a rule* r' *in* $\mathcal{P}$, *the firing of* r' *is inhibited by the irreflexivity condition. Thus,* $\mathcal{S}_!(\tau_t)$ *is a fixed point w.r.t.* $\mathcal{P}$.

So finally, if from $\sigma_t = \langle l(\mathbb{L}) \uplus p(\mathbb{P}) \uplus B; \emptyset; \top; \emptyset\rangle_0^{\mathbb{V}}$ *we can derive a fixed point* $\tau_t = \langle \emptyset; \mathbb{C}; \mathbb{B}'; \mathbb{T}\rangle_n^{\mathbb{V}}$ *where* $chr(\mathbb{C}) = l(\mathbb{L}') \uplus p(\mathbb{P}')$, *then from the* $\omega_!$ *state* $\langle \mathbb{L}; \mathbb{P}; B; \mathbb{V}\rangle$ *we can derive a fixed point* $\langle \mathbb{L}'; \mathbb{P}'; \mathbb{B}'; \mathbb{V}\rangle$. *It follows by Def. 17 that for any goal* $G \wedge B$, *we have* $\mathcal{A}^t_{[\![\mathcal{P}]\!]}(G \wedge B) \subseteq \mathcal{A}^!_{\mathcal{P}}(G \wedge B)$.

Completeness: *The* ***Introduce*** *and* ***Solve*** *rules of* ω_t *guarantee that for every* $\sigma_t \in \Sigma_t$ *there exists* $\mathbb{T}, k$ *such that*

$$\mathcal{S}_!(\sigma_t) = \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V}\rangle \quad \Rightarrow \quad \sigma_t \rightarrowtail_t^* \langle \emptyset; \mathbb{C}; \mathbb{B}; \mathbb{T}\rangle_k^{\mathbb{V}} \text{ s.t. } chr(\mathbb{C}) = l(\mathbb{L}) \uplus p(\mathbb{P}) \quad (1)$$

With respect to ***ApplyLinear***, *assume* $\sigma_t = \langle \emptyset; \mathbb{C}; \mathbb{B}; \mathbb{T}\rangle_k^{\mathbb{V}}$, $\sigma_! = \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V}\rangle$, $\tau_!$ *such that* $chr(\mathbb{C}) = l(\mathbb{L}) \uplus p(\mathbb{P})$, *and a rule* $r \ @ \ H_1^l \uplus H_1^p \setminus H_2^l \uplus H_2^p \Leftrightarrow G \mid B_c, B_b$ *such that* $\sigma_! \rightarrowtail_!^r \tau_!$.

From $\sigma_! \rightarrowtail_!^r \tau_!$ *follows that* $\sigma_! \equiv_! \sigma_!' = \langle H_1^l \uplus H_2^l \uplus \mathbb{L}'; H_1^p \uplus H_2^p \uplus \mathbb{P}'; G \wedge \mathbb{B}'; \mathbb{V}\rangle$ *and* $\tau_! \equiv_! \tau_!' = \langle H_1^l \uplus B_c \uplus \mathbb{L}'; H_1^p \uplus H_2^p \uplus \mathbb{P}'; G \wedge \mathbb{B}' \wedge B_b; \mathbb{V}\rangle$, *where* $H_2^l \neq \emptyset$ *and* $\bar{y}, \bar{y}'$ *are the local variables of* $\sigma_!, \sigma_!'$. *We assume w.l.o.g. that* $\bar{y}, \bar{y}'$ *are disjoint. Hence, Thm. 2 implies:*

$$\mathcal{CT} \models \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{L} = H_1^l \uplus H_2^l \uplus \mathbb{L}') \wedge (\mathbb{P} \bowtie H_1^p \uplus H_2^p \uplus \mathbb{P}') \wedge \mathbb{B}' \wedge G))) \quad (2)$$

$$\mathcal{CT} \models \forall((\mathbb{B}' \wedge G) \rightarrow \exists \bar{y}.((\mathbb{L} = H_1^l \uplus H_2^l \uplus \mathbb{L}') \wedge (\mathbb{P} \bowtie H_1^p \uplus H_2^p \uplus \mathbb{P}') \wedge \mathbb{B}))) \quad (3)$$

By step 1 of our encoding, $[\![\mathcal{P}]\!]$ *contains a rule*

$$r' \ @ \ l(H_1^l) \uplus p(H_1^p) \uplus p(H_2^p) \setminus l(H_2^l) \Leftrightarrow G \mid l(B_c), B_b$$

We aptly decompose $\mathbb{L}$ *into three components* $\mathbb{L} = H_1^{l'} \uplus H_2^{l'} \uplus \mathbb{L}''$ *such that:*

$$\mathbb{L} = H_1^l \uplus H_2^l \uplus \mathbb{L}' \quad \Rightarrow \quad (H_1^{l'} = H_1^l) \wedge (H_2^{l'} = H_2^l) \wedge (\mathbb{L}'' = \mathbb{L}') \quad (4)$$

It is not guaranteed that $\mathbb{P} \bowtie H_1^p \uplus H_2^p \uplus \mathbb{P}' \Rightarrow (H_1^p \uplus H_2^p) \subseteq \mathbb{P}$. *However, we can decompose* $\mathbb{P}$ *into two components* $\mathbb{P} = H^{p'} \uplus \mathbb{P}''$ *such that*

$$\mathbb{P} \bowtie H_1^p \uplus H_2^p \uplus \mathbb{P}' \quad \Rightarrow \quad (H^{p'} \bowtie H_1^p \uplus H_2^p) \wedge (H^{p'} \subseteq H_1^p \uplus H_2^p) \quad (5)$$

Step 3 then guarantees that $[\![\mathcal{P}]\!]$ *contains a rule*

$$r'' \ @ \ l(H_1^l) \uplus p(H^p) \setminus l(H_2^l) \Leftrightarrow G' \wedge G \mid l(B_c), B_b$$

such that

$$\mathcal{CT} \models G' \leftrightarrow H^p \bowtie H_1^p \uplus H_2^p \quad (6)$$

and

$$\mathbb{P} \bowtie H_1^p \uplus H_2^p \uplus \mathbb{P}' \quad \Rightarrow \quad H^{p\prime} = H^p \tag{7}$$

Applying (7),(5), and (6) gives us

$$\mathcal{CT} \models (\mathbb{P} \bowtie H_1^p \uplus H_2^p \uplus \mathbb{P}') \Rightarrow G' \tag{8}$$

Hence, from (2), we get

$$\mathcal{CT} \models \forall(\mathbb{B} \rightarrow \exists \bar{y}'.((\mathbb{L} = H_1^l \uplus H_2^l \uplus \mathbb{L}') \wedge (\mathbb{P} \bowtie H_1^p \uplus H_2^p \uplus \mathbb{P}') \wedge G' \wedge \mathbb{B}' \wedge G))) \tag{9}$$

By Def. 7, we can thus derive $\sigma_t \rightarrowtail_t^{r''} \tau_t$ *for*

$$\tau_t = \langle l(B_c) \uplus B_b; \mathbb{C}'; \mathbb{B} \wedge (\tilde{\mathbb{B}} \wedge G' \wedge \mathbb{B}' \wedge G; \mathbb{T}'\rangle_k^{\mathbb{V}}$$

for some $\mathbb{T}'$ *and where* $chr(\mathbb{C}') = l(H_1^{l\prime} \uplus \mathbb{L}'') \uplus p(H^{p\prime} \uplus \mathbb{P}'')$ *and* $\tilde{\mathbb{B}} = (\mathbb{L} = H_1^l \uplus H_2^l \uplus \mathbb{L}') \wedge (\mathbb{P} \bowtie H_1^p \uplus H_2^p \uplus \mathbb{P}')$*. Consequently,*

$$\mathcal{S}_!(\tau_t) = \langle H_1^{l\prime} \uplus \mathbb{L}'' \uplus B_c; H^{p\prime} \uplus \mathbb{P}''; \mathbb{B} \wedge \tilde{\mathbb{B}} \wedge G' \wedge \mathbb{B}' \wedge G \wedge B_b; \mathbb{V}\rangle$$

