A Language-Based Comparison of Extensions of Petri Nets with and without Whole-Place Operations

P. A. Abdulla[1], G. Delzanno[2], and L. Van Begin[3]

[1] Uppsala University, Sweden, parosh@it.uu.se
[2] Università di Genova, Italy, giorgio@disi.unige.it
[3] Université Libre de Bruxelles, Belgium, lvbegin@ulb.ac.be

Abstract. We use language theory to study the relative expressiveness of infinite-state models laying in between finite automata and Turing machines. We focus here our attention on well structured transition systems that extend Petri nets. For these models, we study the impact of whole-place operations like transfers and resets on nets with indistinguishable tokens and with tokens that carry data over an infinite domain. Our measure of expressiveness is defined in terms of the class of languages recognized by a given model using coverability of a configuration as accepting condition. Our main result is that, perhaps surprisingly, whole-place operations augment the expressive power of Petri nets only in the case of black indistinguishable tokens. The results of our analysis can be used to build a complete hierarchy of well-structured models like Petri nets, affine well-structured nets, lossy fifo channel systems, constrained multiset rewriting systems, and data nets.

Keywords Languages; Expressiveness; Well-structured systems; Verification.

1 Introduction

The class of *well-structured transition systems* (wsts) [10] includes several interesting examples of infinite-state models whose expressiveness lay in between that of finite automata and that of Turing machines. Some examples of wsts are *Petri nets* [10], *transfer* and *reset nets* [8], *lossy FIFO channel systems* (LCS) [4, 6], and *constrained multiset rewriting systems* (CMRS) [2]. Petri nets is a widely used model of concurrent systems defined by a finite set of places containing multisets of tokens and by a finite set of transitions that define the flow of tokens among places. Each transition consumes/produces a fixed number of tokens in each place. Transfer/reset nets extend Petri net with *whole-place operations* that operate simultaneously on all tokens in a given set of places. In a lossy FIFO channel system places are viewed instead as unreliable FIFO channels. Finally, CMRS can be viewed as an extension of Petri nets in which tokens carry natural numbers and transitions are guarded by constraints on data attached to tokens. For all the above mentioned models, the *coverability problem* is decidable [2, 4, 6, 8]. This decision problem is of great importance for verification of safety properties like mutual exclusion.

An interesting research question concerns the study of the relative expressiveness of well-structured models. For this purpose, it comes natural to use tools from language theory, i.e., to compare the languages generated by labeled transition systems that describe their operational semantics. Unfortunately, standard notions of acceptance like *reachability* of a configuration are not adequate to obtain a fine-grained classification of wsts models. For instance, with this notion of acceptance transfer/reset nets are equivalent to Turing machines.

As shown in [3, 9, 11], a finer classification of wsts can be obtained by considering the class of languages recognized with *coverability acceptance conditions* (c-languages for short). A classification of wsts based on c-languages is particularly interesting since it can be used to extend the applicability of a decision procedure for coverability (e.g. the symbolic backward reachability algorithm in [1]) from a particular wsts model to an entire class.

New contribution In this paper we use c-languages as a formal tool to study the impact of *whole-place operations* on the expressiveness of Petri nets with black indistinguishable tokens and with tokens that carry data over an ordered domain. For this purpose, we compare the expressiveness of Petri nets, LCS, and CMRS with that of *affine well-structured nets* (AWNs) [13], and *data nets* [14]. AWNs are a generalization of Petri nets and transfer/reset nets in which the firing of a transition is split into three steps: subtraction, multiplication, and addition of black tokens. Data nets can be viewed as a generalization of AWNs in which these steps are defined on tokens that carry data taken from an infinite, ordered domain. Conditions on data values can be used here to restrict the type of tokens on which apply whole-place operations. Although presented in a different style, a data net can be viewed as a CMRS enriched with whole-place operations. For all these models, our technical results are as follows.

We first show that AWNs are strictly more expressive than Petri nets and strictly less expressive than lossy FIFO channel systems. The proof of the second result exploits a non-trivial property of the class of c-languages recognized by AWNs based on Dickson's lemma [7].

We then show that, differently from nets with indistinguishable tokens, whole-place operations do not augment the expressive power of models in which tokens carry data taken from an ordered domain. The proof is based on a weak, effectively constructible encoding of data nets into CMRS that can be used to reduce the coverability problem from one model to the other. Weakness refers here to the fact that the CMRS encoding simulates a *lossy* version of data nets, i.e., data nets in which tokens may get lost. However this is enough to show that the two models define the same class of c-languages.

Our analysis has several interesting consequences. First, it can be used to give a strict classification of the expressiveness of a large class of wsts models taken from the literature. Furthermore, it shows that the symbolic backward reachability algorithm for solving the CMRS coverability problem given [2] can also be applied in presence of whole-place operations like transfer and reset of colored tokens. Finally, as discussed in the conclusions, our weak encoding of data nets into CMRS can naturally be adapted to extend the decidability of coverability to a more general definition of data nets transition than the one given in [14]. Our extensions include, for instance, generation of fresh values, a feature present in several models of concurrency like CCS and π-calculus.

Related work In [9, 11] the authors compare the relative expressiveness of Petri nets with reset, transfer, and non-blocking arcs. A classification of infinite-state systems in terms of decidable properties is presented in [12]. The classification is extended to well-structured systems in [5]. Both classifications do not include models like CMRS and data nets. A classification of the complexity of the decision procedures for coverability of different formulations of data nets is studied in [14]. In [3] we have compared CMRS with lossy FIFO channel systems and other weaker models like relational automata. However, we have not considered whole-place operations like those in AWNs and data nets. We believe that a comparative study of all these sophisticated models can be useful to find new applications of the theory of well-structured transition systems.

Preliminary Notions In this paper we consider extensions of finite automata defined by using labelled transition systems. A transition system $T = (S, R)$ consists of a set S of configurations and of a set R of transitions, where a transition $\xrightarrow{\rho} \subseteq S \times S$. A transition system T is said to be *well-structured* (wsts) with respect to a quasi ordering $\preceq$ on configurations iff the following conditions hold: (i) $\preceq$ is a *well-quasi ordering*, i.e., for any infinite sequence of configurations $\gamma_1\gamma_2\ldots\gamma_i\ldots$ there exist indexes $i < j$ such that $\gamma_i \preceq \gamma_j$; (ii) T is *monotonic*, i.e., for any $\xrightarrow{\rho} \in R$, if $\gamma_1 \preceq \gamma_2$ and $\gamma_1 \xrightarrow{\rho} \gamma_3$, then there exists γ_4 s.t. $\gamma_3 \preceq \gamma_4$ and $\gamma_2 \xrightarrow{\rho} \gamma_4$.

