Well-Abstracted Transition Systems: Application to FIFO Automata

Alain Finkel[1]
LSV, ENS Cachan & CNRS UMR 8643
61, avenue du Président Wilson
94235 Cachan cedex, France
E-mail: finkel@lsv.ens-cachan.fr

and

S. Purushothaman Iyer[2]
Dept of Computer Science
NC State University
Raleigh, NC 27615, USA
E-mail: purush@csc.ncsu.edu

and

Grégoire Sutre[1, 3]
LSV, ENS Cachan & CNRS UMR 8643
61, avenue du Président Wilson
94235 Cachan cedex, France
E-mail: sutre@lsv.ens-cachan.fr

Formal methods based on symbolic representations have been found to be very effective. In the case of infinite state systems, there has been a great deal of interest in *accelerations* – a technique for characterizing the result of iterating an execution sequence an arbitrary number of times, in a sound, but not necessarily complete, way. We propose the use of abstractions as a general framework to design accelerations. We investigate SemiLinear Regular Expressions (SLREs) as symbolic representations for FIFO automata. In particular, we show that (a) SLREs are *easy to manipulate*, (b) SLREs form the *core* of known FIFO symbolic representations, and (c) SLREs are *sufficient* to represent the effect of arbitrary iterations of a loop for FIFO automata with one channel.

Key Words: infinite state systems, abstraction, symbolic representation, acceleration, protocols, FIFO automata, regular expressions, flatness.

Extended version of *Well-Abstracted Transition Systems* published in CONCUR'2000 proceedings.
[1]Supported in part by FORMA, a project funded by DGA, CNRS and MENRT.
[2]Supported in part by ARO under grant DAAG55-98-1-03093.
[3]Partially supported by the NSF Theory grant CCR-9988172 and the NSF ITR grant CCR-0085949.

1. INTRODUCTION

Formal methods are now routinely applied in design and implementation of finite state systems, such as those that occur in VLSI circuits. It has also been applied fairly regularly in the design and implementation of network protocols. Based on the success of formal methods in reasoning about finite state systems [9] there has been a great deal of interest in reasoning about infinite state systems. Given that programs, as well as network protocols, are infinite state in nature there is a need for automatic techniques to extend the reach of formal methods to a much larger class.

Infinite state systems could, in general, be Turing-powerful. Consequently, in reasoning about any non trivial property of such systems we will have to contend with incompleteness. At least two approaches have been considered in the literature: (a) semi-computation of the set of reachable states [4, 5, 1, 26], and (b) computation of a superset of reachability set [14, 32]. A requirement common to both approaches is that an infinite set of reachable states (from some given initial state) be finitely described. Clearly, the finite description should be such that it admits questions of membership and emptiness to be answered effectively. However, given that the reachability set is explored in an iterative fashion, an even more important question is "how does one infer the existence of an infinite set of states in the reachability set? And how does one calculate it?". Techniques called *accelerations* or *meta*-transitions have been discussed in the literature [10, 4, 5, 1, 12, 20]. We focus in this paper on symbolic representations for the computation of the reachability set of FIFO automata — a finite control with multiple unbounded FIFO channels. To the best of our knowledge, Pachl uses for the first time regular expressions to represent infinite sets of channel contents [31]. In [17], linear regular expressions have been defined and used. Boigelot et al. chosed a deterministic finite automata based representation, namely *Queue-content Decision Diagrams* [4] and afterwards Bouajjani et al. added Pressburger formulas, namely *Constrained QDDs* [5]. Simple regular expressions have been introduced for lossy FIFO automata [1].

We propose to address the issue "what are symbolic representations?". In this paper, we show how symbolic representations and accelerations can be couched in terms of abstract interpretation [14], a powerful semantics-based technique for explaining data flow analysis. We present a generic algorithm which, given an abstraction of a labelled transition system and an acceleration, computes a symbolic tree. We illustrate the usefulness of this approach by exploring *Linear* and *SemiLinear Regular Expressions* (LREs and SLREs) as symbolic representations for FIFO automata. In particular, we show the following about SLREs:

- SLREs are *easy to manipulate*: indeed, SLREs are exactly regular languages of *polynomial density* [35]. This class enjoys good complexity properties. In particular, we prove that inclusion between two SLREs is in $\mathrm{NP} \cap \mathrm{coNP}$.

- SLREs form the *core* of known FIFO symbolic representations: more formally, a set of queue contents is SLRE representable iff it is both CQDD representable and QDD representable ($SLREs = QDDs \cap CQDDs$),

- SLREs are usually *sufficient* since for FIFO automata with one channel, an arbitrary iteration of a loop is SLRE representable. Moreover, several examples in the literature have a SLRE representable reachability set: the alternating bit protocol [31], the bounded retransmission protocol of Philips [1], the producer/consumer described in [4] and the connection/deconnection protocol [28].

We say that a labelled transition system is *flat* when its set of traces is included in a SLRE language. We give an algorithm which computes an exact symbolic reachability set of any *flat* labelled transition system whose abstraction has a sound and complete acceleration.

The road map for the paper is as follows: in Section 2 we introduce labelled transition systems, in Section 3 we discuss FIFO automata and symbolic representations based on SLREs. In Section 4 we rephrase the notions of symbolic representations and accelerations in the framework of abstractions. In Section 5 we discuss accelerations based on SLREs. Finally, in Section 6 we compare QDDs, CQDDs and SLREs.

2. LABELLED TRANSITION SYSTEMS

We write 2^S (resp. $\mathcal{P}_f(S)$) to denote the set of subsets (resp. finite subsets) of a set S. Let S^I be the set of I-indexed (I finite) vectors of elements of S. Given $\mathsf{w} \in S^I$, $i \in I$ and $s \in S$, define $\mathsf{w}[i := s]$ to be the vector w' that differs from w at the index i ($\mathsf{w}'[i] = s$), but is the same elsewhere. Any element $s \in S$ induces the vector $\mathsf{s} \in S^I$ defined by $\mathsf{s}[i] = s$ for all $i \in I$. An I-indexed vector $\mathsf{w} \in S^I$ may also be written as $< i \mapsto \mathsf{w}[i] >$.

A *quasi-ordered set* is a pair $(S, \preceq)$ where S is a set and $\preceq$ is a reflexive and transitive relation over S ; we denote by $\prec$ the relation over S defined by $a \prec b$ if $a \preceq b$ and $b \not\preceq a$. A quasi-ordered set $(S, \preceq)$ satisfies the *ascending chain condition* if there does not exist an infinite strictly increasing sequence $s_0 \prec s_1 \prec s_2 \cdots s_k \prec s_{k+1} \cdots$ in S.

Let Σ be an alphabet (a finite, non empty set). We write Σ^* for the set of all *finite words* (shortly *words*), and ε denotes the empty word. We denote by Σ^+ the set $\Sigma^* \setminus \{\varepsilon\}$. For two words $x, y \in \Sigma^*$, $x \cdot y$ (shortly written xy) is their *concatenation*. A word y is a *left factor* of a word x, written $y \leq x$, if $x = y \cdot z$ for some word z and moreover we write $z = y^{-1}x$. The pair $(\Sigma^*, \leq)$ is a quasi-ordered set. For any *language* $L \subseteq \Sigma^*$, we write $\mathrm{LF}(L) = \{y \in \Sigma^* \mid \exists\, x \in L, y \leq x\}$. We write Σ^ω for the set of all *infinite words*.

DEFINITION 2.1. A *labelled transition system* LTS is a triple $LTS = (\mathsf{S}, \Sigma, \rightarrow)$ where S is a set of states, Σ is a finite set a labels and $\rightarrow \subseteq \mathsf{S} \times \Sigma \times \mathsf{S}$ is a labelled transition relation.

For any state $s \in \mathsf{S}$, we write $s \xrightarrow{\varepsilon} s$. If $\sigma = l_1 l_2 \cdots l_k$ is a finite word in Σ^+ and $s, s' \in \mathsf{S}$, we write $s \xrightarrow{\sigma} s'$, and we say that σ is an *execution sequence*, when there exists a finite sequence of states $s_0, s_1, \ldots, s_k \in \mathsf{S}$ such that $s = s_0$, $s' = s_k$ and $s_0 \xrightarrow{l_1} s_1 \xrightarrow{l_2} s_2 \cdots s_{k-1} \xrightarrow{l_k} s_k$.

DEFINITION 2.2. Let $LTS = (\mathsf{S}, \Sigma, \rightarrow)$ be a labelled transition system and $X \subseteq \mathsf{S}$ be a subset of states. We write $Traces(LTS, X)$ for the set:

$$Traces(LTS, X) = \bigcup_{s \in X} \{\sigma \in \Sigma^* \mid \exists\, s' \in \mathsf{S}, s \xrightarrow{\sigma} s'\}$$

A *set of initial states* of LTS is any subset of S. Given two labelled transition systems with initial states (LTS_1, X_1) and (LTS_2, X_2), a *simulation* between (LTS_1, X_1) and (LTS_2, X_2) is any binary relation $R \subseteq \mathsf{S}_1 \times \mathsf{S}_2$ such that $X_1 \times X_2 \subseteq R$ and satisfying: whenever $(s_1, s_2) \in R$, if $s_1 \xrightarrow{l}_1 s_1'$ then $s_2 \xrightarrow{l}_2 s_2'$ for some s_2' such that $(s_1', s_2') \in R$. We say that (LTS_1, X_1) *simulates* (LTS_2, X_2) if there is a simulation between (LTS_1, X_1) and (LTS_2, X_2).

Note that the labelled transition relation $\rightarrow$ captures the system behavior as it moves from one state to another. Since the central problem we wish to discuss is based on set of states reachable from a given state s by the $\rightarrow$ relation we associate with every labelled transition system LTS, two total functions $post : 2^{\mathsf{S}} \times \Sigma^* \rightarrow 2^{\mathsf{S}}$ and $post^* : 2^{\mathsf{S}} \rightarrow 2^{\mathsf{S}}$ defined by:

- $post(X, \sigma) = \{s' \in \mathsf{S} \mid \exists\, s \in X, s \xrightarrow{\sigma} s'\}$, and,
- $post^*(X) = \bigcup_{\sigma \in \Sigma^*} post(X, \sigma)$.

Observe that we don't enforce the labelled transition systems we consider to be finitely branching. In other words, $post(s_0, l)$ may be infinite for some single state $s_0 \in \mathsf{S}$ and for some label $l \in \Sigma$.

The main aim of our paper is a *fast computation* of $post^*(s_0)$, for a given $s_0 \in \mathsf{S}$. However, in general, $post^*(s_0)$ could be infinite (and even not recursive). To deal with this problem *symbolic representations* have been used [4, 5, 12]. Typically, assumptions are made about what properties these symbolic representations should satisfy. In section 4, we will discuss a minimal set of assumptions on these abstractions (*aka* symbolic representations) and how they relate to transition systems.

3. FIFO SYMBOLIC REPRESENTATIONS TO COMPUTE $POST^*$

Let us present the very well-known Alternating Bit Protocol. It is generally modelled by a system of 2 extended automata which communicate through 2 one directional FIFO channels. It is clear that we may always compute the cartesian product of the 2 extended automata, and this yields a single extended automaton with 2 bi-directional FIFO channels, namely a *FIFO automaton*.

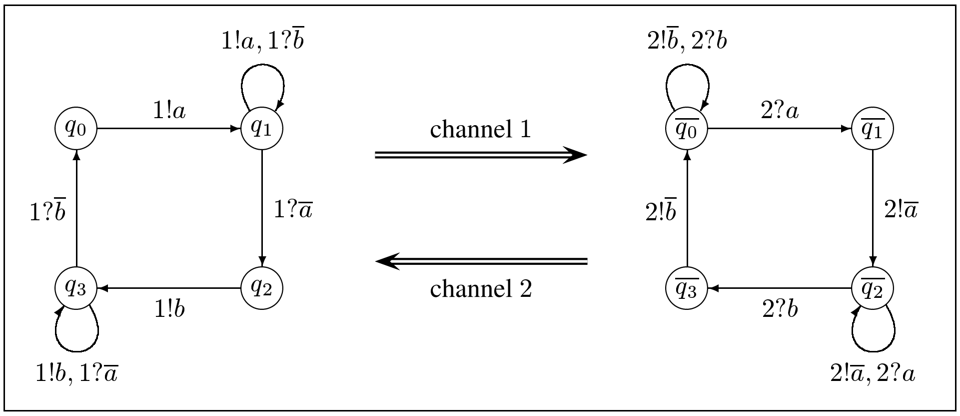

FIG. 1. A FIFO automaton ABP modeling the Alternating Bit Protocol

FIFO automata — a finite control with a set of FIFO channels — is a Turing powerful [7] mathematical model for the protocol specification languages Estelle and SDL. Even one channel FIFO automata can simulate Turing Machines and hence we cannot expect to find an algorithm computing exactly $post^*$. However, we will define a general semi-algorithm using symbolic representations, which computes exactly $post^*$ (or an over-approximation).

3.1. FIFO automata

Formally, a *FIFO automaton* is a 4-tuple $F = (Q, C, M, \delta)$ where

- Q is a finite set of *control states*,
- C is a finite set of *channel names*,
- M is a finite set of *message types*,
- $\delta \subseteq Q \times (C \times \{?, !\} \times M) \times Q$ is a finite set of *control transitions*.

Transitions of the form $(q, c?m, q')$ denote receive actions and transitions of the form $(q, c!m, q')$ denote send actions. A *loop* in a FIFO automaton is any non empty word $\sigma = (q_0, l_0, q'_0)(q_1, l_1, q'_1) \cdots (q_k, l_k, q'_k) \in \delta^*$ such that $q'_k = q_0$ and for all $0 \leq i \leq k-1$, $q'_i = q_{i+1}$. Moreover, we say that σ is a loop *on* q_0.