Applying (4) and Def. 12.1 gives us:

$$\mathcal{S}_!(\tau_t) \equiv \langle H_1^l \uplus \mathbb{L}' \uplus B_c; H^{p\prime} \uplus \mathbb{P}''; \mathbb{B} \wedge \tilde{\mathbb{B}} \wedge G' \wedge \mathbb{B}' \wedge G \wedge B_b; \mathbb{V}\rangle$$

Since $\mathbb{P} = H^{p\prime} \uplus \mathbb{P}''$*, we can apply the matching* $(\mathbb{P} \bowtie H_1^p \uplus H_2^p \uplus \mathbb{P}')$ *we find in the guard to get*

$$\mathcal{S}_!(\tau_t) \equiv \langle H_1^l \uplus \mathbb{L}' \uplus B_c; H_1^p \uplus H_2^p \uplus \mathbb{P}'; \mathbb{B} \wedge \tilde{\mathbb{B}} \wedge G' \wedge \mathbb{B}' \wedge G \wedge B_b; \mathbb{V}\rangle$$

As the variables in $\bar{y}, \bar{y}'$ *are disjoint, we apply (3), (8), and Def. 12.2 to receive:*

$$\mathcal{S}_!(\tau_t) \equiv \langle H_1^l \uplus \mathbb{L}' \uplus B_c; H_1^p \uplus H_2^p \uplus \mathbb{P}'; \mathbb{B}' \wedge G \wedge B_b; \mathbb{V}\rangle = \tau_!$$

We proceed similarly for ***ApplyPersistent****. Hence, for any* $\sigma_t \in \Sigma_t, \tau_! \in \Sigma_!$ *such that* $\mathcal{S}_!(\sigma_t) \rightarrowtail_!^r \tau_!$*, there exists a* $\tau_t \in \Sigma_t$ *s.t.* $\sigma_t \rightarrowtail_t \tau_t$ *and* $\mathcal{S}_!(\tau_t) \equiv_! \tau_!$*.*
Fixed points: *Assume that* $\sigma_! = \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V}\rangle$ *is a fixed point in* $\omega_!$*. According to Def. 14, one of the following applies: (1) There is no rule* r @ $H_1^l \uplus H_1^p \backslash H_2^l \uplus H_2^p \Leftrightarrow G \mid B_c, B_b$ *in* $\mathcal{P}$ *such that* $\sigma_! \equiv_! \langle H_1^l \uplus H_1^p \uplus \mathbb{L}'; H_1^p \uplus H_2^p \uplus \mathbb{P}'; G \wedge \mathbb{B}'; \mathbb{V}\rangle$*. (2) Such a rule exists but its application violates the non-reflexivity condition, i.e. for the hypothetical follow-up state* $\tau_!$*, we have* $\sigma_! \equiv_! \tau_!$*.*

Now consider a state σ_t *s.t.* $\mathcal{S}_!(\sigma_t) = \sigma_!$*. Hence, it is of the form* $\sigma_t = \langle \emptyset; \mathbb{C}; \mathbb{B}; \mathbb{T}\rangle_k^V$ *s.t.* $chr(\mathbb{C}) = l(\mathbb{L}) \uplus p(\mathbb{P})$*. In case (1), no rules in* $[\![\mathcal{P}]\!]$ *are applicable to* $\sigma_t = \langle \emptyset; l(\mathbb{L}) \uplus p(\mathbb{P}); \mathbb{B}; \mathbb{T}\rangle_k^V$*, except for those of the form* $c(\bar{t}) \setminus c(\bar{t}) \Leftrightarrow \top$*. The program will quiesce in a state* σ_t' *s.t.* $\mathcal{S}_!(\sigma_t') \equiv_! \sigma_!$ *after finitely many applications of such rules. In case (2) – assuming a non-pathological CHR program – all possible applications are of the type* ***ApplyPersistent*** *(cf. Lemma 8) . Consequently, all rules applicable to* σ_t *in* $[\![\mathcal{P}]\!]$ *are of the form* r' @ $l(H_1^l) \uplus p(\hat{H}^p) \Rightarrow G \mid p(B_c) \uplus B_b$ *or* $c(\bar{t}) \setminus c(\bar{t}) \Leftrightarrow \top$*.*

For each such rule r' @ $l(H_1^l) \uplus p(\hat{H}^p) \Rightarrow G \mid p(B_c) \uplus B_b$*, we can tell by* $\sigma_! \equiv_! \tau_!$ *that* $p(B_c)$ *is contained in* $p(\mathbb{P})$ *and* B_b *is contained in* $\mathbb{B}$*. Hence, we can apply* r' *to* σ_t*, followed by finitely many applications of rules of the form* $c(\bar{t}) \setminus c(\bar{t}) \Leftrightarrow \top$ *to finitely derive a state* $\tau_t = \langle \emptyset; \mathbb{C}; \mathbb{B}'; \mathbb{T}' \rangle_{k'}^{V}$ *such that* $\mathcal{S}_!(\sigma_t) \equiv_! \mathcal{S}_!(\tau_t)$ *and* r' *is not applicable to* τ_t*. We repeat this for every applicable rule* r'*. After finitely many such sequences of derivation steps, no such rule remains applicable. Thus, we can finally derive a fixed point* ${\tau_t}'$ *such that* $\mathcal{S}_!(\sigma_t) \equiv_! \mathcal{S}_!({\tau_t}')$*.*

It follows by Def. 17 that for any goal $G \wedge B$*, we have* $\mathcal{A}_{\mathcal{P}}^{!}(G \wedge B) \subseteq \mathcal{A}_{[\![\mathcal{P}]\!]}^{t}(G \wedge B)$*.*

□

Example 2 (Termination Correspondence). The termination behavior of $\omega_!$ programs encoded in ω_t, via the encoding used to prove Lemma 14, changes. Consider a program $\mathcal{P}$ consisting only of the rule $a \Longrightarrow a$ that is clearly terminating in $\omega_!$. It's corresponding encoded program $[\![\mathcal{P}]\!]$ is given below.

$$\begin{array}{lll} r_1 \ @\ a(l) & \Longrightarrow & a(p) \\ r_2 \ @\ a(p) & \Longrightarrow & a(p) \\ r_3 \ @\ a(p) \backslash a(p) & \Leftrightarrow & \top \end{array}$$

It is an acceptable encoding according to Definition 18, and hence, answers are preserved. Nevertheless, there exists the following infinite computation.

$$\begin{array}{ll} \sigma = & \langle a(l); \emptyset; \top; \emptyset \rangle_0^{\emptyset} \\ \rightarrowtail_t & \langle \emptyset; a(l)\#0; \top; \emptyset \rangle_1^{\emptyset} \\ \rightarrowtail_t^{r_1} & \langle a(p); a(l)\#0; \top; \{(r_1, 0)\} \rangle_1^{\emptyset} \\ \rightarrowtail_t & \langle \emptyset; a(l)\#0, a(p)\#1; \top; \{(r_1, 0)\} \rangle_2^{\emptyset} \\ \rightarrowtail_t^{r_2} & \langle a(p); a(l)\#0, a(p)\#1; \top; \{(r_1, 0), (r_2, 1)\} \rangle_2^{\emptyset} \\ \rightarrowtail_t & \langle \emptyset; a(l)\#0, a(p)\#1, a(p)\#2; \top; \{(r_1, 0), (r_2, 1)\} \rangle_3^{\emptyset} \\ \rightarrowtail_t^{r_2} & \langle a(p); a(l)\#0, a(p)\#1, a(p)\#2; \top; \{(r_1, 0), (r_2, 1), (r_2, 2)\} \rangle_3^{\emptyset} \\ \rightarrowtail_t & \ldots \end{array}$$

The reason for this difference is found in rules r_2 and r_3: they enforce set semantics on the constraints, supposedly corresponding to irreflexivity in $\omega_!$. However, the non-determinism of ω_t seems to hinder proper enforcing of irreflexivity via rules.