Given a wsts T, we label each transition in R either with a symbol ℓ from an alphabet Σ or with the empty word ϵ (*silent transition*). If we associate to a wsts T an *initial* configuration γ_0 and a *final* configuration γ_{acc}, the language recognized by T with coverability acceptance (c-language for short) is defined as follows:

$$L_c(T) = \{w \in \Sigma^* \mid \gamma_0 \stackrel{w}{\Longrightarrow} \gamma \text{ and } \gamma_{acc} \preceq \gamma\}$$

where $\gamma_0 \stackrel{w}{\Longrightarrow} \gamma$ denotes a finite sequence of application of transitions such that the concatenation of their labels produces the word w. We use $L_c(\mathcal{M})$ to denote the class of c-languages recognized by instances T of a given model $\mathcal{M}$ (e.g. Petri nets, transfer nets, etc.), i.e., $L_c(\mathcal{M}) = \{L \mid \exists S \in \mathcal{M},\ L = L_c(S)\}$.
Given a wsts $T = (S, R, \preceq)$ with labels in $\Sigma \cup \{\epsilon\}$, a *lossy* version of T is a wsts $T' = (S, R', \preceq)$ for which there exists a bijection $h : R \mapsto R'$ such that $\xrightarrow{\rho} \in R$ and $\xrightarrow{h(\rho)}$ have the same label, $\xrightarrow{\rho} \subseteq \xrightarrow{h(\rho)}$ and if $\gamma \xrightarrow{h(\rho)} \gamma'$, then $\gamma \xrightarrow{\rho} \gamma''$ with $\gamma' \preceq \gamma''$. In a lossy version of a wsts, the set of reachable configurations contains configurations that are smaller than those of the original model. The following lemma then holds.

Lemma 1. *For any lossy version T' of a wsts T, we have that $L_c(T) = L_c(T')$.*

2 Whole-place operations in nets with black tokens

In this section we use c-languages as a formal tool to compare the expressiveness of Petri nets, affine well-structured nets (AWNs) [13], and lossy FIFO channel systems (LCS) [4, 6]. AWNs are a generalization of Petri nets in which transitions admit *whole-place* operations, i.e., operations that operate simultaneously on the whole set of tokens in a given place. Examples of whole-place operations are reset (all tokens in a place

$$F_t = \begin{pmatrix} p & q \\ 1 & 0 \end{pmatrix} \qquad G_t = \begin{pmatrix} & p & q \\ p & 1 & 0 \\ q & 0 & 0 \end{pmatrix} \qquad H_t = \begin{pmatrix} p & q \\ 0 & 1 \end{pmatrix}$$

Fig. 1. An example of AWN transition.

are consumed) and transfer arcs (all tokens in a place are transferred to another place) [8]. Formally, an AWN consists of a finite set P of places and of a finite set T of transitions. As in Petri nets, AWN-configurations, called *markings*, are vectors in $\mathbb{N}^P$, i.e., finite multisets with symbols in P. A marking counts the current number of tokens in a given place in P. In the rest of the paper we use $[a_1, \ldots, a_n]$ to indicate a multiset with elements $a_1, \ldots, a_n$. Furthermore, for a marking M, we use $M(a)$ to denote the number of tokens in place a. Finally we use, $-$ and $+$ to denote multiset difference and union.

An AWN-transition t is defined by two vectors F_t and H_t in $\mathbb{N}^P$, and by a $\mathbb{N}^P \times \mathbb{N}^P$-matrix G_t. Intuitively, F_t defines a subtraction step (how many tokens to remove from each place), G_t defines a multiplication step (whole-place operations), and H_t defines an addition step (how many tokens are added to each place). t is enabled at marking M if $F_t \leq M$ where $\leq$ denotes marking (multiset) inclusion, i.e., $M \leq M$ iff $M(p) \leq M'(p)$ for each $p \in P$. The firing of t at a marking M amounts to the execution of the three steps in sequence. Formally, it produces a new marking $M' = ((M - F_t) \cdot G_t) + H_t$, where $\cdot$ denotes the multiplication of vector $(M - F_t)$ and matrix G_t. As an example, let $P = \{p, q\}$ and consider the transition t in Fig. 1. This transition removes a token from p and resets the number of tokens in q to 1. For instance, from the marking $M = [p, p, q, q, q]$, i.e., the vector $(2, 3) \in \mathbb{N}^P$, we obtain the new marking $M' = [p, q]$ defined by the vector $((2, 3) - (1, 0)) \cdot G_t + (0, 1) = (1 * 1 + 3 * 0, 1 * 0 + 3 * 0) + (0, 1) = (1, 1)$

As shown in [13], AWN are well-structured with respect to marking inclusion $\leq$. Petri nets are the subclass of AWNs in which G_t is the identity matrix, i.e., with no whole-place operations. In [9] the authors have shown that there exists a c-language $L \in L_c(transfer\ nets)$ such that $L \notin L_c(Petri\ nets)$. Since transfer nets are a special case of AWNs, we obtain the following property.

Proposition 1. $L_c(Petri\ nets) \subset L_c(AWN)$.

To obtain a sort of upper bound bound on the expressive power of nets with whole-place operations, we can consider nets in which places maintain some kind of order between their tokens as in *lossy FIFO channel systems* (LCS). A LCS is a tuple (Q, C, N, δ), where Q is a finite set of control states, C is a finite set of channels, N is a finite set of messages, δ is a finite set of transitions, each of which is of the form (q_1, Op, q_2) where $q_1, q_2 \in Q$, and Op is a mapping from channels to channel operations. For any $c \in C$ and $a \in N$, an operation $Op(c)$ is either a *send* operation $!a$, a *receive* operation $?a$, the *empty* test $\epsilon?$, or the *null* operation nop. A configuration γ is a pair (q, w) where $q \in Q$, and w is a mapping from C to N^* giving the content of each channel. The initial configuration γ_{init} of $\mathcal{F}$ is the pair (q_0, ε) where $q_0 \in Q$, and ε denotes the mapping that assigns the empty sequence ϵ to each channel. The (strong) transition relation (that defines the semantics of machines with *perfect* FIFO channels) is defined as follows: $(q_1, w_1) \xrightarrow{\sigma} (q_2, w_2)$ if and only if $\sigma = (q_1, Op, q_2) \in \delta$ such that if $Op(c) = !a$, then $w_2(c) = w_1(c) \cdot a$; if $Op(c) = ?a$, then $w_1(c) = a \cdot w_2(c)$; if $Op(c) = \epsilon?$ then $w_1(c) = \epsilon$ and $w_2(c) = \epsilon$; if $Op(c) = nop$, then $w_2(c) = w_1(c)$. Now let $\preceq_l$ be the well-quasi ordering on LCS configurations defined as: $(q_1, w_1) \preceq_l (q_2, w_2)$ iff $q_1 = q_2$ and $\forall c \in C : w_1(c) \preceq_w w_2(c)$, where $\preceq_w$ indicates the subword relation. We introduce then the weak transition relation $\xRightarrow{\sigma}$ that defines the semantics of LCS: we have $\gamma_1 \xRightarrow{\sigma} \gamma_2$ iff there exists γ_1' and γ_2' s.t. $\gamma_1' \preceq_l \gamma_1$, $\gamma_1' \xrightarrow{\sigma} \gamma_2'$, and $\gamma_2 \preceq_l \gamma_2'$. Thus, $\gamma_1 \xRightarrow{\sigma} \gamma_2$ means that γ_2 is reachable from γ_1 by first losing messages from the channels and reaching γ_1', then performing a transition, and, thereafter losing again messages from channels. As shown in [4, 6], LCS are well-structured w.r.t. $\preceq_l$. The following theorem then holds.