Example 3.1. For ABP, we have for instance that $\sigma_1 = (q_1, \overline{q_0}) \xrightarrow{1!a} (q_1, \overline{q_0})$ and $\sigma_2 = (q_0, \overline{q_0}) \xrightarrow{1!a} (q_1, \overline{q_0}) \xrightarrow{1?\overline{a}} (q_2, \overline{q_0}) \xrightarrow{2!\overline{b}} (q_2, \overline{q_0}) \xrightarrow{1!b} (q_3, \overline{q_0}) \xrightarrow{1?\overline{b}} (q_0, \overline{q_0})$ are loops.

We assume that the channels are perfect and that messages sent by a sender are received by a receiver in the same order they are sent. The operational semantics of a FIFO automaton is given by the following labelled transition system.

Definition 3.1. A FIFO automaton $F = (Q, C, M, \delta)$ defines the labelled transition system $LTS(F) = (\mathsf{S}, \Sigma, \rightarrow)$ as follows:

- $\mathsf{S} = Q \times (M^*)^C$, the set of states containing a control state q and a C-indexed vector of words w denoting the channel contents.
- $\Sigma = \delta$.
- $(q, \mathsf{w}) \xrightarrow{(p,c?m,p')} (q', \mathsf{w}')$ provided $q = p$, $q' = p'$ and $\mathsf{w} = \mathsf{w}'[c := m\mathsf{w}'[c]]$.
- $(q, \mathsf{w}) \xrightarrow{(p,c!m,p')} (q', \mathsf{w}')$ provided $q = p$, $q' = p'$ and $\mathsf{w}' = \mathsf{w}[c := \mathsf{w}[c]m]$.

For convenience, we write $(q, \mathsf{w}) \xrightarrow{c?m} (q', \mathsf{w}')$ (resp. $(q, \mathsf{w}) \xrightarrow{c!m} (q', \mathsf{w}')$) when we have $(q, \mathsf{w}) \xrightarrow{(q,c?m,q')} (q', \mathsf{w}')$ (resp. $(q, \mathsf{w}) \xrightarrow{(q,c!m,q')} (q', \mathsf{w}')$).

Example 3.2. For ABP, we have for instance: $(q_0, \overline{q_0}; \varepsilon, \varepsilon) \xrightarrow{1!a} (q_1, \overline{q_0}; a, \varepsilon) \xrightarrow{1!a} (q_1, \overline{q_0}; aa, \varepsilon) \xrightarrow{2?a} (q_1, \overline{q_1}; a, \varepsilon) \xrightarrow{2!\overline{a}} (q_1, \overline{q_2}; a, \overline{a}) \xrightarrow{2!\overline{a}} (q_1, \overline{q_2}; a, \overline{a}\overline{a}) \xrightarrow{1?\overline{a}} (q_2, \overline{q_2}; a, \overline{a})$.

3.2. Linear regular expressions

Since $post^*$ may be infinite for FIFO automata, we need to finitely represent infinite sets of states in order to compute $post^*$. Moreover, the unboundedness of $post^*$ arises from the unbounded channels (the set of control states is finite). Thus, a natural way to proceed is to finitely represent infinite sets of channel contents, i.e. vectors of words.

Example 3.3. For ABP, we have:

$$\begin{array}{rlcl}
post^*(q_0, \overline{q_0}; \varepsilon, \varepsilon) = & (q_0, \overline{q_0}) \times (b^*, \overline{b}^*) & \cup & (q_1, \overline{q_0}) \times (b^* a\, a^*, \overline{b}^*) \\
\cup & (q_1, \overline{q_1}) \times (a^*, \overline{b}^*) & \cup & (q_1, \overline{q_2}) \times (a^*, \overline{b}^* \overline{a}\, \overline{a}^*) \\
\cup & (q_2, \overline{q_2}) \times (a^*, \overline{a}^*) & \cup & (q_3, \overline{q_2}) \times (a^* b\, b^*, \overline{a}^*) \\
\cup & (q_3, \overline{q_3}) \times (b^*, \overline{a}^*) & \cup & (q_3, \overline{q_0}) \times (b^*, \overline{a}^* \overline{b}\, \overline{b}^*)
\end{array}$$

In the rest of the paper, we write M for a finite set of message types and C for a finite set of channel names. Following Pachl, we will use regular expressions to represent infinite sets of FIFO channel contents. We write $REG(M)$ for the set of *regular expressions (REGs)* over M. We denote by $[\![\rho]\!]$ the language associated to a regular expression ρ, and we write $\emptyset$ for the regular expression denoting an empty set of words.

To compactly represent the result of receiving a message from the front of a channel we will use the notion of derivatives [8]. Formally, given a message $m \in M$, we define the derivative operator $m^{-1} : REG(M) \to REG(M)$ as follows:

$$m^{-1}(\rho) = \begin{cases} \varepsilon & \text{if } \rho = m \\ \emptyset & \text{if } \rho = \emptyset \text{ or if } \rho = m' \in M \text{ with } m' \neq m \\ m^{-1}(\rho_1)\rho_2 & \text{if } \rho = \rho_1\rho_2 \text{ and } \varepsilon \notin [\![\rho]\!] \\ m^{-1}(\rho_2) + m^{-1}(\rho_1)\rho_2 & \text{if } \rho = \rho_1\rho_2 \text{ and } \varepsilon \in [\![\rho]\!] \\ m^{-1}(\rho_1) + m^{-1}(\rho_2) & \text{if } \rho = \rho_1 + \rho_2 \\ m^{-1}(\rho_1)\rho_1^* & \text{if } \rho = \rho_1^* \end{cases}$$

PROPOSITION 3.1. *For every $\rho \in REG(M)$, we have $[\![m^{-1}(\rho)]\!] = \{x \mid mx \in [\![\rho]\!]\}$.*

Let us define the two following subclasses of regular expressions:

- A *linear regular expression (written LRE)* is any regular expression of the form $x_0{y_0}^*x_1{y_1}^* \cdots x_{n-1}{y_{n-1}}^*x_n$ with $x_i \in M^*$ and $y_j \in M^+$.
- A *semilinear regular expression (written SLRE)* is any finite sum of LREs (possibly the empty sum $\emptyset$).

We write $LRE(M)$ (resp. $SLRE(M)$) for the set of linear (resp. semilinear) regular expressions over M.

Recall that the *density function* $d(L, n)$ counts the number of words of length n in a language L. It is shown in [35] that SLREs coincide with regular languages of *polynomial density* but it is stricly included in regular languages of star height one. SLREs enjoy good closure and complexity properties.

EXAMPLE 3.4. Note that $post^*$ for the Alternating Bit Protocol is composed of 16 LREs. There are other examples in the literature where $post^*$ can be described by SLREs, namely the bounded retransmission protocol of Philips [1], the producer/consumer described in [4] and the connection/deconnection protocol [28].

We define the *size* of LREs and SLREs as follows:

- the size $|\rho|$ of a LRE $\rho = x_0{y_0}^*x_1{y_1}^* \cdots x_{n-1}{y_{n-1}}^*x_n$ is given by $|\rho| = |x_n| + \sum_{i=0}^{n-1}(|x_i| + |y_i| + 1)$, and,
- the size $|\rho|$ of a SLRE $\rho = \rho_0 + \rho_1 \cdots + \rho_m$ is given by $|\rho| = \sum_{i=0}^{m} |\rho_i|$.

This definition will allow us in particular to express a complexity result about SLREs in Section 6. The following proposition expresses several closure properties of SLREs that will be used in the rest of the paper.

PROPOSITION 3.2. *The class of SLRE languages is closed under union, under left factor and under intersection with regular languages.*

Proof. We only prove the last closure property. We show by induction on the size of ρ that for every LRE ρ and for every regular expression r, we have $[\![\rho]\!] \cap [\![r]\!] = [\![\rho']\!]$ for some SLRE ρ'. Most cases are routine and are omitted. Consider the last case when $\rho = y^* \cdot \rho''$ with $y \in \Sigma^+$ and suppose that the claim holds for ρ''. Let r be any regular expression. We have:

$$[\![y^* \cdot \rho'']\!] \cap [\![r]\!] = \bigcup_{i \in \mathbb{N}} y^i \cdot ([\![\rho'']\!] \cap [\![(y^i)^{-1}(r)]\!])$$

From the induction hypothesis, we get that for every $i \in \mathbb{N}$ we have $[\![\rho'']\!] \cap [\![(y^i)^{-1}(r)]\!] = [\![\rho_i]\!]$ for some SLRE ρ_i. Now it is readily seen that there exists $l, k \in \mathbb{N}$ with $l > k$ such that $[\![(y^l)^{-1}(r)]\!] = [\![(y^k)^{-1}(r)]\!]$. Hence, we get that $\rho_{h(l-k)+j} = \rho_j$ for every $j, h \in \mathbb{N}$ such that $j \geq k$. Thus we come to:

$$\begin{aligned}
[\![\rho]\!] \cap [\![r]\!] &= \bigcup_{i \in \mathbb{N}} y^i \cdot [\![\rho_i]\!] \\
&= \bigcup_{i=0}^{k-1} y^i \cdot [\![\rho_i]\!] \quad \bigcup_{j=k}^{l-1} \quad \bigcup_{h \geq 0} y^{h(l-k)+j} \cdot [\![\rho_{h(l-k)+j}]\!] \\
&= \bigcup_{i=0}^{k-1} y^i \cdot [\![\rho_i]\!] \quad \bigcup_{j=k}^{l-1} y^j (y^{l-k})^* \cdot [\![\rho_j]\!] \\
&= \left[\!\!\left[\sum_{i=0}^{k-1} y^i \cdot \rho_i \quad + \quad \sum_{j=k}^{l-1} y^j (y^{l-k})^* \cdot \rho_j \right]\!\!\right]
\end{aligned}$$

■

3.3. FIFO symbolic representations

In order to handle different kinds of symbolic representations for FIFO automata (based on QDDs [4], CQDDs [5] or SLREs), we now define formally what we mean by FIFO symbolic representation. We essentially require a FIFO symbolic representation to be closed under *symbolic reception* (written ??) and under *symbolic emission* (written !!), in order to symbolically compute $post^*$.

DEFINITION 3.2. A (C, M)-*FIFO symbolic representation* (shortly a *FIFO symbolic representation*) is a 6-tuple $\mathcal{F}_{C,M}$ (shortly $\mathcal{F}$) such that $\mathcal{F}_{C,M} = (\mathsf{F}, \perp, ??, !!, [\![\cdot]\!], +)$ where F is a set of symbolic channel contents, $\perp \in \mathsf{F}$, $?? : C \times M \times \mathsf{F} \to \mathsf{F}$, $!! : C \times M \times \mathsf{F} \to \mathsf{F}$, $[\![\cdot]\!] : \mathsf{F} \to 2^{(M^*)^C}$ and $+ : \mathsf{F} \times \mathsf{F} \to \mathsf{F}$ are four total functions satisfying for every $f, f' \in \mathsf{F}$, $c \in C$ and $m \in M$:

$$\begin{aligned}
[\![\perp]\!] &= \emptyset & (1)\\
[\![??(c, m, f)]\!] &= \{\mathsf{w}[c := m^{-1}\mathsf{w}[c]] \mid \mathsf{w} \in [\![f]\!], m \leq \mathsf{w}[c]\} & (2)\\
[\![!!(c, m, f)]\!] &= \{\mathsf{w}[c := \mathsf{w}[c]m] \mid \mathsf{w} \in [\![f]\!]\} & (3)\\
[\![f + f']\!] &= [\![f]\!] \cup [\![f']\!] & (4)
\end{aligned}$$

Note that we suppose that FIFO symbolic representations are closed under union and contain the empty set, but these assumptions essentially allow us to simplify notations.

We are now ready to show how a FIFO symbolic representation can be defined based on SLREs.

PROPOSITION 3.3. *Given $m \in M$ and a semilinear regular expression ρ, we have that $m^{-1}(\rho)$ is a semilinear regular expression.*

DEFINITION 3.3. Let $\mathcal{SLRE} = (\mathcal{P}_f(SLRE(M)^C), \emptyset, ??, !!, [\![\cdot]\!], \cup)$, where $??$, $!!$ and $[\![\cdot]\!]$ are three total functions defined as follows:

$$\begin{aligned} ??(c, m, \chi) &= \{\mathsf{r}[c := m^{-1}(\mathsf{r}[c])] \mid \mathsf{r} \in \chi\} \\ !!(c, m, \chi) &= \{\mathsf{r}[c := \mathsf{r}[c] \cdot m] \mid \mathsf{r} \in \chi\} \\ [\![\chi]\!] &= \bigcup_{\mathsf{r} \in \chi} \prod_{c \in C} [\![\mathsf{r}[c]]\!] \end{aligned}$$

PROPOSITION 3.4. *$\mathcal{SLRE}$ is a FIFO symbolic representation.*

Now, starting from a finite set of states, how to reach a "symbolic state" denoting an infinite set of states ? A natural way is to compute the effect of a loop, for instance iterating the loop σ_1 of ABP gives us: $(q_1, \overline{q_0}; \varepsilon, \varepsilon) \xrightarrow{(!a)^*} (q_1, \overline{q_0}; a^*, \varepsilon)$.

4. ABSTRACTION AND ACCELERATION OF LABELLED TRANSITION SYSTEMS

We present in this section our general framework of abstraction and acceleration formalizing this intuitive strategy of computing the effect of loops. This general setting is given for labelled transition system, and is then applied to FIFO automata.