Lemma 15 **($\omega_p \not\rightarrow \omega_!$).** *There exists no acceptable encoding of* ω_p *into* $\omega_!$*.*

Proof. Follows from [21]. Note that [21] considers only data sufficient answers, however, as there exists no acceptable encoding of the program given in their proof, the negative result carries over to the generic case of answers. □

Lemma 16 **($\omega_! \rightarrow \omega_p$).** *There exists an acceptable encoding of* $\omega_!$ *into* ω_p*.*

Proof. We show how to encode any $\omega_!$ *program* $\mathcal{P}$ *in* ω_p*. For every* n*-ary constraint* c/n *in* $\mathcal{P}$*, there exists a constraint* $c/(n+1)$ *in* $[\![\mathcal{P}]\!]$*. In the following, for a multiset of user-defined* $\omega_!$*-constraints* $M = \{c_1(\bar{t}_1), \ldots, c_n(\bar{t}_n)\}$ *let*

$l(M) = \{c_1(l,\bar{t}_1),\ldots,c_n(l,\bar{t}_n)\}$, $p(M) = \{c_1(p,\bar{t}_1),\ldots,c_n(p,\bar{t}_n)\}$, and $c(M) = \{c_1(c,\bar{t}_1),\ldots,c_n(c,\bar{t}_n)\}$. The encoded program $[\![\mathcal{P}]\!]$ is constructed as follows:

Apply rules 1-3 from the proof of Lemma 14, but in rule 2 replace $p(B_c)$ with $c(B_c)$. Assign to each of these rules the constant priority 3. Additionally, add the following rules to $[\![\mathcal{P}]\!]$ for each constraint c/n where $\bar{t}$ is a sequence of n different variables:

$$1 :: c(p,\bar{t})\backslash c(c,\bar{t}) \Leftrightarrow \top$$
$$2 :: c(c,\bar{t}) \Leftrightarrow c(p,\bar{t})$$

The translation of goals is defined as $[\![\mathbb{L}\wedge !\mathbb{P}]\!]_g ::= l(\mathbb{L}) \wedge p(\mathbb{P})$.
Soundness: *Let $\mathcal{S}_! : \Sigma_p \to \Sigma_!$, $\sigma_p = \langle l(\mathbb{L}) \uplus p(\mathbb{P}) \uplus c(\mathbb{P}_c) \uplus \mathbb{B}'; \mathbb{C}; \mathbb{B}; \mathbb{T}\rangle_k^{\mathbb{V}} \mapsto \langle \mathbb{L} \uplus \mathbb{L}'; \mathbb{P}\uplus\mathbb{P}'\uplus\mathbb{P}_c\uplus\mathbb{P}'_c; \mathbb{B}\wedge\mathbb{B}'; \mathbb{V}\rangle$ where $\mathrm{chr}(\mathbb{C}) = l(\mathbb{L}')\uplus p(\mathbb{P}')\uplus c(\mathbb{P}'_c)$. In the following, we will show that for all $\sigma_p, \tau_p \in \Sigma_p$, $\sigma_p \rightarrowtail_p^* \tau_p$ implies $\mathcal{S}_!(\sigma_p) \rightarrowtail_!^* \mathcal{S}_!(\tau_p)$.*

The proof is analogous to Lemma 14 for the rules of priority 3. As $c(\bar{t}) \bowtie c(\bar{t}) \uplus c(\bar{t})$ rules of priority 1 and 2 are invariant to $\mathcal{S}_!$.

*Now assume τ_p is a **fixed point** w.r.t. $[\![\mathcal{P}]\!]$. Analogously to Lemma 14, $\mathcal{S}_!(\tau_p)$ is a fixed point w.r.t. $\mathcal{P}$. The only difference being $c(B_c)$ in the body instead of $p(B_c)$, but rules of priority 1 and 2 would then be applicable to convert $c(B_c)$ into $p(B_c)$ modulo set semantics. Therefore, it follows that $\mathcal{A}^p_{[\![\mathcal{P}]\!]}([\![G]\!]_g) \subseteq [\![\mathcal{A}^!_{\mathcal{P}}(G)]\!]_g$.*
Completeness: *Analogously to Lemma 14 we have that for any $\sigma_p \in \Sigma_p$, $\tau_! \in \Sigma_!$ such that $\mathcal{S}_!(\sigma_p) \rightarrowtail_!^r \tau_!$, there exists a $\tau_p \in \Sigma_p$ such that $\sigma_p \rightarrowtail_p^* \tau_p$ and $\mathcal{S}_!(\tau_p) \equiv_! \tau_!$. The only change to the proof is that after applying a rule of the encoded program we also apply all possible **Introduce** and **Solve** transitions, as well as all rule applications with priorities 1 and 2 (all these operations are invariant to $\mathcal{S}_!$). Hence, the resulting state τ_p contains only identified constraints whose first argument is either l or p. All constraints with argument c are either replaced by the corresponding one with argument p by the rule of priority 2, or they are removed, because a corresponding constraint already exists.*

*Now assume $\sigma_! = \langle \mathbb{L}; \mathbb{P}; \mathbb{B}; \mathbb{V}\rangle$ is a **fixed point** of $\mathcal{P}$, then there exists $\sigma_p \in \Sigma_p$ with $\mathcal{S}_!(\sigma_p) = \sigma_!$. There are two possible cases:*

1. *$\sigma_!$ is not equivalent to a state applicable to any rule r in $\mathcal{P}$*
2. *all rule applications would violate irreflexivity*

In case 1, σ_p clearly is a fixed point as well (otherwise the above soundness result violates the assumption).

*Therefore, consider case 2. We assume non-pathological programs, so that, according to Lemma 8, **ApplyLinear** never violates irreflexivity. Hence, there exists a rule in $[\![\mathcal{P}]\!]$:*

$$3 :: r' \ @ \ l(H^l) \uplus p(H^p) \Longrightarrow G \mid c(B_c), B_b$$

*In the following, for a set M of constraints let $\#M$ denote the corresponding set of identified constraints. Assume σ_p is no fixed point, then $\sigma_p = \langle \emptyset; \#l(\hat{H}^l) \cup \#p(\hat{H}^p) \cup \mathbb{C}; \mathbb{B}; \mathbb{T}\rangle_k^{\mathbb{V}}$ and $\mathcal{CT} \models \forall(\mathbb{B} \to (A \wedge G))$ with $\sigma_p \rightarrowtail_p^{r'} \tau_p = \langle c(B_c) \uplus B_b; \#l(\hat{H}^l) \cup \#p(\hat{H}^p) \cup \mathbb{C}; \mathbb{B} \wedge A; \mathbb{T}'\rangle_k^{\mathbb{V}}$, where $A ::= \mathrm{chr}(\#l(\hat{H}^l)) = l(H^l) \wedge \mathrm{chr}(\#p(\hat{H}^p)) = p(H^p)$. Applying **Introduce** and **Solve** we get $\tau_p \rightarrowtail_p^* \langle \emptyset; \#l(\hat{H}^l) \cup \#p(\hat{H}^p) \cup \mathbb{C} \cup \#c(B_c); \mathbb{B} \wedge B_b \wedge A; \mathbb{T}'\rangle_m^{\mathbb{V}}$.*

The rule r' corresponds to a rule r in $\mathcal{P}$ and $\sigma_!$ is applicable to r, except for irreflexivity (this follows from soundness). The irreflexivity and Theorem 2 imply $\mathcal{CT} \models \mathbb{B} \rightarrow \exists \bar{x}.(H^p \bowtie H^p \uplus B_c) \wedge \mathbb{B} \wedge B_b$. Therefore, $\mathcal{CT} \models (\mathbb{B} \wedge B_b \wedge A) \rightarrow (\hat{H}^p \bowtie \hat{H}^p \uplus B_c)$. It follows that $\mathcal{CT} \models (\mathbb{B} \wedge B_b \wedge A) \rightarrow \forall c(c, \bar{t}) \in c(B_c).\exists c(p, \bar{t}') \in p(\hat{H}^p).\bar{t} = \bar{t}'$.

Therefore, for each $c(c, \bar{t}) \in c(B_c)$ we can apply the corresponding rule of priority $1 :: c(p, \hat{t}) \backslash c(c, \hat{t}) \Leftrightarrow \top$, as $\mathcal{CT} \models \forall(\mathbb{B} \wedge B_b \wedge A \rightarrow \exists \bar{x}.(\mathrm{chr}(c(p, \bar{t}')) = c(p, \hat{t}) \wedge \mathrm{chr}(c(c, \bar{t})) = c(c, \hat{t})))$. Therefore, each constraint in $c(B_c)$ is removed by rules of priority 1 and we get $\sigma_p \rightarrowtail_p^* \langle \emptyset; \#l(\hat{H}^l) \cup \#p(\hat{H}^p) \cup \mathbb{C}; \mathbb{B}; \mathbb{T}' \rangle_m^{\mathbb{V}} = \tau_p'$, such that the above rule application is prohibited by $\mathbb{T}'$.