Theorem 1. $L_c(AWN) \subset L_c(LCS)$.

Proof. (1) We first prove the inclusion $L_c(AWN) \subseteq L_c(LCS)$. Assume an AWN $W = (P, T, F, G, H)$ with $P = \{p_1, \ldots, p_n\}$. We build a LCS $\mathcal{F} = (Q, C, N, \delta)$ such that $L_c(W) = L_c(\mathcal{F})$. The set of channels is defined as $C = P \cup P'$ where P' (auxiliary channels) contains a primed copy of each element in P. The set of messages N contains the symbol $\bullet$ (a representation of a black token). Assume that $q_0 \in Q$ is the initial state of $\mathcal{F}$. Then, a marking M is encoded as a LCS configuration $enc(M)$ with state q_0 and in which channel $p_i \in P$ contains the word $\bullet^{m_i}$ containing $m_i = M(p_i)$ occurrences of symbol $\bullet$ for $i \in \overline{n}$[4]
For each transition t with label ℓ, we need to simulate the three steps (subtraction, multiplication, and addition) that correspond to F_t, G_t and H_t. Subtraction and addition can be simulated in a straightforward way by removing/adding the necessary number of tokens from/to each channel. The multiplication step is simulated as follows. For each $i \in \overline{n}$, we first make a copy of the content of channel p_i in the auxiliary channel p_i'. Each copy is defined by repeatedly moving a symbol from p_i to p_i' and terminates when p_i becomes empty. When the copy step is terminated, we start the multiplication step. For each $i \in \overline{n}$, we remove a msg $\bullet$ from p_i' and add as many $\bullet$'s to channel p_j as specified by $G_t(p_i, p_j)$ for $j \in \overline{n}$. This step terminates when the channels $p_1', \ldots, p_n'$ are all empty. For an accepting AWN-marking M_f, the accepting LCS-configuration is such that the control state is q_0, each channel $p_i \in P$ contains $M(p_i)$ occurrences of message $\bullet$, and all channels in P' are empty.
The following properties then hold: *i)* We first notice that $M \leq M'$ iff $enc(M) \preceq_l enc(M')$; *ii)* Furthermore, if $M_0 \xRightarrow{w} M_1$ in W, then $enc(M_0) \xRightarrow{w} enc(M_1)$ in $\mathcal{F}$; *iii)* Finally, since $\bullet$ symbols may get lost in $\mathcal{F}$, if $enc(M_0) \xRightarrow{w} enc(M_1)$ then there exists M_2 such that $M_0 \xRightarrow{w} M_2$ and $M_1 \leq M_2$. Since we consider languages with coverability acceptance, $L_c(W) = L_c(\mathcal{F})$ immediately follows from properties *(i)*,*(ii)*, *(iii)* and Lemma 1.

(2) We prove now that $L_c(LCS) \not\subseteq L_c(AWN)$. For this purpose, we exhibit a language in $L_c(LCS)$ and prove that it cannot be recognized by any AWN.

Fix a finite alphabet $\Sigma = \{a, b, \sharp\}$ and let $\mathcal{L} = \{w\#w' \mid w \in \{a, b\}^* \text{ and } w' \preceq w\}$. It is easy to check a LCS that accepts the language L: we first put w in a lossy channel

[4] We use $\overline{n}$ as an abbreviation of $[1, \ldots, n]$.

and then remove one-by-one all of its messages. Thus, we have that $\mathcal{L} \in L_c(LCS)$. We now prove that there is no AWN that accepts $\mathcal{L}$. Suppose it is not the case and there exists a AWN N, with (say) n places, that recognizes $\mathcal{L}$ with initial marking M_{init} and accepting marking M_f.

For each $w \in \{a, b\}^*$, there is a marking M_w such that $M_{init} \stackrel{w\#}{\Longrightarrow} M_w \stackrel{w}{\Longrightarrow} M$ and $M_f \leq M$ (otherwise $w\#w$ would not be in $L_c(N)$). Consider the sequences $w_0, w_1, w_2, \ldots$ and $M_{w_0}, M_{w_1}, M_{w_2}, \ldots$ of words and markings defined as follows:

– $w_0 := b^n$;
– If $M_{w_i} = (m_1, \ldots, m_n)$ then $w_{i+1} := a^{m_1}\, b\, a^{m_2}\, b \cdots b\, a^{m_n}$, for $i = 0, 2, \ldots$

We observe that (a) $w_0 \not\preceq w_i$ for all $i > 0$, since w_i contains n occurrences of b, while w_i contains only $n-1$ occurrences of b; and (b) for any $i < j$, $M_{w_i} \leq M_{w_j}$ iff $w_{i+1} \preceq w_{j+1}$. By Dickson's lemma [7], there are $i < j$ such that $M_{w_i} \leq M_{w_j}$. Without loss of generality, we can assume that j is the smallest natural number satisfying this property. Remark that we have that $w_i \not\preceq w_j$. Indeed, $w_0 \not\preceq w_j$ for any $j > 0$ by (a), and in the case of $i > 0$ we have by (b) that $w_i \not\preceq w_j$ since $M_{w_{i-1}} \not\leq M_{w_{j-1}}$. Since $M_{w_i} \leq M_{w_j}$, by monotonicity of AWNs, we have that $M_{w_i} \stackrel{w_i}{\Longrightarrow} M$ with $M_f \leq M$ implies that $M_{w_j} \stackrel{w_i}{\Longrightarrow} M'$ with $M_f \leq M \leq M'$. Hence, $M_{init} \stackrel{w_j\#w_i}{\Longrightarrow} M'$ and $w_j\#w_i \in L_c(N) = \mathcal{L}$, which is a contradiction. $\square$

By combining Prop. 1 and Theorem 1 we obtain the following strict classification.

$$L_c(Petri\ nets) \subset L_c(AWN) \subset L_c(LCS)$$

As a corollary, we have that transfer/reset nets are strictly less expressive than LCSs.