4.1. Abstraction

We now introduce the notion of abstraction of labelled transition systems, a fairly general setting for our discussions. "What an abstraction is" is based on a minimal set of assumptions which allows us to present and compare several symbolic representations in an uniform setting. Formally,

DEFINITION 4.1. An *abstraction $\mathcal{A}_{LTS}$ of a labelled transition system LTS* $= (\mathsf{S}, \Sigma, \rightarrow)$ is a 5-tuple $\mathcal{A}_{LTS} = (\mathsf{A}, post_{\mathcal{A}}, \gamma, \sqcup, \Sigma)$ where A is a set of (abstract) states, $post_{\mathcal{A}} : \mathsf{A} \times \Sigma \rightarrow \mathsf{A}$, $\gamma : \mathsf{A} \rightarrow 2^{\mathsf{S}}$ and $\sqcup : \mathsf{A} \times \mathsf{A} \rightarrow \mathsf{A}$ are three total functions satisfying for every $a, b \in \mathsf{A}$ and $l \in \Sigma$:

$$\gamma(a \sqcup b) = \gamma(a) \cup \gamma(b), \text{ and,} \tag{5}$$

$$post(\gamma(a), l) \subseteq \gamma(post_{\mathcal{A}}(a, l)) \subseteq post^*(\gamma(a)) \tag{6}$$

An abstract state a denotes a potentially infinite set $\gamma(a)$ of (concrete) states. Condition (5) enforces the set of abstract states A to be closed under (abstract) union $\sqcup$. Condition (6) ensures that an abstract exact computation of $post^*$ can be performed, using the abstract $post$ function $post_{\mathcal{A}}$. Note that a labelled transition system may have several abstractions. In the rest of the paper, we write $\mathcal{A}_{LTS}$ to denote that $\mathcal{A}_{LTS}$ is an abstraction of a labelled transition system LTS.

An abstraction carries with it a natural quasi-ordering $\sqsubseteq$ induced by the function γ and defined by $a \sqsubseteq b$ if $\gamma(a) \subseteq \gamma(b)$. Furthermore, this quasi-ordering induces a natural equivalence relation $\cong$ defined by $a \cong b$ if $a \sqsubseteq b$ and $b \sqsubseteq a$. Note that two equivalent abstract states denote the same set of states.

The importance of the $\cong$ relation is that the $\sqcup$ operation is commutative and associative with respect to the equivalence. Hence for every finite subset X of A, $\sqcup X$ is well defined with respect to $\cong$. This allows us to compute $post^*$ in an iterative manner, where we take the unions (i.e., $\sqcup$) as we go along, and thus have to maintain only one element of the symbolic representation at any time in computing the reachability set. Note that on the contrary to the usual abstract interpretation framework [14], we are not assuming that the abstraction domain A is closed under arbitrary union (i.e., lubs) but only finite ones.

EXAMPLE 4.1. For any labelled transition system $LTS = (\mathsf{S}, \Sigma, \rightarrow)$, the 5-tuple $(2^{\mathsf{S}}, post, Id, \cup, \Sigma)$ is readily seen to be an abstraction of LTS. However, this trivial abstraction is *not effective* when S is infinitely large, since we cannot finitely represent any subset of S. The abstractions we will define later will be based on finitely describable (potentially infinite) subsets of S.

An abstraction implicitly defines a deterministic labelled transition system $(\mathsf{A}, \Sigma, post_{\mathcal{A}})$. A simple consequence of our definitions is that the notion of abstraction immediately gives us a simulation:

PROPOSITION 4.1. *Let $\mathcal{A}_{LTS} = (\mathsf{A}, post_{\mathcal{A}}, \gamma, \sqcup, \Sigma)$ be an abstraction of a labelled transition system $LTS = (\mathsf{S}, \Sigma, \rightarrow)$. For every $a_0 \in \mathsf{A}$, the labelled transition system $(\mathsf{A}, \Sigma, post_{\mathcal{A}})$ with initial state a_0 simulates LTS with the set of initial states $\gamma(a_0)$.*

Proof. Let R be the relation on $\mathsf{S} \times \mathsf{A}$ defined by $R(s, a)$ if $s \in \gamma(a)$. We prove that R is a simulation relation between $(\mathsf{A}, \Sigma, post_{\mathcal{A}})$ and LTS. Note that the initial states satisfy R. Now if $R(s, a)$ and $s \xrightarrow{l} s'$ then from (6), we get $R(s', a')$ where $a' = post_{\mathcal{A}}(a', l)$. ■

4.2. Abstraction of FIFO automata

A FIFO symbolic representation $\mathcal{F} = (\mathsf{F}, \bot, ??, !!, [\![\cdot]\!], +)$ allows us to associate a canonical abstraction to any FIFO automaton F, as follows.

DEFINITION 4.2. Let $\mathcal{A}^{\mathcal{F}}_{LTS(F)} = (\mathsf{A}, post_{\mathcal{A}}, \gamma, \sqcup, \Sigma)$, where $post_{\mathcal{A}} : \mathsf{A} \times \Sigma \rightarrow \mathsf{A}$, $\gamma : \mathsf{A} \rightarrow 2^{\mathsf{S}}$ and $\sqcup : \mathsf{A} \times \mathsf{A} \rightarrow \mathsf{A}$, be defined by:

$$\begin{aligned}
\mathsf{A} &= \mathsf{F}^Q \\
post_{\mathcal{A}}(\mathsf{a}, (q, c?m, q')) &= \bot[q' := ??(c, m, \mathsf{a}[q])] \\
post_{\mathcal{A}}(\mathsf{a}, (q, c!m, q')) &= \bot[q' := !!(c, m, \mathsf{a}[q])] \\
\gamma(\mathsf{a}) &= \bigcup_{q \in Q} \{q\} \times [\![\mathsf{a}[q]]\!] \\
\sqcup(\mathsf{a}, \mathsf{b}) &= < q \mapsto \mathsf{a}[q] + \mathsf{b}[q] >
\end{aligned}$$

It is readily seen that for any FIFO symbolic representation $\mathcal{F}$ and for any FIFO automaton F, $\mathcal{A}^{\mathcal{F}}_{LTS(F)}$ is an abstraction of $LTS(F)$. Thus, we obtain that $\mathcal{A}^{\mathcal{SLRE}}_{LTS(F)}$ is an abstraction.

4.3. Acceleration

We will now discuss how several strategies alternately called acceleration or meta-transitions can be captured uniformly by our definitions. Without acceleration, the abstract state obtained at each step of the computation of $post^*(s_0)$ may represent only a finite portion of $post^*(s_0)$. Consequently, the advantage of using a symbolic representation might never be realized.

The way out of this dilemma is based on the observation that if an execution sequence σ can be iterated infinitely often from a state s, and if $\bigcup_{i\in\mathbb{N}} post(s, \sigma^i)$ can be calculated in a symbolic manner, then a portion of the reachability set containing an *infinite* set of states can be captured in a *single* step. The motivation is exactly what is behind the notion of *widening* in abstract interpretation [14], where in iterative data flow analysis of programs the effect of executing a loop (in a program) an arbitrary number of times is captured as a widening operator. In the following we adapt Cousot and Cousot's definition for our situation.

DEFINITION 4.3. A total function $\nabla : \mathsf{A}\times\Sigma^+ \rightarrow \mathsf{A}$ is an *acceleration* for an abstraction $\mathcal{A}_{LTS} = (\mathsf{A}, post_{\mathcal{A}}, \gamma, \sqcup, \Sigma)$ provided it satisfies the following condition for every $a \in \mathsf{A}$ and $\sigma \in \Sigma^+$:

$$\nabla(a, \sigma) \sqsupseteq a \tag{7}$$

An acceleration ∇ is *sound* if it satisfies the following condition for every $a \in \mathsf{A}$ and $\sigma \in \Sigma^+$:

$$\gamma(\nabla(a, \sigma)) \subseteq post^*(\gamma(a)) \tag{8}$$

An acceleration ∇ is *complete* if it satisfies the following condition for every $a \in \mathsf{A}$ and $\sigma \in \Sigma^+$:

$$\gamma(\nabla(a, \sigma)) \supseteq \bigcup_{i\in\mathbb{N}} post(\gamma(a), \sigma^i) \tag{9}$$

The three conditions in the definition above deserve some explanation. Condition (7) — an inclusion condition — ensures that the abstract state obtained after an acceleration is bigger. The definition of a *sound* acceleration, Condition (8), enforces the constraint that all states computed by an acceleration are included in $post^*$. Similarly, the definition of a *complete* acceleration, Condition (9), enforces the constraint that all states resulting from an arbitrary number of iterations of σ are included in the result of the acceleration.

Given an acceleration, we suggest developing a reachability tree in an uniform fashion according to the algorithm below.

Most of the details of this algorithm should be clear. The acceleration gets applied, if at all possible, to obtain a potentially infinite set of reachable states. We will now show that this tree captures the reachable states, as required:

PROPOSITION 4.2. *Let $\mathcal{A}_{LTS} = (\mathsf{A}, post_{\mathcal{A}}, \gamma, \sqcup, \Sigma)$ be an abstraction, $a_0 \in \mathsf{A}$ be an initial abstract state and ∇ be an acceleration for $\mathcal{A}_{LTS}$. If T denotes the (possibly*

ALGORITHM 1 $\mathtt{SymbolicTree}(\mathcal{A}_{LTS}, a_0, \nabla)$

Input: an abstraction $\mathcal{A}_{LTS} = (\mathsf{A}, post_{\mathcal{A}}, \gamma, \sqcup, \Sigma)$, an initial abstract state $a_0 \in \mathsf{A}$ and an acceleration ∇ for $\mathcal{A}_{LTS}$.

create root $\mathtt{r}$ labelled with a_0
while there are unmarked nodes **do**
  pick an unmarked node $\mathtt{n}:a$
  if $a \sqsubseteq \bigsqcup\{b \mid \mathtt{m}:b$ is a marked node$\}$ **then**
    skip {the node $\mathtt{n}$ is covered}
  else
    $b \leftarrow a$
    for each ancestor $\mathtt{m}$ of $\mathtt{n}$ **do**
      $b \leftarrow \nabla(b, \sigma)$ where $\mathtt{m} \xrightarrow{\sigma} \mathtt{n}$
    for each label $l \in \Sigma$ **do**
      $a' \leftarrow post_{\mathcal{A}}(b, l)$
      construct a son $\mathtt{n}':a'$ of $\mathtt{n}$ and label the arc $\mathtt{n} \xrightarrow{l} \mathtt{n}'$
  mark $\mathtt{n}$

infinite) tree constructed by the $\mathtt{SymbolicTree}$ *algorithm applied to* $\mathcal{A}_{LTS}$*,* a_0 *and* ∇ *then we have:*

$$post^*(\gamma(a_0)) \subseteq \bigcup_{\mathtt{n}:a \in T} \gamma(a)$$

If moreover ∇ *is a sound acceleration then we have:*

$$post^*(\gamma(a_0)) = \bigcup_{\mathtt{n}:a \in T} \gamma(a)$$

Proof. The first statement follows by an induction on the length of σ that for every $\sigma \in \Sigma^*$, $post(\gamma(a_0), \sigma) \subseteq \bigcup_{\mathtt{n}:a \in T} \gamma(a)$. The second assertion is proved by induction on the depth of a node in T that for every node $\mathtt{n}:a \in T$, we have $\gamma(a) \subseteq post^*(\gamma(a_0))$. ■

Remark. If $(\mathsf{A}, \sqsubseteq)$ satisfies the ascending chain condition then the $\mathtt{SymbolicTree}$ algorithm terminates for any initial abstract state $a_0 \in \mathsf{A}$ and for any acceleration ∇.

4.4. Acceleration of FIFO automata

We are now able to show how a generic acceleration can be defined for FIFO automata. The $\mathtt{SymbolicTree}$ algorithm suggests that we should accelerate an execution sequence σ if σ can be repeated infinitely often. In the case of FIFO automata, loops are the only execution sequences that may be repeated infinitely often, since the initial control state and the final control state should be the same. For any loop σ on some control state q, we define two total functions, also written σ and σ^*, from $2^{(M^*)^C}$ to $2^{(M^*)^C}$ as follows:

- $\sigma(X) = \{\mathsf{w} \in (M^*)^C \mid (q, \mathsf{w}) \in post(\{q\} \times X, \sigma)\}$, and,
- $\sigma^*(X) = \bigcup_{i \in \mathbb{N}} \sigma^i(X)$.

We will use the notion of $\mathcal{F}$*-representable* loops to define a canonical acceleration to any abstraction.

DEFINITION 4.4. A loop σ is $\mathcal{F}$*-representable* if for every $f \in \mathsf{F}$, there exists $g \in \mathsf{F}$ such that $[\![g]\!] = \sigma^*([\![f]\!])$.

When σ is $\mathcal{F}$-representable we write $g = \sigma^*(f)$ even if there is not a unique g such that $[\![g]\!] = \sigma^*([\![f]\!])$. Notice that $\sigma^*(f)$ is well defined with respect to $\cong$. The generic acceleration basically iterates a loop if it is $\mathcal{F}$-representable:

DEFINITION 4.5. Let $\nabla_{\mathcal{F}} : \mathsf{A} \times \Sigma^+ \to \mathsf{A}$ be defined by:

$$\nabla_{\mathcal{F}}(\mathsf{a}, \sigma) = \begin{cases} \mathsf{a}[q := \sigma^*(\mathsf{a}[q])] & \text{if } \sigma \text{ is a } \mathcal{F}\text{-representable loop on } q \\ \mathsf{a} & \text{otherwise} \end{cases}$$

We now show that $\nabla_{\mathcal{F}}$ is indeed an acceleration.

PROPOSITION 4.3. *The two following statements hold:*

1. the total function $\nabla_{\mathcal{F}}$ is a sound acceleration for $\mathcal{A}^{\mathcal{F}}_{LTS(F)}$, and,
2. If every loop of F is $\mathcal{F}$-representable then $\nabla_{\mathcal{F}}$ is complete.

4.5. Flat labelled transition systems and SLREs

We introduce *flat* (and ρ*-flat*) languages which are closely connected to SLREs.

DEFINITION 4.6. A language is *flat* if it is included in a SLRE language.

Note that a flat language is not necessarily regular. We will use, in the sequel, ρ-flat languages because we generally need to know a SLRE ρ such that $L = [\![\rho]\!]$.