Hence, we can w.l.o.g. choose τ_p' as σ_p above and repeat the argument. Therefore, we get a state in which the token store prohibits firing any more propagation rules. As no other rules are applicable either, this state is a fixed point corresponding to $\sigma_!$ as well. □

Lemma 17 **($\omega_! \not\rightarrow \omega_e$).** *There exists no acceptable encoding of $\omega_!$ into ω_e.*

Proof. *Consider the $\omega_!$ program $\mathcal{P} = (a \Rightarrow b)$. Since $\mathcal{A}_{\mathcal{P}}^! = \{a \wedge !b\}$, an acceptable encoding has to satisfy $\mathcal{A}_{[\![\mathcal{P}]\!]}^m([\![a]\!]_g) = \{[\![a \wedge !b]\!]_g\} = \{[\![a]\!]_g \wedge [\![b]\!]_g\}$. Therefore, $\langle [\![a]\!]_g; \top; \emptyset \rangle \rightarrowtail_e^+ \langle [\![a]\!]_g \wedge [\![!b]\!]_g; B; \emptyset \rangle$ where the result state has to be a non-failed final state, which is a contradiction to the monotonicity of ω_e.* □

Lemma 18 **($\omega_e \rightarrow \omega_!$).** *There exists an acceptable encoding of ω_e into $\omega_!$.*

Proof (Sketch). Replace propagation rules with simplification rules that contain a copy of the head in their bodies. □

6 Discussion

In this section, we discuss our insights on the behavior of $\omega_!$ in comparison with existing operational semantics.

6.1 Termination Behavior

Our proposed operational semantics $\omega_!$ exhibits a termination behavior different from ω_t, ω_p, and ω_e. Compared to ω_e, we have solved the problem of trivial non-termination of propagation rules, whereas any program terminating in ω_e also terminates in $\omega_!$. With respect to ω_t and ω_p, we found programs that terminate in $\omega_!$ but not in ω_t and ω_p, and vice versa.

Example 3. Consider the following program for computing the transitive hull of a graph.

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

Due to the presence of propagation rules, this program is non-terminating under ω_e. Under ω_t and ω_p, termination depends on the initial goal: It is shown in

[22] that this program terminates for acyclic graphs. However, goals containing cyclic graphs, such as $\langle (e(1,2), e(2,1)); \emptyset; \top; \emptyset \rangle_0^{\emptyset}$, entail non-terminating behavior:

$$
\begin{array}{ll}
 & \langle e(1,2), e(2,1); \emptyset; \top; \emptyset \rangle_0^{\mathbb{V}} \\
\rightarrowtail_t^* & \langle \emptyset; e(1,2)\#0, e(2,1)\#1; \top; \emptyset \rangle_2^{\mathbb{V}} \\
\rightarrowtail_t^r & \langle e(1,1); e(1,2)\#0, e(2,1)\#1; \top; \{(t,0,1)\} \rangle_2^{\mathbb{V}} \\
\rightarrowtail_t & \langle \emptyset; e(1,2)\#0, e(2,1)\#1, e(1,1)\#2; \top; \{(t,0,1)\} \rangle_3^{\mathbb{V}} \\
\rightarrowtail_t^r & \langle e(1,2); e(1,2)\#0, e(2,1)\#1, e(1,1)\#2; \top; \{(t,0,1), (t,2,0)\} \rangle_3^{\mathbb{V}} \\
\rightarrowtail_t & \ldots
\end{array}
$$

Under $\omega_!$, the previous goal terminates after computing the transitive hull.

$$
\begin{array}{ll}
 & \langle \{e(1,2), e(2,1)\}; \emptyset; \top; \emptyset \rangle \\
\rightarrowtail_!^t & \langle \{e(1,2), e(2,1)\}; \{e(1,1)\}; \top; \emptyset \rangle \\
\rightarrowtail_!^* & \langle \{e(1,2), e(2,1)\}; \{e(1,1), e(1,2), e(2,1), e(2,2)\}; \top; \emptyset \rangle \not\rightarrowtail_!
\end{array}
$$

This is in fact true for all possible inputs:

Proposition 1. *Under $\omega_!$, the transitive hull program terminates for every possible input.*

Proof. The only rule r propagates constraints of type $e/2$, which are necessarily persistent. The propagated constraints contain only the arguments X, Z, received as arguments in the rule head. No new arguments are introduced. Any given initial state contains a finite number of arguments. From these, only finitely many different e constraints can be built. As rule application is irreflexive, the computation therefore has to stop after a finite number of transition steps. □

Nevertheless, program termination in $\omega_!$ is not strictly stronger than that in ω_t or ω_p, as the following counterexample shows:

Example 4. Consider the following exemplary CHR program.

$$
\begin{array}{lcl}
\mathrm{r1} \ @ \ a & \Longrightarrow & b \\
\mathrm{r2} \ @ \ c(X), b & \Leftrightarrow & c(X{+}1)
\end{array}
$$

The program terminates in ω_t (and ω_p): As there can only be a finite number of a-constraints in the initial goal, rule r1 will create but a finite number of b-constraints. These will be consumed by rule r2 in finite time, followed by quiescence:

$$
\begin{array}{ll}
 & \langle (a, c(X)); \emptyset; \top; \emptyset \rangle_0^{\{X\}} \\
\rightarrowtail_t^* & \langle \emptyset; \{a\#0, b\#1, c(X)\#2\}; \top; \{(\mathrm{r1}, 0)\} \rangle_3^{\{X\}} \\
\rightarrowtail_t^{\mathrm{r2}} & \langle c(X{+}1); \{a\#0\}; \top; \{(\mathrm{r1}, 0)\} \rangle_3^{\{X\}} \\
\rightarrowtail_t & \langle \emptyset; \{a\#0, c(X{+}1)\#3; \top; \{(\mathrm{r1}, 0)\} \rangle_4^{\{X\}} \not\rightarrowtail_t
\end{array}
$$

In contrast, the same program exhibits non-terminating behavior in $\omega_!$, as the following infinite derivation shows:

$$\begin{aligned} & \langle \{a, c(X)\}; \emptyset; \top; \{X\} \rangle \\ \rightarrowtail_!^{r1} & \langle \{a, c(X)\}; \{b\}; \top; \{X\} \rangle \\ \rightarrowtail_!^{r2} & \langle \{a, c(X{+}1)\}; \{b\}; \top; \{X\} \rangle \\ \rightarrowtail_!^{r2} & \langle \{a, c(X{+}2)\}; \{b\}; \top; \{X\} \rangle \\ \rightarrowtail_!^{r2} & \ldots \end{aligned}$$

6.2 Limitations of the current approach

As specified in Sect. 3, our approach requires range-restricted programs. In the following we explain why a naive extension to the full segment of CHR by dropping the restriction to range-restricted programs would violate both soundness and completeness.

We recall that a persistent constraint is a finite representation of an arbitrary number of identical constraints, such as a propagation rule from the range-restricted segment may generate under ω_e once it is applicable. Under the same conditions, however, a propagation rule with local variables would generate an arbitrary number of nearly but *not quite* identical constraints, as the local variables would be renamed apart between any two of those nearly identical constraints. Consider the following example:

$$\begin{aligned} & r_1 \ @\ a && \Longrightarrow b(X) \\ & r_2 \ @\ b(X), b(X) && \Leftrightarrow\ c \end{aligned}$$

When executed with the initial goal a, this program causes the following infinite derivation under ω_e.

$$\begin{aligned} & \langle a; \top; \emptyset \rangle \\ \rightarrowtail_e^{r_1} & \langle a, b(X'); \top; \emptyset \rangle \\ \rightarrowtail_e^{r_1} & \langle a, b(X'), b(X''); \top; \emptyset \rangle \rightarrowtail_e^{r_1} \ldots \end{aligned}$$

The variables $X', X'', \ldots$ are distinct from each other and from the variable X which occurs in the rule body. Thus, it is impossible to derive the constraint c from goal a under ω_e.