3 Whole-place operations in nets with colored tokens

In this section we study the impact of whole-place operations on the expressiveness of well-structured colored Petri nets like CMRS [2] and data nets [14]. CMRS is an extension of Petri nets in which tokens are labelled with natural numbers. For a fixed number of places P, if we represent a token in place p with value v as the term $p(v)$, then CMRS configurations are nothing but multisets of ground terms like $[p(1), p(3), q(4)]$ (we recall that markings are multisets over P, i.e., a special case of CMRS configurations). We use P-terms to denote terms associated to colored tokens. CMRS transitions are defined in terms of conditional multiset rewriting rules of the form $L \rightsquigarrow R : \Psi$ where L and R are terms with variables that describe colored tokens and Ψ is a condition over such variables. Conditions are expressed by a finite conjunction of constraints in the following form: $x + c < y$, $x \leq y$, $x = y$, $x < c$, $x > c$, $x = c$ where x, y are variables appearing in L and/or R and $c \in \mathbb{N}$ is a constant. A rule r is enabled at a configuration c if there exists a valuation of the variables Val ($Val(x) \in \mathbb{N}$) such that $Val(\Psi)$ is satisfied. Firing r at c leads to a new multi-set $c' = c - Val(L) + Val(R)$, where $Val(L)$, resp. $Val(R)$, is the multi-set of ground terms obtained from L, resp. R, by replacing each variable x by $Val(x)$. As an example, consider the CMRS rule:

$$\rho = [p(x)\ ,\ q(y)] \quad\rightsquigarrow\quad [q(z)\ ,\ r(x)\ ,\ r(w)]\ :\ \{x + 2 < y\ ,\ x + 4 < z\ ,\ z < w\}$$

$$s = \left(\begin{array}{cc|cc|cc|cc} \multicolumn{2}{c}{e_1} & \multicolumn{2}{c}{e_2} & \multicolumn{2}{c}{e_3} & \multicolumn{2}{c}{e_4} \\ p & q & p & q & p & q & p & q \\ 3 & 2 & 5 & 1 & 2 & 10 & 2 & 2 \end{array}\right) \quad s' = \left(\begin{array}{cc|cc|cc|cc} \multicolumn{2}{c}{e_1} & \multicolumn{2}{c}{e_2} & \multicolumn{2}{c}{e_3} & \multicolumn{2}{c}{e_4} \\ p & q & p & q & p & q & p & q \\ 29 & 28 & 5 & 1 & 25 & 1 & 2 & 2 \end{array}\right)$$

$$F_t = \left(\begin{array}{cc|cc|cc} \multicolumn{2}{c}{R_0} & \multicolumn{2}{c}{S_1} & \multicolumn{2}{c}{R_1} \\ p & q & p & q & p & q \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right) \quad H_t = \left(\begin{array}{cc|cc|cc} \multicolumn{2}{c}{R_0} & \multicolumn{2}{c}{S_1} & \multicolumn{2}{c}{R_1} \\ p & q & p & q & p & q \\ 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right)$$

$$G_t = \left(\begin{array}{cc|cc|cc|cc} & & \multicolumn{2}{c}{R_0} & \multicolumn{2}{c}{S_1} & \multicolumn{2}{c}{R_1} \\ & & p & q & p & q & p & q \\ \hline R_0 & p & 1 & 0 & 3 & 0 & 0 & 0 \\ & q & 3 & 1 & 0 & 0 & 0 & 0 \\ \hline S_1 & p & 0 & 0 & 1 & 0 & 0 & 0 \\ & q & 2 & 0 & 0 & 0 & 0 & 0 \\ \hline R_1 & p & 0 & 0 & 0 & 0 & 1 & 0 \\ & q & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

Fig. 2. Two data net markings (s and s') and a transition t with arity 1.

A valuation which satisfies the condition is $Val(x) = 1$, $Val(y) = 4$, $Val(z) = 8$, and $Val(w) = 10$. Thus, to fire t on $c = [p(1), p(3), q(4)]$ we first remove $p(1)$ and $q(4)$ and then add the new tokens $q(8)$, $r(1)$, and $r(1)$, producing the configuration $c' = [p(3), q(8), r(1), r(10)]$.

The coverability problem for CMRS is decidable for an ordering $\preceq_c$ that extends multi-inclusion by taking into consideration the relative "gaps" among the values on different tokens [2]. We come back to this point later.

It is important to remark that CMRS rules does not provide whole-place operations (the semantics is defined using rewriting applied to sub-multisets of tokens). Despite of it, in [3] we show that colors and gap-order conditions is enough to obtain a model that is strictly more powerful than LCS. By combining this property with Theorem 1, we have that $L_c(Petri\ nets) \subset L_c(AWN) \subset L_c(LCS) \subset L_c(CMRS)$.

A natural research question now is whether whole-place operations add power to models like CMRS or not. To answer this question, instead of defining a new version of CMRS, we compare its expressiveness with that of *data nets* [14]. *Data nets* are an extension of AWNs in which tokens are colored with data taken from a generic infinite domain D equipped with a linear ordering $\prec$ As discussed in [14], for coverability we can equivalently consider dense or discrete orderings. A data net has a finite sets of places P and transitions T. A data net marking s is a multiset of tokens that carry (linearly ordered) data in D, i.e., s is a finite sequence of vectors in $\mathbb{N}^P \setminus \{\mathbf{0}\}$, where $\mathbf{0}$ is the vector that contains only 0's. Each index i in the sequence s corresponds to some $d_i \in D$ (data values that occur in some token) such that $i \leq j$ if and only if $d_i \prec d_j$; $s(i)(p)$ is the number of tokens with data d_i in place p. In Fig. 2 we show two examples of configurations, namely, s and s', for a data net with places $P = \{p, q\}$. The data in D that occur in tokens in s and s' are $e_1 \prec e_2 \prec e_3 \prec e_4$.