DEFINITION 4.7. A language L is ρ*-flat* if $L \subseteq [\![\rho]\!]$, where ρ is a SLRE.

We recall that a language L is *bounded* if L is included in a language of the form $w_1^* \cdots w_k^*$ where each w_i is a word [23, 27]. Let us note that the complexity of the equivalence and containment problems between bounded regular languages has been studied in [27].

Remark. A language is flat iff it is bounded.

The notions of flat languages, flat automata and flat formulas appear in different (past and recent) papers about FIFO automata, timed automata, flat counter automata, flat temporal logics and computation of reachability sets by using flat languages. More precisely, let us recall that:

• *monogeneous* and *linear* FIFO automata are such that the projection of traces on emissions in each channel is a particular flat language, formally a ρ-flat language where:

monogeneous ρ is a finite sum of LREs of the form uv^* where u and v are words.

linear ρ is a LRE of the form $a^*b^*c^* \cdots z^*$ where $a, b, c \ldots, z$ are letters.

Reachability is decidable for monogeneous FIFO automata [30] and for linear FIFO automata [24, 18].

• Any timed automaton (a la Alur & Dill) can be translated into an extended *flat counter machine* (for each control state q, at most one loop of the machine on q may modify the counter value) [13]. An extension of LTL enabling to talk about counters has been defined and studied in [11]. The formulas of this extension are said *flat* because on the left of an until, only atomic constraints or propositionnal LTL formulas are allowed. The models of this new logic are recognized by flat counter machines.

• In [21, 22], languages of the form $w_1^* \cdots w_k^*$ (and called *flat*) are used to compute the binary reachability relation of BPP nets.

DEFINITION 4.8. A labelled transition system $LTS = (\mathsf{S}, \Sigma, \rightarrow)$ equipped with a non empty set X of initial states is *ρ-flat* if $Traces(LTS, X)$ is ρ-flat, where $\rho \in SLRE(\Sigma)$.

Note that a set of traces is necessarily closed under left factor. Thus, we may have $Traces(LTS, X) \subseteq \mathrm{LF}(\llbracket\rho\rrbracket)$ for some SLRE ρ, whereas $Traces(LTS, X)$ is not ρ-flat. However, since SLRE languages are closed under left factor, there would exist another SLRE ρ' such that $\llbracket\rho'\rrbracket = \mathrm{LF}(\llbracket\rho\rrbracket)$ and hence $Traces(LTS, X)$ would be ρ'-flat.

A ρ-flat LTS has not necessarily a SLRE traces language. Let us consider, for instance, a labelled transition system such that $Traces(LTS, s_0) = \{l_1^n l_2^m \mid m \leq n\}$. Because $\{l_1^n l_2^m \mid m \leq n\}$ is included in $l_1^* l_2^*$, the labelled transition system LTS is $(l_1^* l_2^*)$-flat but $Traces(LTS, s_0)$ is not regular, hence it is not equal to any SLRE. Nevertheless, we may use flatness of LTS to compute $post^*$ by first iterating l_1 and then iterating l_2. This intuitive idea leads to the following algorithm.

ALGORITHM 2 $\texttt{SymbolicSet}(\mathcal{A}_{LTS}, a, \nabla, \rho)$

Input: an abstraction $\mathcal{A}_{LTS} = (\mathsf{A}, post_{\mathcal{A}}, \gamma, \sqcup, \Sigma)$, an initial abstract state $a \in \mathsf{A}$, an acceleration ∇ for $\mathcal{A}_{LTS}$ and a $\rho \in SLRE(\Sigma)$, $\rho \neq \emptyset$.
Output: an abstract state.

if $\rho = \varepsilon$ **then**
  return a
else if $\rho = \rho_1 + \rho_2$ **then**
  return $\texttt{SymbolicSet}(\mathcal{A}_{LTS}, a, \nabla, \rho_1) \sqcup \texttt{SymbolicSet}(\mathcal{A}_{LTS}, a, \nabla, \rho_2)$
else if $\rho = l \cdot \rho'$ with $l \in \Sigma$ **then**
  return $\texttt{SymbolicSet}(\mathcal{A}_{LTS}, post_{\mathcal{A}}(a, l), \nabla, \rho')$
else if $\rho = x^* \cdot \rho'$ with $x \in \Sigma^*$ **then**
  return $\texttt{SymbolicSet}(\mathcal{A}_{LTS}, \nabla(a, x), \nabla, \rho')$

Remark. According to its recursive definition, the `SymbolicSet` algorithm always terminates.

As stated by the following proposition, when ∇ is sound and complete, the `SymbolicSet` algorithm computes exactly the set of states reachable by an execution sequence in $\llbracket\rho\rrbracket$.

PROPOSITION 4.4. *Let $\mathcal{A}_{LTS} = (\mathsf{A}, post_{\mathcal{A}}, \gamma, \sqcup, \Sigma)$ be an abstraction and ∇ be an acceleration for $\mathcal{A}_{LTS}$. For every abstract state $a \in \mathsf{A}$ and for every $\rho \in SLRE(\Sigma)$, $\rho \neq \emptyset$, the two following assertions hold:*

1. if ∇ is a sound acceleration for $\mathcal{A}_{LTS}$ then we have:

$$\gamma(\texttt{SymbolicSet}(\mathcal{A}_{LTS}, a, \nabla, \rho)) \subseteq post^*(\gamma(a))$$

2. if ∇ is a complete acceleration for $\mathcal{A}_{LTS}$ then we have:

$$\bigcup_{\sigma \in [\![\rho]\!]} post(\gamma(a), \sigma) \subseteq \gamma(\texttt{SymbolicSet}(\mathcal{A}_{LTS}, a, \nabla, \rho))$$

Proof. We prove both assertions by induction on the size of the SLRE ρ. Most cases are routine and are omitted. Consider the last case when $\rho = x^* \cdot \rho'$ with $x \in \Sigma^+$ and suppose that both assertions hold for ρ' and for $x \cdot \rho'$. From the definition of the `SymbolicSet` algorithm, we have:

$$\gamma(\texttt{SymbolicSet}(\mathcal{A}_{LTS}, a, \nabla, \rho)) = \texttt{SymbolicSet}(\mathcal{A}_{LTS}, \nabla(a, x), \nabla, \rho'))$$

Let us first prove that the first assertion holds for ρ. Assume that ∇ is a sound acceleration for $\mathcal{A}_{LTS}$. We get that $\gamma(\texttt{SymbolicSet}(\mathcal{A}_{LTS}, a, \nabla, \rho)) \subseteq post^*(\gamma(\nabla(a, x)))$ because the first assertion holds for ρ'. Since ∇ is sound, we have $\gamma(\nabla(a, x)) \subseteq post^*(\gamma(a))$. Hence we get:

$$\gamma(\texttt{SymbolicSet}(\mathcal{A}_{LTS}, a, \nabla, \rho)) \subseteq post^*(\gamma(a))$$

Let us now prove that the second assertion holds for ρ. Assume that ∇ is a complete acceleration for $\mathcal{A}_{LTS}$. Since $\gamma(\nabla(a, x)) \supseteq \bigcup_{i \in \mathbb{N}} post(\gamma(a), x^i)$, we get that for every $\sigma \in \Sigma^*$,

$$\bigcup_{i \in \mathbb{N}} post(\gamma(a), x^i \cdot \sigma) \subseteq post(\gamma(\nabla(a, x), \sigma))$$

As $[\![\rho]\!] = \{x^i \mid i \in \mathbb{N}\} \cdot [\![\rho']\!]$, we get that:

$$\bigcup_{\sigma \in [\![\rho]\!]} post(\gamma(a), \sigma) \quad = \bigcup_{\substack{\sigma \in [\![\rho']\!] \\ i \in \mathbb{N}}} post(\gamma(a), x^i \cdot \sigma) \quad \subseteq \bigcup_{\sigma \in [\![\rho']\!]} post(\gamma(\nabla(a, x), \sigma))$$

Finally, because the second assertion holds for ρ', we obtain that:

$$\begin{aligned} \bigcup_{\sigma \in [\![\rho]\!]} post(\gamma(a), \sigma) \ &\subseteq \ \gamma(\texttt{SymbolicSet}(\mathcal{A}_{LTS}, \nabla(a, x), \nabla, \rho')) \\ &\subseteq \ \gamma(\texttt{SymbolicSet}(\mathcal{A}_{LTS}, a, \nabla, \rho)) \end{aligned}$$

■

In the particular case of flat labelled transitions systems, the `SymbolicSet` algorithm gives an upper (resp. exact) finite representation of $post^*$, provided that the acceleration is complete (resp. sound and complete).

THEOREM 4.1. *Let $\mathcal{A}_{LTS} = (\mathsf{A}, post_{\mathcal{A}}, \gamma, \sqcup, \Sigma)$ be an abstraction and ∇ be a complete acceleration for $\mathcal{A}_{LTS}$. For every initial abstract state $a_0 \in \mathsf{A}$, if $(LTS, \gamma(a_0))$ is ρ-flat then we have:*

$$post^*(\gamma(a_0)) \subseteq \gamma(\texttt{SymbolicSet}(\mathcal{A}_{LTS}, a_0, \nabla, \rho))$$

If moreover ∇ is a sound acceleration then we have:

$$post^*(\gamma(a_0)) = \gamma(\texttt{SymbolicSet}(\mathcal{A}_{LTS}, a, \nabla, \rho))$$

Proof. Since $(LTS, \gamma(a_0))$ is ρ-flat, we have that $Traces(LTS, \gamma(a_0)) \subseteq [\![\rho]\!]$. Hence we get that $post^*(\gamma(a_0)) = \bigcup_{\sigma \in [\![\rho]\!]} post(\gamma(a_0), \sigma)$. We conclude with Proposition 4.4. ■

Another application of the `SymbolicSet` algorithm is, given any LTS and any flat formula φ (i.e. a formula representing a SLRE language of execution sequences), to compute the set of states of LTS reached by an execution sequence satisfying φ.

5. ACCELERATION OF FIFO AUTOMATA USING LINEAR REGULAR EXPRESSIONS

In this section we will consider SLREs as the basis of symbolic representations for FIFO automata and will present a new acceleration scheme. We will first consider accelerations for FIFO automata that works on a single channel, a class of automata which can represent the behavior of ring networks [32]. For this case we show that SLREs are sufficient and necessary to represent the result of accelerations. We then consider the general case of FIFO automata acting on multiple channels.

5.1. Acceleration for one channel

The generic acceleration for FIFO automata, from Section 4.4, suggests that we should characterize $\mathcal{SLRE}$-representable loops. To that end, consider that a FIFO automaton moves from (q, x) to (q, y) on an execution sequence σ. If it was possible for σ to be repeated infinitely often, then the only messages being removed from, or being added to the channel, come from σ. Clearly, the order in which messages are received, or sent, is preserved.

Notation. Given a state (q, w) and a loop σ on q, we say that σ is *infinitely iterable* from (q, w) if for every $n \in \mathbb{N}$ we have $(q, w) \xrightarrow{\sigma^n} (q, w_n)$ for some w_n.

Notation. We define $in : \Sigma^* \to M^*$ and $out : \Sigma^* \to M^*$ as follows:

$$in(\sigma) = \begin{cases} \varepsilon & \text{if } \sigma = \varepsilon \\ m \cdot in(\sigma') & \text{if } \sigma = (?m)\sigma' \\ in(\sigma') & \text{if } \sigma = (!m)\sigma' \end{cases} \qquad out(\sigma) = \begin{cases} \varepsilon & \text{if } \sigma = \varepsilon \\ out(\sigma') & \text{if } \sigma = (?m)\sigma' \\ m \cdot out(\sigma') & \text{if } \sigma = (!m)\sigma' \end{cases}$$

Assume σ is infinitely iterable from (q, w). Furthermore, assume without loss of generality that $in(\sigma)$ and $out(\sigma)$ are both not ε. Then, after the i^{th} iteration of σ, we will be left with $(in(\sigma)^i)^{-1}(w \cdot out(\sigma)^i)$ in the channel. In the limit what has been taken out is precisely what has been put in. To relate the two we need the following lemma on words.

LEMMA 5.1. *For every $w, x, y \in M^*$, the two following assertions are equivalent:*

1. $x^\omega = wy^\omega$

2. there exists three words $x', x'', z \in M^*$ such that we have $x = x'x''$, $w \in x^*x'$ and $(x''x'), y \in z^*$.

Proof. An application of the defect theorem (see [33]). ■

Now we are ready to relate the effect of an arbitrary iteration of σ:

PROPOSITION 5.1 ([17]). *Let $F = (Q, M, \delta)$ be a FIFO automaton with one channel. Given a state (q, w) and a loop σ on q, if σ is infinitely iterable from (q, w) then there exists $z_\sigma \in M^*$ such that $\sigma^*(\{w\}) = w{z_\sigma}^*$.*

Proof. Assume that σ is infinitely iterable from (q, w). Then we have $x^\omega = wy^\omega$ where $x = in(\sigma)$ and $y = out(\sigma)$. From Lemma 5.1, we obtain that there exists $x', x'', z \in M^*$ and $i, j, k \in \mathbb{N}$ such that $x = x'x''$, $w = x^k x'$, $x''x' = z^i$ and $y = z^j$. Notice that $i \leq j$ because $|x| \leq |y|$ (otherwise $|in(\sigma)| > |out(\sigma)|$ and hence σ would not be infinitely iterable from (q, w)). Hence we may fix $z_\sigma = z^{j-i}$ and we get that $y = x''x'z_\sigma$. Now, for any $n \in \mathbb{N}^*$, we have $(q, w) \xrightarrow{\sigma^n} (q, (x^n)^{-1}(wy^n))$ and $(x^n)^{-1}(wy^n)$ may be written as:

$$\begin{array}{rcll}
(x^n)^{-1}(wy^n) & = & (x^n)^{-1}(x^k x' y^n) & (\text{as } w = x^k x')) \\
& = & (x'(x''x')^{n-1}x'')^{-1}(x'(x''x')^k y^n) & (\text{as } x = x'x'') \\
& = & ((x''x')^{n-1}x'')^{-1}((x''x')^{k+n}{z_\sigma}^n) & (\text{as } y = x''x'z_\sigma) \\
& = & x'(x''x')^k {z_\sigma}^n & \\
& = & x^k x' {z_\sigma}^n & (\text{as } x = x'x'') \\
& = & w{z_\sigma}^n & (\text{as } w = x^k x')
\end{array}$$

■

Let σ denote a loop of a FIFO automaton with one channel. Starting from a single channel content $w \in M^*$, we get from the previous proposition that $\sigma^*(\{w\})$ is either a finite set (if σ is not infinitely iterable) or a SLRE expressible infinite set of channel contents. The following theorem, which was used, but not proved, in [17], expresses an even stronger property: starting from a SLRE expressible set of channel contents $[\![\chi]\!]$, $\sigma^*([\![\chi]\!])$ is still SLRE expressible.