Under the current approach, we cannot finitely represent an arbitrary number of such nearly identical constraints. A naive extension of $\omega_!$ to the full segment of CHR as specified above would discard the distinction between the two types of generated constraints altogether.

With respect to our example, the following derivation would be possible:

$$\begin{aligned} & \langle \{a\}; \emptyset; \top; \emptyset \rangle \\ \rightarrowtail_!^{r_1} & \langle \{a\}; \{b(X')\}; \top; \emptyset \rangle \equiv_! \langle \{a\}; \{b(X'), b(X')\}; \top; \emptyset \rangle \\ \rightarrowtail_!^{r_2} & \langle \{a\}; \{b(X'), b(X'), c\}; \top; \emptyset \rangle \end{aligned}$$

We assume that a non-naive extension to the full segment of CHR preserving soundness and completeness is possible, though it is beyond the scope of this paper.

6.3 Related Work

In [23] the set-based semantics ω_{set} has been introduced. Its development was, among other considerations, driven by the intention to eliminate the propagation history. Besides addressing the problem of trivial non-termination in a novel manner, it reduces non-determinism similarly to the refined operational semantics ω_r [13]. In ω_{set}, a propagation rule cannot be fired infinitely often for a possible matching. However, multiple firings are possible, the exact number depending on the built-in store.

The authors of [23] justify their set-based approach by the following statement:

> "When working with a multi-set-based constraint store, it appears that propagation history is essential to provide a reasonable semantics."

Our approach can be understood as a compromise since we avoid a propagation history by imposing an implicit set semantics on persistent constraints. The distinction between linear and persistent constraints, however, allows us to restrict the set behavior to those constraints, whereas the multiset semantics is preserved for linear constraints.

Linear logical algorithms [15] (LLA) is a programming language based on bottom-up reasoning in linear logic, inspired by logical algorithms [24]. The first implementation of logical algorithms was realized in CHR with rule priorities [25].

Our proposed operational semantics $\omega_!$ is related to LLA [15], but displays significant differences: Firstly, the notion of a constraint theory with built-in constraints is absent in LLA. Secondly, LLA rules are restricted such that persistent propositions cannot be derived multiple times, whereas $\omega_!$ makes no such restriction and solves this problem via the irreflexive transition system. Thirdly, LLA requires a strict separation of propositions into linear and persistent ones. In $\omega_!$ a CHR constraint can occur in the linear store, in the persistent store, or both.

On the other hand, the separation of propositions in LLA allows the corresponding rules to freely mix linear and persistent propositions in bodies. This is not directly possible with our approach, as CHR constraints in a body are either added as linear or persistent constraints.

7 Conclusion and Future Work

The main motivation of this work is the observation that CHR research spans a spectrum ranging from an analytical to a pragmatic end: on the analytical side of the spectrum, emphasis is put on the formal aspects and properties of the language while on the pragmatic side, it is put on implementation and efficiency. A variety of operational semantics has been brought forth in the past, each aligning with one side of the spectrum. In this work we propose the novel operational semantics $\omega_!$, heeding both analytical and pragmatic aspects. Unlike other operational semantics with a strong analytical foundation, $\omega_!$ thus provides a terminating execution model and may be implemented as is.

Our operational semantics $\omega_!$ is based on the concept of persistent constraints. These are finite representations of an arbitrarily large number of syntactically equivalent constraints. They enable us to subsume trivially non-terminating computations in a single derivation step.

We proved soundness and completeness of our operational semantics $\omega_!$ with respect to ω_e. The latter stands exemplarily for analytical formalizations of the operational semantics, thus providing a strong analytical foundation for $\omega_!$. This facilitates program analysis and formal proofs of program properties.

Applying Shapiro's concept of acceptable encodings [20], we compared the expressivity of $\omega_!$ with respect to the operational semantics ω_e, ω_t, and ω_p. One significant result is a faithful encoding of $\omega_!$ into ω_p, which may effectively serve as an implementation of $\omega_!$.

In its current formulation, $\omega_!$ is only applicable to range-restricted CHR programs – a limitation we plan to address in the future. Furthermore, similar to ω_t being the basis for numerous extensions to CHR [17], we plan to investigate the effect of building these extensions on $\omega_!$.

In a concurrent environment, some kind of conflict resolution is required for the case that multiple rules try to remove the same constraint. For example, in [10] a transaction-based approach is used, leading to a rollback, if the first evaluated rule application removed the constraint. The formulation of the **ApplyPersistent** transition reveals that for persistent constraints, no such conflicts have to be taken into account. A closer investigation of potential benefits of the persistent constraint approach in concurrent settings remains to be conducted.

References

1. Frühwirth, T.: Constraint Handling Rules. Cambridge University Press (2009)
2. Frühwirth, T., Hanschke, P.: Terminological reasoning with Constraint Handling Rules. In: Principles and Practice of Constraint Programming, MIT Press (1995)
3. Frühwirth, T.: Theory and practice of constraint handling rules. Journal of Logic Programming, Special Issue on Constraint Logic Programming **37**(1-3) (October 1998) 95–138
4. Frühwirth, T., Abdennadher, S.: Essentials of Constraint Programming. Springer-Verlag (2003)
5. Abdennadher, S., Frühwirth, T.: Operational equivalence of CHR programs and constraints. In Jaffar, J., ed.: Principles and Practice of Constraint Programming, CP 1999. Volume 1713 of Lecture Notes in Computer Science., Springer-Verlag (1999) 43–57
6. Abdennadher, S., Frühwirth, T., Meuss, H.: Confluence and semantics of constraint simplification rules. Constraints **4**(2) (1999) 133–165
7. Betz, H., Frühwirth, T.: A linear-logic semantics for constraint handling rules. In van Beek, P., ed.: Principles and Practice of Constraint Programming, 11th International Conference, CP 2005. Volume 3709 of Lecture Notes in Computer Science., Sitges, Spain, Springer-Verlag (October 2005) 137–151

8. Frühwirth, T.: Parallelizing union-find in constraint handling rules using confluence analysis. In: Principles and Practice of Constraint Programming, 11th International Conference, CP 2005, Sitges, Spain (October 2005)
9. Sulzmann, M., Lam, E.S.L.: A concurrent constraint handling rules semantics and its implementation with software transactional memory. In: Declarative Aspects of Multicore Programming, ACM SIGPLAN Workshop. (2007)
10. Sulzmann, M., Lam, E.S.L.: Parallel execution of multi-set constraint rewrite rules. In Antoy, S., Albert, E., eds.: Proceedings of the 10th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP), Valencia, Spain, ACM (July 2008) 20–31
11. Raiser, F., Betz, H., Frühwirth, T.: Equivalence of CHR states revisited. In Raiser, F., Sneyers, J., eds.: 6th International Workshop on Constraint Handling Rules (CHR). (2009) 34–48
12. Abdennadher, S.: Operational semantics and confluence of constraint propagation rules. In: Principles and Practice of Constraint Programming. (1997) 252–266
13. Duck, G.J., Stuckey, P.J., García de la Banda, M., Holzbaur, C.: The refined operational semantics of Constraint Handling Rules. In Demoen, B., Lifschitz, V., eds.: Logic Programming, 20th International Conference, ICLP 2004. Volume 3132 of Lecture Notes in Computer Science., Saint-Malo, France, Springer-Verlag (September 2004) 90–104
14. De Koninck, L., Schrijvers, T., Demoen, B.: User-definable rule priorities for CHR. In: PPDP '07: Proceedings of the 9th ACM SIGPLAN international conference on Principles and practice of declarative programming, New York, NY, USA, ACM (2007) 25–36
15. Simmons, R.J., Pfenning, F.: Linear logical algorithms. In Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I., eds.: Automata, Languages and Programming, 35th International Colloquium, ICALP 2008. Volume 5126 of Lecture Notes in Computer Science., Springer-Verlag (2008) 336–347
16. Betz, H., Raiser, F., Frühwirth, T.: Persistent constraint in constraint handling rules. In: Proceedings of 23rd Workshop on (Constraint) Logic Programming, WLP 2009. (2009) to appear.
17. Sneyers, J., Van Weert, P., Schrijvers, T., De Koninck, L.: As time goes by: Constraint Handling Rules – A survey of CHR research between 1998 and 2007. Accepted by *Journal of Theory and Practice of Logic Programming* (2010)
18. De Koninck, L., Stuckey, P.J., Duck, G.J.: Optimizing compilation of CHR with rule priorities. In Garrigue, J., Hermenegildo, M.V., eds.: Functional and Logic Programming, 9th International Symposium (FLOPS). Volume 4989 of Lecture Notes in Computer Science., Springer-Verlag (2008) 32–47
19. Sneyers, J., Schrijvers, T., Demoen, B.: The computational power and complexity of constraint handling rules. ACM Trans. Program. Lang. Syst. **31**(2) (2009) 1–42
20. Shapiro, E.: The family of concurrent logic programming languages. ACM Comput. Surv. **21**(3) (1989) 413–510
21. Gabbrielli, M., Mauro, J., Meo, M.C.: On the expressive power of priorities in CHR. In Porto, A., López-Fraguas, F.J., eds.: Proceedings of the 11th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, Coimbra, Portugal, ACM (2009) 267–276
22. Pilozzi, P., Schreye, D.D.: Proving termination by invariance relations. In Hill, P.M., Warren, D.S., eds.: 25th International Conference Logic Programming, ICLP. Volume 5649 of Lecture Notes in Computer Science., Pasadena, CA, USA, Springer-Verlag (July 2009) 499–503