Data net transitions like that in Fig. 2 are defined by vectors (of vectors) F_t and H_t and by a matrix G_t that define resp. subtraction, addition, and multiplication of colored tokens. These matrices are indexed by regions $R(\alpha_t) = (R_0, S_1, R_1, \ldots, S_k, R_k)$ associated to the arity $\alpha_t = k$ of the rule. The arity is used to select k data values $d_1, \ldots, d_k$ either fresh or occurring in the current configuration). Region S_i represents the singleton $\{d_i\}$. Regions R_i's are used to define whole-place operations (e.g. transfers) for tokens whose data are not in $\{d_1, \ldots, d_k\}$. R_0 contains all data $d : d \prec d_1$ in s, R_i contains all $d : d_i \prec d \prec d_{i+1}$ in s for $i : 1, \ldots, k-1$, and R_k contains all $d : d_k \prec d$ in s. To explain this idea, consider the marking s and the rule t of Fig. 2. t has arity 1, thus $R(\alpha_t) = \{R_0, S_1, R_1\}$. Let us assume that t (non-deterministically) partitions the data in s as follows $R_0 = \{e_1, e_2\}$, $S_1 = \{e_3\}$, and $R_1 = \{e_4\}$, its firing is defined as follows.

Subtraction: F_t specifies the number of tokens with data $d_1, \ldots, d_k$ that have to be removed, for each place in P, from the current configuration s. t is enabled if places have enough tokens to remove. In our example p contains two tokens with value e_3, and F_t specifies that one token with value e_3 must be removed. Thus, t is enabled in s. The subtraction step produces an intermediate configurations s_1 obtained from s by removing one token with data e_3 from place p.

Multiplication: G_t specifies whole-place operations on the regions in $R(\alpha_t)$. In our example the third column of G_t defines the effect of multiplication on the number of tokens with data e_3 in place p in s_1. Specifically, we add to the tokens in place p with value e_3 (1 in position S_1, p, S_1, p in G_t), three new tokens with value e_3 for each token with value in R_0 that lay into place p in s_1 (3 in position R_0, p, S_1, p in G_t). Thus, the total number of tokens with value e_3 in p becomes $(3 + 5) * 3 + 1 = 25$. Furthermore, since the fourth column has only zeroes, all tokens with data e_3 are removed from place q (a reset restricted to all tokens with value e_3 in q). The first column of G_t defines the effect on the tokens with values in R_0 in place p. Specifically, for each $d \in R_0$, we add to p: three tokens with value d for each token with the same value laying into q in s_1 (3 in position R_0, q, R_0, p in G_t); two tokens with data d for each token with data e_3 in q (2 in position S_1, q, R_0, p in G_t). Thus, the total number of tokens with value e_1 in p is now $3 + 3 * 2 + 2 * 10 = 29$ and that for value e_2 in p is now $5 + 3 * 1 + 2 * 10 = 28$. The other columns of G_t leave the same tokens as those in the corresponding regions and places in s_1. We use s_2 to refer to the resulting intermediate configuration.

Addition H_t specifies the number of tokens that are added, for each place, region, and data to the configuration s_2 to obtain the successor configuration s'. In our example, we simply add one token with data e_3 to place q. Finally, the new configuration s' is given in Fig. 2.

It is important to remark that whole-place operations are uniformly applied to each data value in a region. Whole-place operations between region R_i and R_j as well as subtractions from a region R_i are forbidden. Furthermore, in case of whole-place operations from R_i to S_j (or viceversa) tokens may change data value (e.g. all tokens with data $d \in R_i$ in p are moved to place q with value d_j), whereas in operations within a single region R_i tokens do not change data value.

As proved in [14], data nets are well-structured with respect to the well-quasi ordering $\preceq_d$ defined on markings as follows. Let $Data(s)$ be the set of data values that occur in a marking s. Then, $s_1 \preceq_d s_2$ iff there exists an injective function $h : Data(s_1) \mapsto Data(s_2)$ such that (i) h is monotonic and (ii) $s_1(d)(p) \leq s_2(h(d))(p)$ for each $d \in Data(s_1)$ and $p \in P$. In other words we compose subword ordering (condition (i)) with multiset inclusion (condition (ii)).

3.1 CMRS, Petri data nets, and Data nets

Data nets without whole place operations (i.e. in which G_t is the identity matrix) are called *Petri data nets*. Petri data nets defined on a domain with a single data value d are equivalent to Petri nets. Furthermore, as discussed in [14], it is possible to effectively build an encoding of CMRS into Petri data nets such that coverability in CMRS can be reduced to coverability into Petri data nets. Indeed, the well-quasi ordering $\preceq_c$ used in CMRS is basically the same as that used in Data nets (the only technical difference is due to the presence of constants in conditions of CMRS rules). Thus, we have that

$$L_c(CMRS) = L_c(Petri\ data\ nets) \subseteq L_c(Data\ nets)$$

We show next that the inclusion is not strict, and that Petri data nets, CMRS, and data nets have all the same expressive power. To prove this result, we have to show that for each Data nets $\mathcal{D}$ we can effectively build a Petri data net or a CMRS $\mathcal{S}$ such that $L_c(\mathcal{S}) = L_c(\mathcal{D})$. Since CMRS rules have a format similar to a (logic) programming language, we find more convenient to describe the encoding in CMRS.

Configurations Given a multi-set M with symbols in P and a value or variable x, we use M^x to denote the multi set of P-terms such that $M^x(p(x)) = M(p)$ (=number of occurrences of p in M) for each $p \in P$, and $M^x(p(y)) = 0$ for any $y \neq x$ and $p \in P$. Now assume an initial data net marking s_0 with data $d_1 \prec \ldots \prec d_n$. We build a CMRS representation of s_0 by non-deterministically selecting n natural numbers $v_1 < \ldots < v_n$ strictly included in some interval $[f, l]$. P-terms with parameter v_i represent tokens with data d_i in place p. Formally, we generate the representation of s_0 by adding to $\mathcal{S}$ a rule that rewrites an initial zerary term $init$ as follows:[5]

$$[init] \rightsquigarrow [first(f), last(l)] + \textstyle\sum_{i:1,\ldots,n} M_i^{x_i} \ : \ f < x_1 < \ldots < x_n < l \qquad (init)$$

where M_i is the multiset $s_0(d_i)$ for each $i \in \overline{n}$. The non-determinism in the choice of $f, l, x_1, \ldots, x_n$ make the CMRS representation of s_0 independent from specific parameters assumed by terms.