THEOREM 5.1. *Every loop of a FIFO automaton with one channel is $\mathcal{SLRE}$-representable.*

Proof. We have to show that for every loop σ and for every finite set χ of SLREs, $\sigma^*([\![\chi]\!]) = [\![\chi']\!]$ for some finite set χ' of SLREs. Since σ^* distributes over union, it suffices actually to prove that for every LRE ρ, $\sigma^*([\![\rho]\!]) = [\![\rho']\!]$ for some SLRE ρ'.

If $in(\sigma) = \varepsilon$ then for every LRE ρ, we have $\sigma^*([\![\rho]\!]) = [\![\rho \cdot out(\sigma)^*]\!]$. Hence, we assume in the following of the proof that $in(\sigma) \neq \varepsilon$.

For every LRE ρ, we write n_ρ for the number of '$*$' in ρ. We introduce the *measure* Δ on LREs as follows:

$$\Delta(\rho) = \begin{cases} 2 * n_\rho - 1 & \text{if } \rho \text{ is of the form } y^* \cdot \rho' \\ 2 * n_\rho & \text{otherwise} \end{cases}$$

We show by induction on $\Delta(\rho)$ that for every LRE ρ, for every loop σ such that $in(\sigma) \neq \varepsilon$ and for every $\alpha \in \mathbb{N}^*$:

$$\begin{array}{rcll}
\sigma^*([\![\rho]\!]) & = & [\![\rho']\!] \quad \text{for some SLRE } \rho' & (\mathbf{Claim_1}) \\
\sigma^*([\![\rho \cdot (out(\sigma)^\alpha)^*]\!]) & = & [\![\rho'']\!] \quad \text{for some SLRE } \rho'' & (\mathbf{Claim_2})
\end{array}$$

Notice that when $out(\sigma) = \varepsilon$, $\mathbf{Claim_2}$ is equivalent to $\mathbf{Claim_1}$. Hence, we will always assume that $out(\sigma) \neq \varepsilon$ when proving $\mathbf{Claim_2}$ in the following induction.

$\boxed{\text{basis}}$ Let ρ be a LRE such that $\Delta(\rho) = 0$. We have $\rho = x$ for some $x \in \Sigma^*$. To prove $\mathbf{Claim_1}$, notice that $\sigma^*(x)$ is:

- either a finite set (and hence a SLRE) if σ is not infinitely iterable,
- or the LRE $xz_\sigma{}^*$ (according to Proposition 5.1) if σ is infinitely iterable.

Let us now prove $\mathbf{Claim_2}$ and assume that $out(\sigma) \neq \varepsilon$. There are two cases:

1. if $in(\sigma)^\omega \neq x \cdot out(\sigma)^\omega$ then we have $\sigma^k([\![x \cdot (out(\sigma)^\alpha)^*]\!]) = \emptyset$ for some k and hence $\sigma^*([\![x \cdot (out(\sigma)^\alpha)^*]\!]) = \bigcup_{i=0}^{k-1} \sigma^i([\![x \cdot (out(\sigma)^\alpha)^*]\!]) = [\![\rho']\!]$ where $\rho' = \sum_{i=0}^{k-1} \sigma^i(x \cdot (out(\sigma)^\alpha)^*)$ is a SLRE.
2. if $in(\sigma)^\omega = x \cdot out(\sigma)^\omega$ then we have $x \cdot out(\sigma)^{h\,|in(\sigma)|} = in(\sigma)^{h\,|out(\sigma)|} \cdot x$ for every $h \in \mathbb{N}$. Hence, there exists $x' \in \Sigma^*$ and $k \in \mathbb{N}$ such that $|x'| < |out(\sigma)|$ and $x = x' \cdot out(\sigma)^k$. We get:

$$\sigma^{\alpha\,|out(\sigma)|}([\![x \cdot (out(\sigma)^\alpha)^*]\!]) = x' \cdot out(\sigma)^{k+\alpha(\beta+|out(\sigma)|-|in(\sigma)|)} \cdot (out(\sigma)^\alpha)^*$$

where $\beta \in \mathbb{N}$ denotes how many times the word $out(\sigma)^\alpha$ was needed (from the part $out(\sigma)^\alpha)^*$) during the execution of $\sigma^{\alpha\,|out(\sigma)|}$. Notice that β satisfies $k+\alpha(\beta+|out(\sigma)|-|in(\sigma)|) \geq 0$. Let $y = x' \cdot out(\sigma)^{k+\alpha(\beta+|out(\sigma)|-|in(\sigma)|)}$. We have:

$$\begin{aligned}\sigma^*([\![x \cdot (out(\sigma)^\alpha)^*]\!]) &= \sigma^*([\![y \cdot (out(\sigma)^\alpha)^*]\!]) \quad \bigcup_{i=0}^{\alpha\,|out(\sigma)|-1} \quad \sigma^i([\![x \cdot (out(\sigma)^\alpha)^*]\!]) \\ &= \sigma^*([\![y \cdot (out(\sigma)^\alpha)^*]\!]) \quad \bigcup \quad [\![\rho']\!]\end{aligned}$$

where $\rho' = \sum_{i=0}^{\alpha\,|out(\sigma)|-1} \sigma^i(x \cdot (out(\sigma)^\alpha)^*)$ is a SLRE. There are again two cases:

(i) If $\beta + |out(\sigma)| - |in(\sigma)| \geq 0$, then we get that $y \in x \cdot (out(\sigma)^\alpha)^*$ and hence $[\![y \cdot (out(\sigma)^\alpha)^*]\!] \subseteq [\![x \cdot (out(\sigma)^\alpha)^*]\!]$. Consequently, $\sigma^{\alpha\,|out(\sigma)|}([\![x \cdot (out(\sigma)^\alpha)^*]\!]) \subseteq x \cdot (out(\sigma)^\alpha)^*$ and we get that $\sigma^*([\![x \cdot (out(\sigma)^\alpha)^*]\!]) = [\![\rho']\!]$.

(ii) Otherwise, since ρ' is a SLRE, we may replace x by y and repeat the same method to show that $\sigma^*([\![y \cdot (out(\sigma)^\alpha)^*]\!]) = [\![\rho'']\!]$ for some SLRE ρ''. This process will terminate since $y < x$.

$\boxed{\text{induction step}}$ Let $K \in \mathbb{N}$ and assume that $\mathbf{Claim_1}$ and $\mathbf{Claim_2}$ are true for any LRE ρ such that $\Delta(\rho) \leq K$. Let ρ be a LRE such that $\Delta(\rho) = K + 1$. There are two cases corresponding to the parity of K.

Assume that $K + 1$ is even. Then $\rho = x \cdot \rho'$ for some $x \in \Sigma^*$ and for some LRE ρ' such that $\Delta(\rho') = K$. There are again two cases:

1. if $x \notin \mathrm{LF}(in(\sigma)^*)$ then there exists $k \in \mathbb{N}$ such that $x \not\leq in(\sigma)^k$ and $|x| \leq |in(\sigma)^k|$. Hence, for every LRE π we have $\sigma^k([\![x \cdot \pi]\!]) = \emptyset$ and we get that:

$$\sigma^*([\![x \cdot \pi]\!]) = \bigcup_{i=0}^{k-1} \sigma^i([\![x \cdot \pi]\!]) = [\![\sum_{i=0}^{k-1} \sigma^i(x \cdot \pi)]\!]$$

This shows both $\mathbf{Claim_1}$ and $\mathbf{Claim_2}$ for ρ.

2. if $x \in \mathrm{LF}(in(\sigma)^*)$ then there exists $k \in \mathbb{N}$ and $x' \leq in(\sigma)$ such that $x = in(\sigma)^k \cdot x'$. For every LRE π, we have:

$$\begin{aligned}\sigma^*([\![x \cdot \pi]\!]) &= \sigma^*(\sigma([\![x' \cdot \pi \cdot out(\sigma)^k]\!])) \bigcup_{i=0}^{k} in(\sigma)^{k-i} \cdot x' \cdot \pi \cdot out(\sigma)^i \\ &= \sigma^*(\sigma([\![x' \cdot \pi \cdot out(\sigma)^k]\!])) \bigcup [\![\sum_{i=0}^{k} in(\sigma)^{k-i} \cdot x' \cdot \pi \cdot out(\sigma)^i]\!]\end{aligned}$$

Note that $\sum_{i=0}^{k} in(\sigma)^{k-i} \cdot x' \cdot \pi \cdot out(\sigma)^i$ is a SLRE. The loop σ may be decomposed as $\sigma = \sigma_1\sigma_2$ where $in(\sigma_1) = x'$. Notice that $(\sigma_2\sigma_1)$ is also a loop satisfying $in(\sigma_2\sigma_1) \neq \varepsilon$. We get:

$$\begin{aligned}\sigma^*(\sigma([\![x' \cdot \pi \cdot out(\sigma)^k]\!])) &= \sigma_2((\sigma_2\sigma_1)^*(\sigma_1([\![x' \cdot \pi \cdot out(\sigma)^k]\!]))) \\ &= \sigma_2((\sigma_2\sigma_1)^*([\![\pi \cdot out(\sigma)^k \cdot out(\sigma_1)]\!])))\end{aligned}$$

Since $\Delta(\rho' \cdot out(\sigma)^k \cdot out(\sigma_1)) = \Delta(\rho') = K$, we get from the induction hypothesis ($\mathbf{Claim_1}$) that $(\sigma_2\sigma_1)^*([\![\rho' \cdot out(\sigma)^k \cdot out(\sigma_1)]\!]) = [\![\rho'']\!]$ for some SLRE ρ''. Hence, we obtain $\sigma^*(\sigma([\![x' \cdot \rho' \cdot out(\sigma)^k]\!])) = [\![\rho'']\!]$ and we come to $\sigma^*([\![\rho]\!]) = \sigma^*([\![x \cdot \rho']\!]) = [\![\rho'' + \sum_{i=0}^{k} in(\sigma)^{k-i} \cdot x' \cdot \rho' \cdot out(\sigma)^i]\!]$. This proves $\mathbf{Claim_1}$ for ρ. To prove $\mathbf{Claim_2}$ for ρ, assume that $out(\sigma) \neq \varepsilon$ and notice that:

$$\begin{aligned}&\sigma_2((\sigma_2\sigma_1)^*([\![\rho' \cdot (out(\sigma)^\alpha)^* \cdot out(\sigma)^k \cdot out(\sigma_1)]\!]))) \\ &\quad= \sigma_2((\sigma_2\sigma_1)^*([\![\rho' \cdot out(\sigma)^k \cdot (out(\sigma)^\alpha)^* \cdot out(\sigma_1)]\!]))) \\ &\quad= \sigma_2((\sigma_2\sigma_1)^*([\![\rho' \cdot out(\sigma)^k \cdot out(\sigma_1) \cdot (out(\sigma_2\sigma_1)^\alpha)^*]\!])))\end{aligned}$$

Since $\Delta(\rho' \cdot out(\sigma)^k \cdot out(\sigma_1)) = \Delta(\rho') = K$, we get from the induction hypothesis ($\mathbf{Claim_2}$) that $(\sigma_2\sigma_1)^*([\![\rho' \cdot out(\sigma)^k \cdot out(\sigma_1) \cdot (out(\sigma_2\sigma_1)^\alpha)^*]\!]) = [\![\rho'']\!]$ for some SLRE ρ''. Hence, we obtain $\sigma^*(\sigma([\![x' \cdot \rho' \cdot (out(\sigma)^\alpha)^* \cdot out(\sigma)^k]\!])) = [\![\rho'']\!]$ and we come to $\sigma^*([\![\rho]\!]) = \sigma^*([\![x \cdot \rho' \cdot (out(\sigma)^\alpha)^*]\!]) = [\![\rho'' + \sum_{i=0}^{k} in(\sigma)^{k-i} \cdot x' \cdot \rho' \cdot (out(\sigma)^\alpha)^* \cdot out(\sigma)^i]\!]$. This proves $\mathbf{Claim_2}$ for ρ.