23. Sarna-Starosta, B., Ramakrishnan, C.: Compiling Constraint Handling Rules for efficient tabled evaluation. In Hanus, M., ed.: 9th Intl. Symp. Practical Aspects of Declarative Languages, PADL. Volume 4354 of Lecture Notes in Computer Science., Nice, France, Springer-Verlag (jan 2007) 170–184
24. Ganzinger, H., McAllester, D.A.: Logical algorithms. In Stuckey, P.J., ed.: Logic Programming, 18th International Conference, ICLP 2002. Volume 2401 of Lecture Notes in Computer Science., Springer-Verlag (2002) 209–223
25. De Koninck, L.: Logical Algorithms meets CHR: A meta-complexity result for Constraint Handling Rules with rule priorities. Theory and Practice of Logic Programming **9**(2) (March 2009) 165–212

Liste der bisher erschienenen Ulmer Informatik-Berichte
Einige davon sind per FTP von `ftp.informatik.uni-ulm.de` erhältlich
Die mit * markierten Berichte sind vergriffen

List of technical reports published by the University of Ulm
Some of them are available by FTP from `ftp.informatik.uni-ulm.de`
Reports marked with * are out of print

91-01 *Ker-I Ko, P. Orponen, U. Schöning, O. Watanabe*
Instance Complexity

91-02* *K. Gladitz, H. Fassbender, H. Vogler*
Compiler-Based Implementation of Syntax-Directed Functional Programming

91-03* *Alfons Geser*
Relative Termination

91-04* *J. Köbler, U. Schöning, J. Toran*
Graph Isomorphism is low for PP

91-05 *Johannes Köbler, Thomas Thierauf*
Complexity Restricted Advice Functions

91-06* *Uwe Schöning*
Recent Highlights in Structural Complexity Theory

91-07* *F. Green, J. Köbler, J. Toran*
The Power of Middle Bit

91-08* *V.Arvind, Y. Han, L. Hamachandra, J. Köbler, A. Lozano, M. Mundhenk, A. Ogiwara, U. Schöning, R. Silvestri, T. Thierauf*
Reductions for Sets of Low Information Content

92-01* *Vikraman Arvind, Johannes Köbler, Martin Mundhenk*
On Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets

92-02* *Thomas Noll, Heiko Vogler*
Top-down Parsing with Simulataneous Evaluation of Noncircular Attribute Grammars

92-03 *Fakultät für Informatik*
17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen

92-04* *V. Arvind, J. Köbler, M. Mundhenk*
Lowness and the Complexity of Sparse and Tally Descriptions

92-05* *Johannes Köbler*
Locating P/poly Optimally in the Extended Low Hierarchy

92-06* *Armin Kühnemann, Heiko Vogler*
Synthesized and inherited functions -a new computational model for syntax-directed semantics

92-07* *Heinz Fassbender, Heiko Vogler*
A Universal Unification Algorithm Based on Unification-Driven Leftmost Outermost Narrowing

92-08* *Uwe Schöning*
On Random Reductions from Sparse Sets to Tally Sets

92-09* *Hermann von Hasseln, Laura Martignon*
Consistency in Stochastic Network

92-10 *Michael Schmitt*
A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function

92-11 *Johannes Köbler, Seinosuke Toda*
On the Power of Generalized MOD-Classes

92-12 *V. Arvind, J. Köbler, M. Mundhenk*
Reliable Reductions, High Sets and Low Sets

92-13 *Alfons Geser*
On a monotonic semantic path ordering

92-14* *Joost Engelfriet, Heiko Vogler*
The Translation Power of Top-Down Tree-To-Graph Transducers

93-01 *Alfred Lupper, Konrad Froitzheim*
AppleTalk Link Access Protocol basierend auf dem Abstract Personal Communications Manager

93-02 *M.H. Scholl, C. Laasch, C. Rich, H.-J. Schek, M. Tresch*
The COCOON Object Model

93-03 *Thomas Thierauf, Seinosuke Toda, Osamu Watanabe*
On Sets Bounded Truth-Table Reducible to P-selective Sets

93-04 *Jin-Yi Cai, Frederic Green, Thomas Thierauf*
On the Correlation of Symmetric Functions

93-05 *K.Kuhn, M.Reichert, M. Nathe, T. Beuter, C. Heinlein, P. Dadam*
A Conceptual Approach to an Open Hospital Information System

93-06 *Klaus Gaßner*
Rechnerunterstützung für die konzeptuelle Modellierung

93-07 *Ullrich Keßler, Peter Dadam*
Towards Customizable, Flexible Storage Structures for Complex Objects

94-01 *Michael Schmitt*
On the Complexity of Consistency Problems for Neurons with Binary Weights

94-02 *Armin Kühnemann, Heiko Vogler*
A Pumping Lemma for Output Languages of Attributed Tree Transducers

94-03 *Harry Buhrman, Jim Kadin, Thomas Thierauf*
On Functions Computable with Nonadaptive Queries to NP

94-04 *Heinz Faßbender, Heiko Vogler, Andrea Wedel*
Implementation of a Deterministic Partial E-Unification Algorithm for Macro Tree Transducers

94-05 *V. Arvind, J. Köbler, R. Schuler*
On Helping and Interactive Proof Systems

94-06 *Christian Kalus, Peter Dadam*
Incorporating record subtyping into a relational data model

94-07 *Markus Tresch, Marc H. Scholl*
A Classification of Multi-Database Languages

94-08 *Friedrich von Henke, Harald Rueß*
Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge

94-09 *F.W. von Henke, A. Dold, H. Rueß, D. Schwier, M. Strecker*
Construction and Deduction Methods for the Formal Development of Software

94-10 *Axel Dold*
Formalisierung schematischer Algorithmen

94-11 *Johannes Köbler, Osamu Watanabe*
New Collapse Consequences of NP Having Small Circuits

94-12 *Rainer Schuler*
On Average Polynomial Time

94-13 *Rainer Schuler, Osamu Watanabe*
Towards Average-Case Complexity Analysis of NP Optimization Problems

94-14 *Wolfram Schulte, Ton Vullinghs*
Linking Reactive Software to the X-Window System

94-15 *Alfred Lupper*
Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen

94-16 *Robert Regn*
Verteilte Unix-Betriebssysteme

94-17 *Helmuth Partsch*
Again on Recognition and Parsing of Context-Free Grammars:
Two Exercises in Transformational Programming

94-18 *Helmuth Partsch*
Transformational Development of Data-Parallel Algorithms: an Example

95-01 *Oleg Verbitsky*
On the Largest Common Subgraph Problem

95-02 *Uwe Schöning*
Complexity of Presburger Arithmetic with Fixed Quantifier Dimension

95-03 *Harry Buhrman,Thomas Thierauf*
The Complexity of Generating and Checking Proofs of Membership

95-04 *Rainer Schuler, Tomoyuki Yamakami*
Structural Average Case Complexity

95-05 *Klaus Achatz, Wolfram Schulte*
Architecture Indepentent Massive Parallelization of Divide-And-Conquer Algorithms