Transitions are encoded by CMRS rules that operate on the values in $[f, l]$ used in the representation of a marking. Most of the CMRS rule are based on left-to-right traversals of P-terms with parameters in $[f, l]$.

Subtraction Consider a transition t with $\alpha_t = k$. We first define a (silent) CMRS-rule that implements the subtraction step of t:

$$\begin{array}{ll} [first(f), last(l)] + F_t(S_1)^{x_1} + \ldots + F_t(S_k)^{x_k} \ \rightsquigarrow & (subtract) \\ \quad [\imath_0(f), \imath_1(x_1), ..., \imath_k(x_k), \imath_{k+1}(l), new_t] : \ f < x_1 < ... < x_k < l & \end{array}$$

In the $subtract$ rule we non-deterministically associate a value w_i to region S_i. The selection is performed by removing (from the current configuration) the multiset $F_t(S_i)^{x_i}$ that contains $F_t(S_i, p)$ occurrences of $p(x_i)$ for each $p \in P$. The association between

[5] We recall that $[t_1, \ldots, t_n]$ denotes a multisets of terms. Furthermore, $\sum_{i:1,\ldots,k} M_i = M_1 + \ldots + M_k$, where $+$ is multiset union.

value x_i and region S_i is maintained by storing x_i in a $\imath_i$-term (introduced in the right-hand side of the rule). If $F_t(S_i, p) = 0$ for any $p \in P$, then v_i may be associated to a data d_i not occurring in the current marking (i.e. selection of fresh data is a special case). Furthermore, by removing both the $first$- and the $last$-term, we disable the firing of rules that encode other data net transitions.

In the rest of the section we refer to The values $x_1, \ldots, x_k$ stored in $\imath_1$-,...,$\imath_k$-terms play the role of pointers to the regions $S_1, \ldots, S_k$ We refer to them as to the set of α_t-*indexes*. The parameters of terms in $[f, l]$ associated to the other regions $R_0, \ldots, R_k$ are called *region-indexes*.

Multiplication To simulate the multiplication step we proceed as follows. We first make a copy of the multiset of P-terms with parameters $v_1, \ldots, v_n$ in $[f, l]$ by copying each p-term with parameter v_i in a $\overline{p}$-term with parameter w_i such that $f' < w_1 < \ldots < w_n < l'$ and $[f', l']$ is an interval to the right of $[f, l]$, i.e., $l < f'$. The new_t-term in the $subtract$ rule is used to enable the set of (silent) CMRS rules in Section A in appendix that create the copy-configuration. During the copy we add a ✓-term for any *visited* region index. These terms are used to remember region indexes whose corresponding $\overline{P}$-terms are all removed in the multiplication step (e.g. when all tokens with data $d \in R_i$ are removed).

For instance, $[p(v_1), p(v_2), p(v_2), q(v_3)]$ with $f < v_1 < v_2 < v_3 < l$ is copied as $[\overline{p}(w_1), \checkmark(w_1), \overline{p}(w_2), \overline{p}(w_2), \checkmark(w_2), \overline{q}(w_3)\checkmark(w_3)]$ for some w_1, w_2, w_3 such that $f < l < f' < w_1 < w_2 < w_3 < l'$. The CMRS rules of Section A use a special term as a pointer scan the indexes in $[f, l]$ from left to right and create new $\overline{P}$-term with parameters in the interval $[f', l']$. The pointer is non-deterministically moved to the right. Thus during the traversal we may forget to copy some token. This is the first type of loss we find in our encoding. Notice that lost tokens have parameters strictly smaller that f'.

The simulation of the multiplication step operates on the copy-configuration only (that with $\overline{P}$-terms). The (silent) CMRS rules that implement this step are shown in Section A in appendix. The intuition behind their definition is as follows.

We first consider all α_t-indexes of $\overline{P}$-terms from left to right. For each α_t-index v_i, we proceed as follows. We first select and remove a term $\overline{p}(v_i)$ (encoding a given token). We compute then the effect of the whole-place operation on the entire set of α_t-indexes (including v_i itself). More specifically, for an α_t-index v_j we add $G_t(S_i, p, S_j, q)$ occurrences of the term $q(v_j)$ to the current CMRS configuration. The use of P- and $\overline{P}$-terms with parameters in the same interval allows us to keep track of tokens still to transfer ($\overline{P}$-terms) and tokens already transferred (P-terms). We then consider all remaining indexes by means of a left-to-right traversal of region-indexes in the current configuration. During the traversal, we add new P-terms with region-indexes as parameters as specified by G_t. During this step, we may forget to transfer some $\overline{P}$-term. This is the second type of loss we find in the encoding. After this step we either consider the next token with α_t-index v_i or we move to the next α_t-index.

After the termination of the whole-place operations for terms with α_t-indexes, we have to simulate the transfer of $\overline{P}$-terms with region-indexes. For each such an index, we transfer tokens within the same region-index or to an α_t-index. To simulate these operations we scan region-indexes from left-to-right to apply the matrix G_t. Furthermore, we mark visited region-indexes using ✓-terms. The ✓-terms are used in the simulation of the addition step. The (silent) CMRS rules that implement this step (enabled by the by term trR_t) are shown in Section A in appendix.

As a last step we add tokens to α_t-indexes and visited region indexes as specified by H_t. For α_t-indexes, we need a single rule that applies the matrix H_t. For region-indexes, we traverse from left-to-right the current configuration and apply H_t to each marked (with a ✓-term) region-index w. As mentioned before, the ✓-term allows us to apply H_t to regions emptied by the multiplication step. The rules for this step (associated to terms add_t and $addR_t$) are shown in Section A. All the rules are silent except the last one whose label is the same as that of t.

During the traversal, we may ignore some (marked) region-index. This is the last type of loss in our encoding. The new configuration is the final result of the simulation of the transition. Due to the possible losses in the different simulation steps, we may get a representation of a data net configuration smaller than the real successor configuration.