Assume that $K + 1$ is odd. Then $\rho = y^* \cdot \rho'$ for some $y \in \Sigma^+$ and for some LRE ρ' such that $\Delta(\rho') \leq K$. There are again two cases:

1. if $y^\omega \neq in(\sigma)^\omega$ then we have $y^k \notin \mathrm{LF}(in(\sigma)^*)$ for some $k \in \mathbb{N}$. Hence there exists $l \in \mathbb{N}$ such that $y^k \not\leq in(\sigma)^l$ and $|y^k| \leq |in(\sigma)^l|$. Thus, for every LRE π we obtain $\sigma^l([\![y^k \cdot y^* \cdot \pi]\!]) = \emptyset$ and we get that:

$$\begin{aligned}\sigma^*([\![y^* \cdot \pi]\!]) &= \sigma^*([\![y^k \cdot y^* \cdot \pi]\!]) \quad \bigcup_{j=0}^{k-1} \sigma^*([\![y^j \cdot \pi]\!]) \\ &= \bigcup_{i=0}^{l-1} \sigma^i([\![y^k \cdot y^* \cdot \pi]\!]) \quad \bigcup_{j=0}^{k-1} \sigma^*([\![y^j \cdot \pi]\!])\end{aligned}$$

Now, for every $0 \leq j \leq k-1$, we have $\Delta(y^j \cdot \rho') \leq K$. Hence we get from the induction hypothesis that:

$(\mathbf{Claim_1})$ for every $0 \leq j \leq k-1$, $\sigma^*([\![y^j \cdot \rho']\!]) = [\![\rho_j]\!]$ for some SLRE ρ_j. Hence, we come to:

$$\sigma^*([\![\rho]\!]) = \left[\!\!\left[\sum_{i=0}^{l-1} \sigma^i(y^k \cdot y^* \cdot \rho' \quad + \quad \sum_{j=0}^{k-1} \rho_j \right]\!\!\right]$$

and this proves $(\mathbf{Claim_1})$ for ρ.

$(\mathbf{Claim_2})$ assuming that $out(\sigma) \neq \varepsilon$, we have that for every $0 \leq j \leq k-1$, $\sigma^*([\![y^j \cdot \rho' \cdot (out(\sigma)^\alpha)^*]\!]) = [\![\rho_j]\!]$ for some SLRE ρ_j. Hence, we come to:

$$\sigma^*([\![\rho \cdot (out(\sigma)^\alpha)^*]\!]) = \left[\!\!\left[\sum_{i=0}^{l-1} \sigma^i(y^k \cdot y^* \cdot \rho' \cdot (out(\sigma)^\alpha)^*) \quad + \quad \sum_{j=0}^{k-1} \rho_j \right]\!\!\right]$$

and this proves $(\mathbf{Claim_2})$ for ρ.

2. if $y^\omega = in(\sigma)^\omega$ then we have $y^{|in(\sigma)|} = in(\sigma)^{|y|}$. We deduce that for every $i, j, k, l \in \mathbb{N}$ and for every LRE π we have:

$$\begin{array}{l} \text{if} \;\; k \geq l \;\; \text{then} \;\; \sigma^{l|y|+i}([\![y^{k|in(\sigma)|+j} \cdot \pi]\!])) = \sigma^i([\![y^{(k-l)|in(\sigma)|+j} \cdot \pi \cdot out(\sigma)^{l|y|}]\!]) \\ \text{if} \;\; k \leq l \;\; \text{then} \;\; \sigma^{l|y|+i}([\![y^{k|in(\sigma)|+j} \cdot \pi]\!])) = \sigma^{(l-k)|y|+i}([\![y^j \cdot \pi \cdot out(\sigma)^{k|y|}]\!]) \end{array}$$

Hence we obtain:

$$\begin{aligned} \sigma^*([\![y^* \cdot \pi]\!]) &= \bigcup_{i,j \geq 0} \sigma^i([\![y^j \cdot \pi]\!]) \\ &= \bigcup_{i=0}^{|y|-1} \bigcup_{j=0}^{|in(\sigma)|-1} \bigcup_{k,l \geq 0} \sigma^{l|y|+i}([\![y^{k|in(\sigma)|+j} \cdot \pi]\!])) \end{aligned}$$

Now, for every $i, j \in \mathbb{N}$, we have:

$$\begin{aligned} &\bigcup_{k,l \geq 0} \sigma^{l|y|+i}([\![y^{k|in(\sigma)|+j} \cdot \pi]\!]) \\ &= \bigcup_{k \geq l \geq 0} \sigma^{l|y|+i}([\![y^{k|in(\sigma)|+j} \cdot \pi]\!]) \bigcup_{l \geq k \geq 0} \sigma^{l|y|+i}([\![y^{k|in(\sigma)|+j} \cdot \pi]\!]) \\ &= \bigcup_{k \geq l \geq 0} \sigma^i([\![y^{(k-l)|in(\sigma)|+j} \cdot \pi \cdot out(\sigma)^{l|y|}]\!]) \bigcup_{l \geq k \geq 0} \sigma^{(l-k)|y|+i}([\![y^j \cdot \pi \cdot out(\sigma)^{k|y|}]\!]) \\ &= \sigma^i([\![(y^{|in(\sigma)|})^* y^j \cdot \pi \cdot (out(\sigma)^{|y|})^*]\!]) \bigcup (\sigma^{|y|})^*(\sigma^i([\![y^j \cdot \pi \cdot (out(\sigma)^{|y|})^*]\!])) \end{aligned}$$

Moreover, since we have:

$$\begin{aligned} \bigcup_{j=0}^{|in(\sigma)|-1} \sigma^i([\![(y^{|in(\sigma)|})^* y^j \cdot \pi \cdot (out(\sigma)^{|y|})^*]\!]) \;&=\; \sigma^i([\![(y^* \cdot \pi \cdot (out(\sigma)^{|y|})^*]\!]) \\ \bigcup_{i=0}^{|y|-1} (\sigma^{|y|})^*(\sigma^i([\![y^j \cdot \pi \cdot (out(\sigma)^{|y|})^*]\!])) \;&=\; \sigma^*([\![y^j \cdot \pi \cdot (out(\sigma)^{|y|})^*]\!]) \end{aligned}$$

we finally get:

$$\sigma^*([\![y^* \cdot \pi]\!]) = \bigcup_{i=0}^{|y|-1} \sigma^i([\![(y^* \cdot \pi \cdot (out(\sigma)^{|y|})^*]\!]) \bigcup_{j=0}^{|in(\sigma)|-1} \sigma^*([\![y^j \cdot \pi \cdot (out(\sigma)^{|y|})^*]\!])$$

Now, for every $0 \leq j \leq |in(\sigma)| - 1$, we have $\Delta(y^j \cdot \rho') \leq K$. Hence we get from the induction hypothesis ($\mathbf{Claim_2}$) that $\sigma^*([\![y^j \cdot \rho' \cdot (out(\sigma)^{|y|})^*]\!]) = [\![\rho_j]\!]$ for some SLRE ρ_j. Hence, we come to:

$$\sigma^*([\![\rho]\!]) = \left[\!\!\left[\sum_{i=0}^{|y|-1} \sigma^i(y^* \cdot \rho' \cdot (out(\sigma)^{|y|})^*) \quad + \quad \sum_{j=0}^{|in(\sigma)|-1} \rho_j \right]\!\!\right]$$

This proves $\mathbf{Claim_1}$ for ρ. To prove $\mathbf{Claim_2}$ for ρ, assume that $out(\sigma) \neq \varepsilon$ and notice that for every $0 \leq j \leq |in(\sigma)| - 1$, we have:

$$[\![y^j \cdot \rho' \cdot (out(\sigma)^{\alpha})^* \cdot (out(\sigma)^{|y|})^*]\!] = \bigcup_{h=0}^{\alpha-1} [\![y^j \cdot \rho' \cdot (out(\sigma)^{|y|})^h \cdot (out(\sigma)^{\alpha})^*]\!]$$

Now, for every $0 \leq j \leq |in(\sigma)| - 1$ and for every $0 \leq h \leq \alpha - 1$, we have $\Delta(y^j \cdot \rho' \cdot (out(\sigma)^{|y|})^h) \leq K$. Hence we get from the induction hypothesis ($\mathbf{Claim_2}$) that $\sigma^*([\![y^j \cdot \rho' \cdot (out(\sigma)^{|y|})^h \cdot (out(\sigma)^{\alpha})^*]\!]) = [\![\rho_j^h]\!]$ for some SLRE ρ_j^h. Hence, we come to:

$$\sigma^*([\![\rho]\!]) = \left[\!\!\left[\sum_{i=0}^{|y|-1} \sigma^i(y^* \cdot \rho' \cdot (out(\sigma)^{\alpha})^* \cdot (out(\sigma)^{|y|})^*) \quad + \quad \sum_{j=0}^{|in(\sigma)|-1} \sum_{h=0}^{\alpha-1} \rho_j^h \right]\!\!\right]$$

This proves $\mathbf{Claim_2}$ for ρ.

■

Note that from [5], we know that every loop of a FIFO automaton is $\mathcal{CQDD}$-representable (see Theorem 6.2). However, their proof extensively uses Pressburger formulas (even for one channel) and hence our theorem cannot be immediately deduced from [5]. A straightforward consequence of Theorem 5.1 is that for any FIFO automaton with one channel F, $\nabla_{\mathcal{SLRE}}$ is a sound and complete acceleration for $\mathcal{A}_{LTS(F)}^{\mathcal{SLRE}}$.

5.2. Acceleration for multiple channels

To characterize $\mathcal{SLRE}$-representable loops in the case of multiple channels, we use a condition on loops defined in [4]. We say that a loop σ is *non counting* if one of the following conditions is satisfied:

$i)$ $|M| > 1$ and every send transition in σ operates on the same channel,
$ii)$ $|M| = 1$ and there is at most one channel which is growing with σ.

Since $\mathcal{SLRE}$ symbolic channel contents cannot express relationships between channels, we have to deal with each channel independently. To that end, we need the following definitions. For every loop σ and for every channel $c \in C$, the *projection of σ on c*, denoted by $\sigma_{|_c}$, is obtained from σ by removing every operation on c' with $c' \neq c$. Recall that from Theorem 5.1, we know that $(\sigma_{|_c})^*(\rho)$ is a SLRE for any SLRE ρ. We define $\overline{\sigma^*} : \mathcal{P}_f(SLRE(M)^C) \to \mathcal{P}_f(SLRE(M)^C)$ by:

$$\overline{\sigma^*}(\chi) = \bigcup_{\mathsf{r} \in \chi} \prod_{c \in C} (\sigma_{|_c})^*(\mathsf{r}[c])$$

LEMMA 5.2. *For every loop σ and for every $\mathcal{SLRE}$ symbolic channel content $\chi \in \mathcal{P}_f(SLRE(M)^C)$, we have $\sigma^*([\![\chi]\!]) \subseteq [\![\overline{\sigma^*}(\chi)]\!]$.*

We are now ready to obtain a characterization of $\mathcal{SLRE}$-representable loops.

THEOREM 5.2. *A loop σ is $\mathcal{SLRE}$-representable iff it is non counting.*

Proof. We first prove the sufficient condition. Assume that σ is a loop satisfying condition $i)$ or condition $ii)$. Since σ^* distributes over union, we just have to prove that for every $\mathsf{r} \in SLRE(M)^C$, $\sigma^*(\mathsf{r}) = [\![\mathsf{r}']\!]$ for some $\mathsf{r}' \in SLRE(M)^C$. Let $\mathsf{r} \in SLRE(M)^C$.

We first show the following claim: there exists a channel c_0 and integers $h, k \in \mathbb{N}$ with $h > 0$ such that $(\sigma_{|_c})^{k+h}([\![\mathsf{r}[c]]\!]) = (\sigma_{|_c})^k([\![\mathsf{r}[c]]\!])$ for every $c \neq c_0$. There are two cases:

1. if $|M| > 1$ then σ satisfies condition $i)$ and hence we get that there exists a channel c_0 such that $out(\sigma_{|_c}) = \varepsilon$ for every $c \neq c_0$. Consider a channel $c \neq c_0$. For every $i \in \mathbb{N}$, we have $(\sigma_{|_c})^i([\![\mathsf{r}[c]]\!]) = ((in(\sigma_{|_c}))^i)^{-1}([\![\mathsf{r}[c]]\!])$, and since $[\![\mathsf{r}[c]]\!]$ is a regular language, we get that there is $h_c, k_c \in \mathbb{N}$ with $h_c > 0$ such that $(\sigma_{|_c})^{k_c+h_c}([\![\mathsf{r}[c]]\!]) = (\sigma_{|_c})^{k_c}([\![\mathsf{r}[c]]\!])$. Using:

$$\begin{aligned} k &= \max(\{k_c \mid c \in C, c \neq c_0\}) \\ h &= \operatorname{lcm}(\{h_c \mid c \in C, c \neq c_0\}) \end{aligned}$$

we come to $(\sigma_{|_c})^{k+h}([\![\mathsf{r}[c]]\!]) = (\sigma_{|_c})^k([\![\mathsf{r}[c]]\!])$ for every channel $c \neq c_0$.

2. if $|M| = 1$ then σ satisfies condition $ii)$ and hence we get that there exists a channel c_0 such that $out(\sigma_{|_c}) \leq in(\sigma_{|_c})$ for every $c \neq c_0$. Observe that since $|M| = 1$, the channels actually behave as counters here. For each channel c, we can split $\sigma_{|_c}$ into two parts $\sigma'_{|_c}$ and $\sigma''_{|_c}$ (with $\sigma_{|_c} = \sigma'_{|_c}\sigma''_{|_c}$) such that any suffix of $\sigma''_{|_c}\sigma'_{|_c}$ contains more receptions (aka decrements) than emissions (aka increments). Now it is readily seen that $\sigma''_{|_c}\sigma'_{|_c}$ is equivalent to the sequence $\sigma'''_{|_c} = (c?m)^d$ where $d = |out(\sigma_{|_c})| - |in(\sigma_{|_c})|$. Thus we are left with sequences $\sigma'''_{|_c}$ statisfying $out(\sigma'''_{|_c}) = \varepsilon$, for all $c \neq c_0$, and we can apply the same technique as in the first case.