95-06 *Christoph Karg, Rainer Schuler*
Structure in Average Case Complexity

95-07 *P. Dadam, K. Kuhn, M. Reichert, T. Beuter, M. Nathe*
ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen

95-08 *Jürgen Kehrer, Peter Schulthess*
Aufbereitung von gescannten Röntgenbildern zur filmlosen Diagnostik

95-09 *Hans-Jörg Burtschick, Wolfgang Lindner*
On Sets Turing Reducible to P-Selective Sets

95-10 *Boris Hartmann*
Berücksichtigung lokaler Randbedingung bei globaler Zieloptimierung mit neuronalen Netzen am Beispiel Truck Backer-Upper

95-12 *Klaus Achatz, Wolfram Schulte*
Massive Parallelization of Divide-and-Conquer Algorithms over Powerlists

95-13 *Andrea Mößle, Heiko Vogler*
Efficient Call-by-value Evaluation Strategy of Primitive Recursive Program Schemes

95-14 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
A Generic Specification for Verifying Peephole Optimizations

96-01 *Ercüment Canver, Jan-Tecker Gayen, Adam Moik*
Formale Entwicklung der Steuerungssoftware für eine elektrisch ortsbediente Weiche mit VSE

96-02 *Bernhard Nebel*
Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class

96-03 *Ton Vullinghs, Wolfram Schulte, Thilo Schwinn*
An Introduction to TkGofer

96-04 *Thomas Beuter, Peter Dadam*
Anwendungsspezifische Anforderungen an Workflow-Mangement-Systeme am Beispiel der Domäne Concurrent-Engineering

96-05 *Gerhard Schellhorn, Wolfgang Ahrendt*
Verification of a Prolog Compiler - First Steps with KIV

96-06 *Manindra Agrawal, Thomas Thierauf*
Satisfiability Problems

96-07 *Vikraman Arvind, Jacobo Torán*
A nonadaptive NC Checker for Permutation Group Intersection

96-08 *David Cyrluk, Oliver Möller, Harald Rueß*
An Efficient Decision Procedure for a Theory of Fix-Sized Bitvectors with Composition and Extraction

96-09 *Bernd Biechele, Dietmar Ernst, Frank Houdek, Joachim Schmid, Wolfram Schulte*
Erfahrungen bei der Modellierung eingebetteter Systeme mit verschiedenen SA/RT–Ansätzen

96-10 *Falk Bartels, Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Formalizing Fixed-Point Theory in PVS

96-11 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Mechanized Semantics of Simple Imperative Programming Constructs

96-12 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Generic Compilation Schemes for Simple Programming Constructs

96-13 *Klaus Achatz, Helmuth Partsch*
From Descriptive Specifications to Operational ones: A Powerful Transformation Rule, its Applications and Variants

97-01 *Jochen Messner*
Pattern Matching in Trace Monoids

97-02 *Wolfgang Lindner, Rainer Schuler*
A Small Span Theorem within P

97-03 *Thomas Bauer, Peter Dadam*
A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration

97-04 *Christian Heinlein, Peter Dadam*
Interaction Expressions - A Powerful Formalism for Describing Inter-Workflow Dependencies

97-05 *Vikraman Arvind, Johannes Köbler*
On Pseudorandomness and Resource-Bounded Measure

97-06 *Gerhard Partsch*
Punkt-zu-Punkt- und Mehrpunkt-basierende LAN-Integrationsstrategien für den digitalen Mobilfunkstandard DECT

97-07 *Manfred Reichert, Peter Dadam*
$ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Loosing Control

97-08 *Hans Braxmeier, Dietmar Ernst, Andrea Mößle, Heiko Vogler*
The Project NoName - A functional programming language with its development environment

97-09 *Christian Heinlein*
Grundlagen von Interaktionsausdrücken

97-10 *Christian Heinlein*
Graphische Repräsentation von Interaktionsausdrücken

97-11 *Christian Heinlein*
Sprachtheoretische Semantik von Interaktionsausdrücken

97-12 *Gerhard Schellhorn, Wolfgang Reif*
Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers

97-13 *Dietmar Ernst, Frank Houdek, Wolfram Schulte, Thilo Schwinn*
Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme

97-14 *Wolfgang Reif, Gerhard Schellhorn*
Theorem Proving in Large Theories

97-15 *Thomas Wennekers*
Asymptotik rekurrenter neuronaler Netze mit zufälligen Kopplungen

97-16 *Peter Dadam, Klaus Kuhn, Manfred Reichert*
Clinical Workflows - The Killer Application for Process-oriented Information Systems?

97-17 *Mohammad Ali Livani, Jörg Kaiser*
EDF Consensus on CAN Bus Access in Dynamic Real-Time Applications

97-18 *Johannes Köbler,Rainer Schuler*
Using Efficient Average-Case Algorithms to Collapse Worst-Case Complexity Classes

98-01 *Daniela Damm, Lutz Claes, Friedrich W. von Henke, Alexander Seitz, Adelinde Uhrmacher, Steffen Wolf*
Ein fallbasiertes System für die Interpretation von Literatur zur Knochenheilung

98-02 *Thomas Bauer, Peter Dadam*
Architekturen für skalierbare Workflow-Management-Systeme - Klassifikation und Analyse

98-03 *Marko Luther, Martin Strecker*
A guided tour through *Typelab*

98-04 *Heiko Neumann, Luiz Pessoa*
Visual Filling-in and Surface Property Reconstruction

98-05 *Ercüment Canver*
Formal Verification of a Coordinated Atomic Action Based Design

98-06 *Andreas Küchler*
On the Correspondence between Neural Folding Architectures and Tree Automata

98-07 *Heiko Neumann, Thorsten Hansen, Luiz Pessoa*
Interaction of ON and OFF Pathways for Visual Contrast Measurement

98-08 *Thomas Wennekers*
Synfire Graphs: From Spike Patterns to Automata of Spiking Neurons

98-09 *Thomas Bauer, Peter Dadam*
Variable Migration von Workflows in *ADEPT*

98-10 *Heiko Neumann, Wolfgang Sepp*
Recurrent V1 – V2 Interaction in Early Visual Boundary Processing

98-11 *Frank Houdek, Dietmar Ernst, Thilo Schwinn*
Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment

98-12 *Gerhard Schellhorn*
Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers

98-13 *Gerhard Schellhorn, Wolfgang Reif*
Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers

98-14 *Mohammad Ali Livani*
SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN

98-15 *Mohammad Ali Livani, Jörg Kaiser*
Predictable Atomic Multicast in the Controller Area Network (CAN)

99-01 *Susanne Boll, Wolfgang Klas, Utz Westermann*
A Comparison of Multimedia Document Models Concerning Advanced Requirements

99-02 *Thomas Bauer, Peter Dadam*
Verteilungsmodelle für Workflow-Management-Systeme - Klassifikation und Simulation

99-03 *Uwe Schöning*
On the Complexity of Constraint Satisfaction

99-04 *Ercument Canver*
Model-Checking zur Analyse von Message Sequence Charts über Statecharts

99-05 *Johannes Köbler, Wolfgang Lindner, Rainer Schuler*
Derandomizing RP if Boolean Circuits are not Learnable

99-06 *Utz Westermann, Wolfgang Klas*
Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets

99-07 *Peter Dadam, Manfred Reichert*
Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications. Paderborn, Germany, October 6, 1999, GI–Workshop Proceedings, Informatik '99

99-08 *Vikraman Arvind, Johannes Köbler*
Graph Isomorphism is Low for ZPP^{NP} and other Lowness results

99-09 *Thomas Bauer, Peter Dadam*
Efficient Distributed Workflow Management Based on Variable Server Assignments

2000-02 *Thomas Bauer, Peter Dadam*
Variable Serverzuordnungen und komplexe Bearbeiterzuordnungen im Workflow-Management-System ADEPT

2000-03 *Gregory Baratoff, Christian Toepfer, Heiko Neumann*
Combined space-variant maps for optical flow based navigation

2000-04 *Wolfgang Gehring*
Ein Rahmenwerk zur Einführung von Leistungspunktsystemen