To formalize the relation between a data net $\mathcal{D}$ and its CMRS encoding $\mathcal{E}(\mathcal{D})$, for a configuration s with data $d_1 \prec \ldots \prec d_k$ we use $s^{\boldsymbol{v}}$ to denote the CMRS representation with indexes $\boldsymbol{v} = (v_1, \ldots, v_k)$. For configurations s_0, s_1, s, s', we have that *(i)* if $s_0 \stackrel{w}{\Longrightarrow} s_1$ in $\mathcal{D}$, then there exists $\boldsymbol{v}$ such that $[init] \stackrel{w}{\Longrightarrow} {s_1}^{\boldsymbol{v}}$ in $\mathcal{E}(\mathcal{D})$. Furthermore, *(ii)* if $[init] \stackrel{w}{\Longrightarrow} c$ in $\mathcal{E}(\mathcal{D})$ and $s^{\boldsymbol{v}} \preceq_c c$ for some $\boldsymbol{v}$, then there exists s_1 such that $s_0 \stackrel{w}{\Longrightarrow} s_1$ in $\mathcal{D}$ with $s \preceq_d s_1$. Finally, suppose that the accepting data net marking is a sequence $M_1 \ldots M_k$ of k vectors (multi-sets) over $\mathbb{N}^P$. Then, we add a silent CMRS rule

$$[first(f), last(l)] + \sum_{i \in \{1,\ldots,k\}} M_i^{x_i} \rightsquigarrow [acc] \;:\; f < x_1 < x_2 < \ldots < x_k < l, x = 0$$

where acc is a fresh (with arity zero) predicate. By adding this rule, the accepting CMRS configuration can be defined as the singleton $[acc]$. From properties (i), (ii) and Lemma 1, we have the following result.

Theorem 2. $L_c(data\ nets) = L_c(CMRS)$

4 Conclusions

By combining the results in the present paper with the relation between LCS and CMRS describe in [3], we obtain the following classification of well-structured extensions of Petri nets

$$L_c(Petri\ nets) \subset L_c(aWSN) \subset L_c(LCS) \subset L_c(CMRS) = L_c(data\ nets)$$

This classification reveals a different impact of whole-place operations on nets with black and colored tokens: they augment the expressive power of basic models like Petri nets, but they can be simulated in extended models in which tokens carry ordered data.

Our analysis can also be applied to extend the scope of the decidability results given in [14]. For instance, in the semantics of data nets some of the k data values selected by a transition *may* be fresh (they do not occur in the current configuration). Our CMRS encoding of the substraction step can naturally be extended to rules in which some of the data must necessarily be fresh (i.e. distinct from all data occurring in the current configuration). For this purpose, before selecting the data values, we make a copy (in a new interval) of the current configuration. In the new configuration we non-deterministically mark (with a predicate) a value x distinct from the values used to represent tokens. After this preliminary step, we apply the subtraction phase by requiring that the value x is one of the selected ones (i.e. we need α_t rules for this last step). Similar extensions of the CMRS encoding of data nets can be used to relax some of the syntactic restrictions in the original definition of data nets. For instance, we can extend the CMRS encoding to simulate transfers between two different regions R_i and R_j (a type of transfer that is forbidden in data nets). The above mentioned extensions of data net transitions are still monotonic w.r.t. $\preceq_d$. Thus, a weak CMRS simulation immediately extends the decidability result for coverability to these more general forms of transitions with whole-place operations for colored tokens.

References

1. P. A. Abdulla, K. Čerāns, B. Jonsson, and T. Yih-Kuen. General decidability theorems for infinite-state systems. LICS 1996: 313–321.
2. P. A. Abdulla and G. Delzanno: On the Coverability Problem for Constrained Multiset Rewriting Systems. AVIS 2006 (ETAPS 2006 workshop), to appear in ENTCS. URL: http://www.disi.unige.it/person/DelzannoG/Papers/avis06.pdf
3. P. A. Abdulla, G. Delzanno, and L. Van Begin: Comparing the Expressive Power of Well-Structured Transition Systems. CSL 2007: 99-114
4. P. A. Abdulla and B. Jonsson: Verifying Programs with Unreliable Channels. Inf. Comput. 127(2): 91-101 (1996)
5. N. Bertrand and Ph. Schnoebelen: A short visit to the STS hierarchy. ENTCS 154(3): 59-69 (2006)
6. G. Cécé, A. Finkel, and S. Purushothaman Iyer: Unreliable Channels are Easier to Verify Than Perfect Channels. Inf. Comput. 124(1): 20-31 (1996)
7. L. E. Dickson: Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. Amer. Journal Math. 35: 413422
8. C. Dufourd, A. Finkel, and Ph. Schnoebelen: Reset Nets Between Decidability and Undecidability. ICALP 1998: 103-115
9. A. Finkel, G. Geeraerts, J.-F. Raskin, and L. Van Begin. On the ω-language expressive power of extended petri nets. TCS 356(3):374-386, 2006.
10. A. Finkel and Ph. Schnoebelen: Well-structured transition systems everywhere! TCS 256(1-2): 63-92 (2001)
11. G. Geeraerts, J.-F. Raskin, and L. Van Begin. Well-structured languages. Acta Informatica 44(3-4):249-288 (2007).
12. T. A. Henzinger, R. Majumdar, and J.-F. Raskin. A classification of symbolic transition systems. ACM Trans. Comput. Log. 6(1): 1-32 (2005)
13. A. Finkel, P. McKenzie, and C. Picaronny: A well-structured framework for analysing petri net extensions. Inf. Comput. 195(1-2): 1-29 (2004)
14. R. Lazic, T. C. Newcomb, J. Ouaknine, A. W. Roscoe, and J. Worrell: Nets with Tokens Which Carry Data. ICATPN 2007: 301-320

A Encoding of Data Nets into CMRS

(1) Silent CMRS rules for new_t The selection of k distinct data values, i.e., k α_t indexes, is defined by means of the following CMRS rules.

$$\begin{array}{l}
\textit{For } k = \alpha_t,\ i \in \{0,\ldots,k\},\ \textit{and any } p \in P:\\
\textit{Copy of indexes in } \alpha_t\\
[\imath_0(x_0),\ldots,\imath_{k+1}(x_{k+1}), new_t] \rightsquigarrow\\
\quad [\imath_0(x_0),\ldots,\imath_{k+1}(x_{k+1}), \jmath_0(x'_0),\ldots,\jmath_{k+1}(x'_{k+1}), \uparrow(x_0), \Uparrow(x'_0)] :\\
\quad\; x_{k+1} < x'_0 < \ldots < x'_{k+1}\\
\textit{Copy } p \textit{ to } \overline{p} \textit{ for } \alpha_t - \textit{indexes}\\
[\uparrow(x), \imath_i(x), \Uparrow(y), \jmath_i(y), p(x)] \rightsquigarrow [\uparrow(x), \imath_i(x), \Uparrow(y), \jmath_i(y), \overline{p}(y)] : true\\
\textit{Copy } p \textit{ to } \overline{p} \textit{ for region} - \textit{indexes}\\
[\imath_i(x), \uparrow(u), p(u), \imath_{i+1}(x'), \jmath_i(y), \Uparrow(v), \jmath_{i+1}(y')] \rightsquigarrow\\
\quad [\imath_i(x), \uparrow(u), \imath_{i+1}(x'), \jmath_i(y), \overline{p}(v), \Uparrow(v), \jmath_{i+1}(y')] : x < u < x', y < v < y'\\
\textit{Move pointers to the right}\\
[\uparrow(u), p(u'), \imath_{k+1}(x), \Uparrow(v), \jmath_{k+1}(y)] \rightsquigarrow\\
\quad [\uparrow(u'), \imath_{k+1}(x), \Uparrow(v'), p(v'), \checkmark(v'), \jmath_{k+1}(y)] : u < u' < x, v < v' < y\\
\textit{Terminate copy, replace current conf with new one}\\
[\imath_0(f), \imath_1(x_1), \ldots., \imath_k(x_k), \imath_{k+1}(l), \jmath_0(f'), \jmath_1(x'_1), \ldots., \jmath_k(x'_k), \jmath_{k+1}(l'), \uparrow(u), \Uparrow(v)] \rightsquigarrow\\
\quad [\imath_0(f'), \imath_1(x'_1), \ldots., \imath_k(x'_k), \imath_{k+1}(l'), tr_t] \ :\ true
\end{array}$$