So we have proved that there exists a channel c_0 and integers $h, k \in \mathbb{N}$ with $h > 0$ such that $(\sigma_{|_c})^{k+h}([\![\mathsf{r}[c]]\!]) = (\sigma_{|_c})^k([\![\mathsf{r}[c]]\!])$ for every $c \neq c_0$. Let us write $\sigma^*([\![\mathsf{r}]\!])$ as follows:

$$\sigma^*([\![\mathsf{r}]\!]) = \bigcup_{i=0}^{k-1} \sigma^i([\![\mathsf{r}]\!]) \quad \bigcup \quad \bigcup_{i=k}^{k+h-1} \bigcup_{j \in \mathbb{N}} \sigma^{i+jh}([\![\mathsf{r}]\!])$$

In order to conclude the proof, it suffices to prove that $\bigcup_{j \in \mathbb{N}} \sigma^{i+jh}([\![\mathsf{r}]\!])$ is an $\mathcal{SLRE}$ expressible set of channel contents for every $i \geq k$. So let us consider $i \geq k$. We have:

$$\begin{aligned} \bigcup_{j \in \mathbb{N}} \sigma^{i+jh}([\![\mathsf{r}]\!]) &= \bigcup_{j \in \mathbb{N}} \prod_{c \in C} (\sigma_{|c})^{i+jh}([\![\mathsf{r}[c]]\!]) \\ &= \left(\prod_{c \in C, c \neq c_0} (\sigma_{|c})^i([\![\mathsf{r}[c]]\!]) \right) \times \left(\bigcup_{j \in \mathbb{N}} (\sigma_{|c_0})^{i+jh}([\![\mathsf{r}[c_0]]\!]) \right) \\ &= \left(\prod_{c \in C, c \neq c_0} (\sigma_{|c})^i([\![\mathsf{r}[c]]\!]) \right) \times \left(((\sigma_{|c_0})^h)^* ((\sigma_{|c_0})^i([\![\mathsf{r}[c_0]]\!])) \right) \end{aligned}$$

Now, according to Theorem 5.1, we get that the loop $((\sigma_{|c_0})^h)$ is $\mathcal{SLRE}$-representable, which concludes the proof of the sufficient condition.

Let us now prove the necessary condition. Assume that σ is a loop such that condition $i)$ and condition $ii)$ are not satisfied. We may suppose without loss of generality that there are exactly two channels 1 and 2. We have to show that there exists $\chi \in \mathcal{P}_f(SLRE(M)^2)$ such that $\sigma^*(\llbracket\chi\rrbracket) \neq \llbracket\chi'\rrbracket$ for all $\chi' \in \mathcal{P}_f(SLRE(M)^2)$. There are two cases:

1. if $|M| > 1$ then we have $out(\sigma_{|_1}) \neq \varepsilon$ and $out(\sigma_{|_2}) \neq \varepsilon$ since condition $i)$ is not satisfied. Let m_1 (resp. m_2) be a message such that $m_1 \not\leq in(\sigma_{|_1})$ (resp. $m_2 \not\leq in(\sigma_{|_2})$. Note that m_1 and m_2 are guaranteed to exist because $|M| > 1$. We define the $\mathcal{SLRE}$ symbolic channel content $\chi = \{(in(\sigma_{|_1})^* \cdot m_1, in(\sigma_{|_2})^* \cdot m_2)\}$. We get that:

$$\sigma^*(\llbracket\chi\rrbracket) = \{(in(\sigma_{|_1})^n \cdot m_1 \cdot out(\sigma_{|_1})^k, in(\sigma_{|_2})^m \cdot m_2 \cdot out(\sigma_{|_2})^k) \mid n, m, k \in \mathbb{N}\}$$

which is not a $\mathcal{SLRE}$ expressible set of channel contents.

2. if $|M| = 1$ then we have $out(\sigma_{|_1}) > in(\sigma_{|_1})$ and $out(\sigma_{|_2}) > in(\sigma_{|_1})$ since condition $ii)$ is not satisfied. Let $\chi = \{(in(\sigma_{|_1}), in(\sigma_{|_2}))\}$. We get that:

$$\sigma^*(\llbracket\chi\rrbracket) = \{(in(\sigma_{|_1}) \cdot out(\sigma_{|_1})^k, in(\sigma_{|_2}) \cdot out(\sigma_{|_2})^k) \mid k \in \mathbb{N}\}$$

which is not a $\mathcal{SLRE}$ expressible set of channel contents.

■

A straightforward consequence of the theorem is that for any FIFO automaton F, $\nabla_{\mathcal{SLRE}}$ is a sound acceleration for $\mathcal{A}^{\mathcal{SLRE}}_{LTS(F)}$. We now show how a complete acceleration can be defined for $\mathcal{A}^{\mathcal{SLRE}}_{LTS(F)}$.

Notation. Given a control state q, an $\mathcal{SLRE}$ symbolic channel content χ and a loop σ on q, we say that σ is *infinitely iterable* from (q, χ) if $\sigma^n(\llbracket\chi\rrbracket) \neq \emptyset$ for every $n \in \mathbb{N}$.

We define the total function $\overline{\nabla}_{\mathcal{SLRE}} : (\mathcal{P}_f(SLRE(M)^C))^Q \times \Sigma^+ \to (\mathcal{P}_f(SLRE(M)^C))^Q$ by:

$$\overline{\nabla}_{\mathcal{SLRE}}(\mathsf{a}, \sigma) = \begin{cases} \mathsf{a}[q := \overline{\sigma}^*(\mathsf{a}[q])] & \text{if } \sigma \text{ is a loop on } q \text{ infinitely iterable from } (q, \mathsf{a}[q]) \\ \mathsf{a}[q := \bigsqcup_{i=0}^{k} \sigma^i(\mathsf{a}[q])] & \text{if } \sigma \text{ is a loop on } q \text{ such that } \sigma^k(\mathsf{a}[q]) = \emptyset \\ \mathsf{a} & \text{otherwise} \end{cases}$$

PROPOSITION 5.2. *For any FIFO automaton $F = (Q, C, M, \delta)$, the total function $\overline{\nabla}_{\mathcal{SLRE}}$ is a complete acceleration for $\mathcal{A}^{\mathcal{SLRE}}_{LTS(F)}$.*

Notice that a one-message FIFO automaton can be viewed as a Petri Net (or equivalently a Vector Addition System with States). We may indeed identify any nonnegative integer n with the word m^n. The following proposition shows that in this case the `SymbolicTree` algorithm, with acceleration $\overline{\nabla}_{\mathcal{SLRE}}$, terminates and actually computes a tight over-approximation of $post^*$.

Given a subset $X \subseteq (M^*)^C$, we write $\downarrow X$ for the *downward closure* of X defined by $\downarrow X = \{\mathsf{y} \in (M^*)^C \mid \exists\, \mathsf{x} \in X, \forall c \in C, \mathsf{x}'[c] \leq \mathsf{x}[c]\}$.

PROPOSITION 5.3. *For any FIFO automaton $F = (Q, C, \{m\}, \delta)$ and for every initial abstract state a_0,* `SymbolicTree`$(\mathcal{A}^{\mathcal{SLRE}}_{LTS(F)}, a_0, \overline{\nabla}_{\mathcal{SLRE}})$ *terminates and computes an abstract state a such that $post^*(\gamma(a_0)) \subseteq \gamma(a) \subseteq \downarrow post^*(\gamma(a_0))$.*

Notice that $\downarrow post^*(\gamma(a_0))$ is precisely what is behind the notion of *coverability set* in the Petri nets framework [29, 16]. The coverability set is actually a particular finite representation (on $\mathbb{N} \cup \{\omega\}$) for the downward closure of the reachability set. However, the `SymbolicTree` algorithm may compute a more precise over-approximation of $post^*$ than the coverability set. In other words, the `SymbolicTree`, with acceleration $\overline{\nabla}_{\mathcal{SLRE}}$, is a refinement of the coverability set constructed by Karp and Miller's algorithm. Consider for instance the example given below:

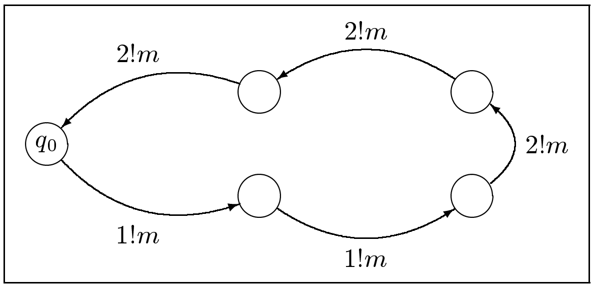

FIG. 2. A simple one-message FIFO automaton

Starting from a single initial state $(q_0, \varepsilon, \varepsilon)$, the reachable states in control state q_0 are $\{(q_0, (aa)^n, (aaa)^n) \mid n \in \mathbb{N}\}$, or equivalently $\{(q_0, 2n, 3n) \mid n \in \mathbb{N}\}$. The over-approximation computed by the `SymbolicTree`, with acceleration $\overline{\nabla}_{\mathcal{SLRE}}$, is readily seen to be $\{(q_0, 2n, 3p) \mid n, p \in \mathbb{N}\}$. However, the coverability set here is $\{(q_0, \omega, \omega)\}$ which yields a much less accurate over-approximation of the reachability set in q_0.

6. COMPARISON OF SLRE, QDD AND CQDD

We now show how our notion of abstraction and acceleration is applicable to current work on meta-transitions (QDDs and CQDDs).

6.1. QDDs as an abstraction

In this subsection we present a slightly different abstraction, namely QDDs due to Boigelot and Godfroid [4]. A QDD D for a FIFO automaton $F = (Q, C, M, \delta)$, where $C = \{c_1, c_2, \cdots c_n\}$ is a finite automaton operating on the alphabet $C \times M$ and satisfying the following condition: for every accepted word $w \in L(D)$, the projection of w on C is in $c_1{}^* c_2{}^* \cdots c_n{}^*$. Intuitively, a channel content $\mathsf{w} \in (M^*)^C$ is in the set of channel contents represented by D iff there exists $u_1 u_2 \cdots u_n \in L(D)$ such that for every $1 \leq i \leq n$, u_i may be written as $u_i = (c_i, m_1)(c_i, m_2) \cdots (c_i, m_k)$ with $\mathsf{w}[c_i] = m_1 m_2 \cdots m_k$. It is proved in [4] that sets of channel contents representable by QDDs are exactly those that are recognizable sets. Hence, QDDs correspond to the FIFO symbolic representation based on regular expressions. More formally, DEFINITION 6.1. Let $\mathcal{QDD} = (\mathcal{P}_f(REG(M)^C), \emptyset, ??, !!, [\![\cdot]\!], \cup)$, where $??$, $!!$ and $[\![\cdot]\!]$ are three total functions defined as follows:

$$\begin{array}{rcl} ??(c, m, \chi) & = & \{\mathsf{r}[c := m^{-1}(\mathsf{r}[c])] \mid \mathsf{r} \in \chi\} \\ !!(c, m, \chi) & = & \{\mathsf{r}[c := \mathsf{r}[c] \cdot m] \mid \mathsf{r} \in \chi\} \\ [\![\chi]\!] & = & \bigcup_{\mathsf{r} \in \chi} \prod_{c \in C} [\![\mathsf{r}[c]]\!] \end{array}$$

Note that $\mathcal{QDD}$ is clearly a FIFO symbolic representation and hence $\mathcal{A}^{\mathcal{QDD}}_{LTS(F)}$ is an abstraction of $LTS(F)$.

THEOREM 6.1 ([4]). *A loop σ is $\mathcal{QDD}$-representable iff it is non counting.*

Let us remark that a loop is $\mathcal{QDD}$-representable iff it is $\mathcal{SLRE}$-representable. A straightforward consequence of the theorem is that for any FIFO automaton F, $\nabla_{\mathcal{QDD}}$ is a sound acceleration for $\mathcal{A}^{\mathcal{QDD}}_{LTS(F)}$.

6.2. CQDDs as an abstraction

A generalization of QDDs, called Constrained Queue Decision Diagrams (CQDDs), was considered in [5]. We don't recall formally the original definition of CQDDs and refer the reader to [5] for details. Intuitively, in the original definition, a CQDD is a finite set of *accepting components*, where an accepting component is composed of a tuple of particular deterministic automata, each recognizing a "deterministic" LRE, and of linear constraints (given as a Presburger formula) on the number of times transitions in these automata are taken. However, it is readily seen that these linear constraints actually relate the number of times the '$*$' is used in the corresponding LREs. Thus, we may retranslate in our formalism the original definition of CQDDs.

DEFINITION 6.2. An *accepting component* is a pair (r, φ) such that:

1. $\mathsf{r} \in LRE(M)^C$, and,
2. for every $c \in C$, the LRE $\mathsf{r}[c]$ contains none of the following factors: $(au)^*a$, u^*v^* with $u, v \in M^*$ and $a \in M$, and,
3. φ is a Presburger arithmetic formula with a set of free variables $X = \{x_c^i \mid c \in C, 1 \leq i \leq n_c\}$ where n_c is the number of occurences of '$*$' in $\mathsf{r}[c]$.

A *Constrained Queue Decision Diagram (CQDD)* is a finite set of accepting components and we write $CQDD(C, M)$ for the set of CQDDs over C and M.

This definition is more simple than the original definition (stated in [5]) given in terms of deterministic restricted simple automata.

We order the occurences of '$*$' in a LRE ρ by reading ρ from left to right. To a LRE ρ with n occurences of '$*$', we associate a function $f_\rho : \mathbb{N}^n \to \Sigma^*$ where $f_\rho(k_1, k_2, \ldots, k_n)$ is the word obtained from ρ by replacing the i^{th} occurence of '$*$' by k_i for every $i, 1 \leq i \leq n$. The set of channel contents $[\![\chi]\!]$ associated with a CQDD χ is given by $[\![\chi]\!] = \cup_{(\mathsf{r},\varphi) \in \chi} [\![(\mathsf{r}, \varphi)]\!]$ where $[\![(\mathsf{r}, \varphi)]\!]$ is the set of channel contents $\mathsf{w} \in (M^*)^C$ such that for some valuation $v : X \to \mathbb{N}$ *satisfying* φ, we have $\mathsf{w}[c] = f_{\mathsf{r}[c]}(v(x_c^1), v(x_c^2), \ldots, v(x_c^{n_c}))$, for every $c \in C$. Notice that if φ is *valid* (i.e. satisfied by all valuations) then $[\![(\mathsf{r}, \varphi)]\!] = [\![r]\!]$ is a $\mathcal{SLRE}$ set of channel contents.