2000-05 *Susanne Boll, Christian Heinlein, Wolfgang Klas, Jochen Wandel*
Intelligent Prefetching and Buffering for Interactive Streaming of MPEG Videos

2000-06 *Wolfgang Reif, Gerhard Schellhorn, Andreas Thums*
Fehlersuche in Formalen Spezifikationen

2000-07 *Gerhard Schellhorn, Wolfgang Reif (eds.)*
FM-Tools 2000: The 4th Workshop on Tools for System Design and Verification

2000-08 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Effiziente Durchführung von Prozessmigrationen in verteilten Workflow-Management-Systemen

2000-09 *Thomas Bauer, Peter Dadam*
Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

2000-10 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Adaptives und verteiltes Workflow-Management

2000-11 *Christian Heinlein*
Workflow and Process Synchronization with Interaction Expressions and Graphs

2001-01 *Hubert Hug, Rainer Schuler*
DNA-based parallel computation of simple arithmetic

2001-02 *Friedhelm Schwenker, Hans A. Kestler, Günther Palm*
3-D Visual Object Classification with Hierarchical Radial Basis Function Networks

2001-03 *Hans A. Kestler, Friedhelm Schwenker, Günther Palm*
RBF network classification of ECGs as a potential marker for sudden cardiac death

2001-04 *Christian Dietrich, Friedhelm Schwenker, Klaus Riede, Günther Palm*
Classification of Bioacoustic Time Series Utilizing Pulse Detection, Time and Frequency Features and Data Fusion

2002-01 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata

2002-02 *Walter Guttmann*
Deriving an Applicative Heapsort Algorithm

2002-03 *Axel Dold, Friedrich W. von Henke, Vincent Vialard, Wolfgang Goerigk*
A Mechanically Verified Compiling Specification for a Realistic Compiler

2003-01 *Manfred Reichert, Stefanie Rinderle, Peter Dadam*
A Formal Framework for Workflow Type and Instance Changes Under Correctness Checks

2003-02 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Supporting Workflow Schema Evolution By Efficient Compliance Checks

2003-03 *Christian Heinlein*
Safely Extending Procedure Types to Allow Nested Procedures as Values

2003-04	*Stefanie Rinderle, Manfred Reichert, Peter Dadam* On Dealing With Semantically Conflicting Business Process Changes.
2003-05	*Christian Heinlein* Dynamic Class Methods in Java
2003-06	*Christian Heinlein* Vertical, Horizontal, and Behavioural Extensibility of Software Systems
2003-07	*Christian Heinlein* Safely Extending Procedure Types to Allow Nested Procedures as Values (Corrected Version)
2003-08	*Changling Liu, Jörg Kaiser* Survey of Mobile Ad Hoc Network Routing Protocols)
2004-01	*Thom Frühwirth, Marc Meister (eds.)* First Workshop on Constraint Handling Rules
2004-02	*Christian Heinlein* Concept and Implementation of C+++, an Extension of C++ to Support User-Defined Operator Symbols and Control Structures
2004-03	*Susanne Biundo, Thom Frühwirth, Günther Palm(eds.)* Poster Proceedings of the 27th Annual German Conference on Artificial Intelligence
2005-01	*Armin Wolf, Thom Frühwirth, Marc Meister (eds.)* 19th Workshop on (Constraint) Logic Programming
2005-02	*Wolfgang Lindner (Hg.), Universität Ulm , Christopher Wolf (Hg.) KU Leuven* 2. Krypto-Tag – Workshop über Kryptographie, Universität Ulm
2005-03	*Walter Guttmann, Markus Maucher* Constrained Ordering
2006-01	*Stefan Sarstedt* Model-Driven Development with ACTIVECHARTS, Tutorial
2006-02	*Alexander Raschke, Ramin Tavakoli Kolagari* Ein experimenteller Vergleich zwischen einer plan-getriebenen und einer leichtgewichtigen Entwicklungsmethode zur Spezifikation von eingebetteten Systemen
2006-03	*Jens Kohlmeyer, Alexander Raschke, Ramin Tavakoli Kolagari* Eine qualitative Untersuchung zur Produktlinien-Integration über Organisationsgrenzen hinweg
2006-04	*Thorsten Liebig* Reasoning with OWL - System Support and Insights –
2008-01	*H.A. Kestler, J. Messner, A. Müller, R. Schuler* On the complexity of intersecting multiple circles for graphical display

2008-02 *Manfred Reichert, Peter Dadam, Martin Jurisch, Ulrich Kreher, Kevin Göser, Markus Lauer*
Architectural Design of Flexible Process Management Technology

2008-03 *Frank Raiser*
Semi-Automatic Generation of CHR Solvers from Global Constraint Automata

2008-04 *Ramin Tavakoli Kolagari, Alexander Raschke, Matthias Schneiderhan, Ian Alexander*
Entscheidungsdokumentation bei der Entwicklung innovativer Systeme für produktlinien-basierte Entwicklungsprozesse

2008-05 *Markus Kalb, Claudia Dittrich, Peter Dadam*
Support of Relationships Among Moving Objects on Networks

2008-06 *Matthias Frank, Frank Kargl, Burkhard Stiller (Hg.)*
WMAN 2008 – KuVS Fachgespräch über Mobile Ad-hoc Netzwerke

2008-07 *M. Maucher, U. Schöning, H.A. Kestler*
An empirical assessment of local and population based search methods with different degrees of pseudorandomness

2008-08 *Henning Wunderlich*
Covers have structure

2008-09 *Karl-Heinz Niggl, Henning Wunderlich*
Implicit characterization of FPTIME and NC revisited

2008-10 *Henning Wunderlich*
On span-P^{cc} and related classes in structural communication complexity

2008-11 *M. Maucher, U. Schöning, H.A. Kestler*
On the different notions of pseudorandomness

2008-12 *Henning Wunderlich*
On Toda's Theorem in structural communication complexity

2008-13 *Manfred Reichert, Peter Dadam*
Realizing Adaptive Process-aware Information Systems with ADEPT2

2009-01 *Peter Dadam, Manfred Reichert*
The ADEPT Project: A Decade of Research and Development for Robust and Fexible Process Support
Challenges and Achievements

2009-02 *Peter Dadam, Manfred Reichert, Stefanie Rinderle-Ma, Kevin Göser, Ulrich Kreher, Martin Jurisch*
Von ADEPT zur AristaFlow® BPM Suite – Eine Vision wird Realität "Correctness by Construction" und flexible, robuste Ausführung von Unternehmensprozessen

2009-03 *Alena Hallerbach, Thomas Bauer, Manfred Reichert*
Correct Configuration of Process Variants in Provop

2009-04 *Martin Bader*
On Reversal and Transposition Medians

2009-05 *Barbara Weber, Andreas Lanz, Manfred Reichert*
Time Patterns for Process-aware Information Systems: A Pattern-based Analysis

2009-06 *Stefanie Rinderle-Ma, Manfred Reichert*
Adjustment Strategies for Non-Compliant Process Instances

2009-07 *H.A. Kestler, B. Lausen, H. Binder H.-P. Klenk. F. Leisch, M. Schmid*
Statistical Computing 2009 – Abstracts der 41. Arbeitstagung

2009-08 *Ulrich Kreher, Manfred Reichert, Stefanie Rinderle-Ma, Peter Dadam*
Effiziente Repräsentation von Vorlagen- und Instanzdaten in Prozess-Management-Systemen

2009-09 *Dammertz, Holger, Alexander Keller, Hendrik P.A. Lensch*
Progressive Point-Light-Based Global Illumination

2009-10 *Dao Zhou, Christoph Müssel, Ludwig Lausser, Martin Hopfensitz, Michael Kühl, Hans A. Kestler*
Boolean networks for modeling and analysis of gene regulation

2009-11 *J. Hanika, H.P.A. Lensch, A. Keller*
Two-Level Ray Tracing with Recordering for Highly Complex Scenes

2009-12 *Stephan Buchwald, Thomas Bauer, Manfred Reichert*
Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen

2010-01 *Hariolf Betz, Frank Raiser, Thom Frühwirth*
A Complete and Terminating Execution Model for Constraint Handling Rules

Ulmer Informatik-Berichte
ISSN 0939-5091

Herausgeber:
Universität Ulm
Fakultät für Ingenieurwissenschaften und Informatik
89069 Ulm