(2) Silent CMRS rules for simulation of whole-place operations.
In the following rules $G_t(S_i, p, \pi)^x$ is the multiset that, for each $q \in P$, contains $G_t(S_i, p, \pi, q)$ occurrences of the term $q(x)$.

$$\begin{array}{l}
\textit{For } k = \alpha_t,\ i \in \{1,\ldots,k\}, j \in \{0,\ldots,k\},\ \textit{and } p \in P:\\
\textit{Start from first index}\\
[tr_t] \rightsquigarrow [tr_{t,1}] \ :\ true\\
\textit{Select a token from an index in } \alpha_t,\ \textit{apply } G_t \textit{ to other indexes}:\\
[\imath_0(x_0), \imath_1(x_1), \ldots, \imath_i(x_i), \ldots, \imath_k(x_k), \imath_{k+1}(x_{k+1}), \overline{p}(x_i), tr_{t,i}] \rightsquigarrow\\
\quad [\imath_0(x_0), \imath_1(x_1), \ldots, \imath_k(x_k), \imath_{k+1}(x_{k+1}), apply_{t,i,p}(x)] + \Sigma_{j=1}^{k} G_t(S_i, p, S_j)^{x_j} : x_0 < x < x_{k+1}\\
\textit{Apply } G_t \textit{ to indexes inside regions, move to the right}\\
[\imath_j(v), apply_{t,i,p}(u), \overline{p}(u), \imath_{j+1}(v')] \rightsquigarrow\\
\quad [\imath_j(v), apply_{t,i,p}(u'), \overline{p}(u), \imath_{j+1}(v')] + G_t(S_i, p, R_j)^u : v < u < v', u < u'\\
\textit{Terminate visit continue with next token}\\
[apply_{t,i,p}(u)] \rightsquigarrow [tr_{t,i}] \ :\ true\\
\textit{Move to next index}\\
[tr_{t,j}] \rightsquigarrow [tr_{t,j+1}] \ :\ true\\
\textit{Terminate transfer of tokens for indexes in } \alpha_t,\ \textit{start transfer of tokens of regions}\\
[\imath_0(f), tr_{t,k}] \rightsquigarrow [trR_t(f)] \ :\ true
\end{array}$$

(3) Silent CMRS rules for trR

The following rules model a transfer inside a region-index and from a region-index to α_t-indexes. We use here $G_t(R_i, p, \pi)^x$ to denote the multiset that, for each $q \in P$, contains $G_t(R_i, p, \pi, q)$ occurrences of the term $q(x)$.

$$\mathrm{For\ k} = \alpha_\mathrm{t},\ \mathrm{i} \in \{0, \ldots, \mathrm{k}\},\ \mathrm{and\ any\ p} \in \mathrm{P}:$$

Remove token and apply G_t to indexes inside regions

$$\begin{array}{l}[\imath_0(x_0), \imath_1(x_1), \ldots, \imath_i(x_i), \imath_{i+1}(x_{i+1}), \ldots, \imath_{k+1}(x_{k+1}), \overline{p}(u), trR_t(u)] \rightsquigarrow \\ \quad [\imath_0(x_0), \imath_1(x_1), \ldots, \imath_i(x_i), \imath_{i+1}(x_{i+1}), \ldots, \imath_{k+1}(x_{k+1}), trR_t(u)] \\ \quad\quad + G_t(R_i, p, R_i)^u + \Sigma_{j=1}^k G_t(R_i, p, S_j)^{x_j} : \ x_i < u < x_{i+1}\end{array}$$

Move pointers to the right

$$[trR_t(u), \imath_{k+1}(l)] \rightsquigarrow [trR_t(u'), \imath_{k+1}(l),] \ : \ u < u' < l$$

Terminate visit, move to addition step

$$[trR_t(u)] \rightsquigarrow [add_t] \ : \ true$$

(4) CMRS rules for add_t and $addR_t$

In the following rules, $H_t(\pi)^x$ is the multiset that, for each $p \in P$, contains $H_t(\pi, p)$ occurrences of the term $p(x)$ for any $\pi \in R(\alpha_t)$. All rules are silent except the last one.

$$\mathrm{For\ k} = \alpha_\mathrm{t},\ \mathrm{i} \in \{0, \ldots, \mathrm{k}\},\ \mathrm{and\ any\ p} \in \mathrm{P}:$$

Apply H_t to indexes in α_t

$$\begin{array}{l}[\imath_0(x_0), \imath_1(x_1), \ldots, \imath_k(x_k), add_t] \rightsquigarrow \\ \quad [\imath_0(x_0), \imath_1(x_1), \ldots, \imath_k(x_k), addR_t(x_0)] + \Sigma_{j=1}^k H_t(S_j)^{x_j} : true\end{array}$$

Apply H_t to an index inside a region and advance pointer

$$\begin{array}{l}[\imath_i(v), \imath_{i+1}(v'), addR_t(u), \checkmark(u)] \rightsquigarrow \\ \quad [\imath_i(v), \imath_{i+1}(v'), addR_t(u')] + H_t(R_i)^u : v < u < v', u < u'\end{array}$$

Terminate simulation of transition t

$$[\imath_0(x_0), \imath_1(x_1), \ldots, \imath_{k+1}(x_{k+1}), addR_t(u)] \overset{\lambda(t)}{\rightsquigarrow} [first(x_0), last(x_{k+1})] : true$$