CQDDs inspite of their power are closed under the operations we need:

PROPOSITION 6.1 ([5]). *Given a CQDD χ, a channel c and a message m, there exists two CQDDs χ' and χ'' such that:*

$$\begin{aligned} [\![\chi']\!] &= \{\mathsf{w}[c := m^{-1}\mathsf{w}[c]] \mid \mathsf{w} \in [\![\chi]\!], m \leq \mathsf{w}[c]\} \\ [\![\chi'']\!] &= \{\mathsf{w}[c := \mathsf{w}[c]m] \mid \mathsf{w} \in [\![\chi]\!]\} \end{aligned}$$

and we write $\chi' = ??(c, m, \chi)$ and $\chi'' = !!(c, m, \chi)$.

We are now ready to define the FIFO symbolic representation based on CQDDs. Defining $\mathcal{CQDD} = (CQDD(C, M), \emptyset, ??, !!, [\![\cdot]\!], \cup)$, we get that $\mathcal{CQDD}$ is a FIFO symbolic representation and hence $\mathcal{A}^{\mathcal{CQDD}}_{LTS(F)}$ is an abstraction of $LTS(F)$.

THEOREM 6.2 ([5]). *Every loop of a FIFO automaton is $\mathcal{CQDD}$-representable.*

A straightforward consequence of the theorem is that for any FIFO automaton F, $\nabla_{\mathcal{CQDD}}$ is a sound and complete acceleration for $\mathcal{A}^{\mathcal{CQDD}}_{LTS(F)}$.

6.3. Comparison

The following theorem shows that $\mathcal{SLRE}$ expressible sets of channel contents are exactly the intersection between $\mathcal{QDD}$ expressible sets of channel contents and $\mathcal{CQDD}$ expressible sets of channel contents.

THEOREM 6.3. *Let L be a subset of $(M^*)^C$. The two following assertions are equivalent:*

1. there exists an $\mathcal{SLRE}$ symbolic channel content χ such that $[\![\chi]\!] = L$,

2. there exists a $\mathcal{QDD}$ symbolic channel content χ and a $\mathcal{CQDD}$ symbolic channel content χ' such that $[\![\chi]\!] = [\![\chi']\!] = L$

Proof. We prove both directions.

$\boxed{1. \Rightarrow 2.}$ Assume that there exists an $\mathcal{SLRE}$ symbolic channel content $\chi \in \mathcal{P}_f(SLRE(M)^C)$ such that $[\![\chi]\!] = L$. Notice χ is also a $\mathcal{QDD}$ symbolic channel content. Hence, it remains to prove to prove that $L = [\![\chi']\!]$ for some $\mathcal{CQDD}$ symbolic channel content χ'. We first show the following claim: for every LRE ρ, there exists a SLRE $\rho' = \rho_0 + \rho_1 + \cdots + \rho_m$ such that

(a) we have $[\![\rho']\!] = [\![\rho]\!]$, and,

(b) for every $0 \leq i \leq m$, ρ_i contains none of the following factors: $(au)^*a$, u^*v^* with $u, v \in M^*$ and $a \in M$,

First notice that if a LRE ρ contains a factor of the form $(au)^*a$ then we may safely replace $(au)^*a$ in ρ by $a(ua)^*$, since this transformation preserves the meaning $[\![\rho]\!]$. We repeat this process until there is no more factor of the form $(au)^*a$ in ρ. We then apply the two following transformations (preserving the meaning of ρ) to remove factors of the form u^*v^*.

- if $u^\omega = v^\omega$ then we have $u^{|v|} = v^{|u|}$. Hence $u^*v^* = \sum_{i=0}^{|v|-1} u^i v^*$ and we may thus eliminate u^*v^* from ρ by splitting ρ into a sum ρ'.

• if $u^\omega \neq v^\omega$ then we have $u^{|v|} \neq v^{|u|}$. Hence there exists $u', u'', v', v'' \in \Sigma^*$ and $k, l \in \mathbb{N}$ such that $v^l v' = u^k u'$, $u = u'u''$, $v = v'v''$, $u'' \neq \varepsilon$, $v'' \neq \varepsilon$ and such that u'' and v'' do not start with the same letter. For every $0 \leq i \leq l$ we have $v^i \in u^* u'_i$ and $u = u'_i u''_i$ for some $u'_i, u''_i \in \Sigma^*$. We get that $u^* v^* = \left(\sum_{i=0}^{l} v^i (u''_i u'_i)^* u''_i\right) + v^l v' (u'' u')^* v'' v^*$ and we may thus eliminate $u^* v^*$ from ρ by splitting ρ into a sum ρ' (notice that $(u''u')^* v'' v^*$ contains none of the factors $(au)^* a$, $u^* v^*$).

Moreover, starting from the left of ρ, these transformations do not add any factor of the form $(au)^* a$ in ρ'. Hence we obtain that there exists a SLRE ρ' satisfying conditions (a) and (b) above.

Now using the above claim, we get that for every $\mathsf{r} \in \chi$ and $c \in C$ there exists a family $(\rho_i)_{i \in I_{\mathsf{r},c}}$ of LREs such that

1. we have $[\![\mathsf{r}[c]]\!] = \bigcup_{i \in I_{\mathsf{r},c}} [\![\rho_i]\!]$, and,
2. for every $i \in I_{\mathsf{r},c}$, ρ_i contains none of the following factors: $(au)^* a$, $u^* v^*$ with $u, v \in M^*$ and $a \in M$,

Hence, we get:

$$\begin{aligned} L = [\![\chi]\!] &= \bigcup_{\mathsf{r} \in \chi} \prod_{c \in C} [\![\mathsf{r}[c]]\!] \\ &= \bigcup_{\mathsf{r} \in \chi} \prod_{c \in C} \bigcup_{i \in I_{\mathsf{r},c}} [\![\rho_i]\!] \\ &= \bigcup_{\mathsf{r} \in \chi} \bigcup_{\mathsf{i} \in \prod_{c \in C} I_{\mathsf{r},c}} \prod_{c \in C} [\![\rho_{\mathsf{i}[c]}]\!] \\ &= [\![\chi']\!] \end{aligned}$$

where χ' is the $\mathcal{CQDD}$ symbolic channel content defined by:

$$\chi' = \bigcup_{\mathsf{r} \in \chi} \bigcup_{\mathsf{i} \in \prod_{c \in C} I_{\mathsf{r},c}} (< c \mapsto \rho_{\mathsf{i}[c]} >, true)$$

$\boxed{2. \Rightarrow 1.}$ Assume that there exists a $\mathcal{QDD}$ symbolic channel content $\chi \in \mathcal{P}_f(REG(M)^C)$ and a $\mathcal{CQDD}$ symbolic channel content $\chi' \in CQDD(C, M)$ such that $[\![\chi]\!] = [\![\chi']\!] = L$. Notice that $[\![\chi']\!] \subseteq \bigcup_{(\mathsf{r}', \varphi) \in \chi'} \prod_{c \in C} [\![\mathsf{r}'[c]]\!]$. Hence, we get:

$$\begin{aligned} [\![\chi]\!] &= \left(\bigcup_{\mathsf{r} \in \chi} \prod_{c \in C} [\![\mathsf{r}[c]]\!]\right) \cap \left(\bigcup_{(\mathsf{r}', \varphi) \in \chi'} \prod_{c \in C} [\![\mathsf{r}'[c]]\!]\right) \\ &= \bigcup_{\mathsf{r} \in \chi, (\mathsf{r}', \varphi) \in \chi'} \left(\prod_{c \in C} [\![\mathsf{r}[c]]\!] \cap \prod_{c \in C} [\![\mathsf{r}'[c]]\!]\right) \\ &= \bigcup_{\mathsf{r} \in \chi, (\mathsf{r}', \varphi) \in \chi'} \prod_{c \in C} ([\![\mathsf{r}[c]]\!] \cap [\![\mathsf{r}'[c]]\!]) \end{aligned}$$

Recall that r' is a LRE for every $(\mathsf{r}', \varphi) \in \chi'$. Therefore, according to Proposition 3.2, for every $\mathsf{r} \in \chi$, $(\mathsf{r}', \varphi) \in \chi'$ and $c \in C$ there exists a SLRE $\rho_{\mathsf{r},\mathsf{r}',c}$ such that $\rho_{\mathsf{r},\mathsf{r}',c} =$

$[\![\mathsf{r}[c]]\!] \cap [\![\mathsf{r}'[c]]\!]$. We finally obtain that $L = [\![\chi]\!] = [\![\chi'']\!]$ where χ'' is the $\mathcal{SLRE}$ symbolic channel content defined by:

$$\chi'' = \bigcup_{\mathsf{r}\in\chi,(\mathsf{r},\varphi)\in\chi'} < c \mapsto \rho_{\mathsf{r},\mathsf{r}',c} >$$

■

Remark. A straightforward proof of the if part of Theorem 5.2 follows from the previous theorem combined with theorems 6.2 and 6.1. Indeed, assume that σ is a non counting loop and let χ be a $\mathcal{SLRE}$ symbolic channel content. From Theorem 6.3, we obtain that there exists a $\mathcal{QDD}$ symbolic channel content χ_1 and a $\mathcal{CQDD}$ symbolic channel content χ_2 such that $[\![\chi_1]\!] = [\![\chi_2]\!] = [\![\chi]\!]$. Now, according to theorems 6.2 and 6.1, we get that $\sigma^*([\![\chi_1]\!])$ is $\mathcal{QDD}$ expressible and that $\sigma^*([\![\chi_2]\!])$ is $\mathcal{CQDD}$ expressible. Hence we deduce from Theorem 6.3 that $\sigma^*([\![\chi]\!])$ is $\mathcal{SLRE}$ expressible, and we can conclude that σ is $\mathcal{SLRE}$-representable. Note that the only if part of Theorem 5.2 cannot be proved that way.

However, the algorithm driven by this direct proof is surely not efficient, since it needs to run the algorithms corresponding to theorems 6.2 and 6.1, and it also needs to compute the intersection of their results as in the proof of Theorem 6.3. On the contrary, the proof of Theorem 5.2 leads to a more direct algorithm manipulating only SLREs.

It is readily seen that the $post_{\mathcal{A}^{\mathcal{SLRE}}}$ and $\sqcup$ operations can be computed in *linear* time for the $\mathcal{SLRE}$ based abstraction. Moreover, we have

THEOREM 6.4. *The inclusion and the equivalence problems between SLREs are* coNP-*complete.*

Proof. Since for every SLREs ρ, ρ' we have

1. $[\![\rho]\!] \not\subseteq [\![\rho']\!]$ iff $[\![\rho']\!] \neq [\![\rho]\!] \cup [\![\rho']\!]$, and,
2. $[\![\rho]\!] \neq [\![\rho']\!]$ iff $[\![\rho']\!] \not\subseteq [\![\rho]\!]$ or $[\![\rho]\!] \not\subseteq [\![\rho']\!]$,

it actually suffices to show that the inequivalence problem between SLREs is NP-complete.

To show that inequivalence between SLREs is in NP, we use a result of [27] stating that inequivalence between regular expression denoting *bounded* languages is in NP. Since any SLRE denotes a bounded language, we conclude that inequivalence between SLREs is in NP.

The proof of NP-hardness is actually contained in [34]. It is proved in [34] that inequivalence between regular expressions over a one-letter alphabet $\{a\}$ is NP-complete. However, they actually prove a stronger result, since they reduce in logarithmic space 3–SAT to the problem $\{\rho \in SLRE(\{a\}) \mid [\![\rho]\!] \neq a^*\}$. This shows that inequivalence between SLREs is NP-hard (and this is still true over a one-letter alphabet). ■

Observe that this does not show that the inclusion problem between $\mathcal{SLRE}$ symbolic channel contents is in coNP. The following theorem gives precise complexity results for the inclusion problem between $\mathcal{SLRE}$ (resp. $\mathcal{QDD}$, $\mathcal{CQDD}$) symbolic channel contents.

THEOREM 6.5. *Table 1 gives the complexity of the inclusion and the equivalence problems between $\mathcal{SLRE}$ (resp. $\mathcal{QDD}$, $\mathcal{CQDD}$) symbolic channel contents.*

TABLE 1

Comparison of the inclusion and equivalenece complexity

	inclusion	equivalence
$\mathcal{SLRE}$	coNP-complete	coNP-complete
$\mathcal{QDD}$	PSPACE-complete	PSPACE-complete
$\mathcal{CQDD}$	N2EXPTIME-hard	N2EXPTIME-hard

Proof. First notice that $\mathcal{SLRE}$ (resp. $\mathcal{QDD}$, $\mathcal{CQDD}$) symbolic channel contents are trivially closed under finite union, and a symbolic channel content denoting the union can be computed in linear time. Hence, using the same trick as in the proof of Theorem 6.4, we obtain that it suffice to show the complexity results for the equivalence problem only.

The N2EXPTIME-hardness result for the $\mathcal{CQDD}$ symbolic representation directly follows from the N2EXPTIME-hardness of the decision problem for Presburger Arithmetic [3]. Indeed, given a closed Presburger formula φ, we define the (single channel) CQDD $\chi = \{(a, \varphi)\}$ and we have $[\![\chi]\!] = [\![\emptyset]\!] = \emptyset$ iff φ is false.

The PSPACE-hardness (resp. coNP-hardness) of the equivalence problem between $\mathcal{QDD}$ (resp. $\mathcal{SLRE}$) symbolic channel contents follow from the PSPACE-hardness (resp. coNP-hardness) of the equivalence problem between regular expressions [35] (resp. between SLREs, see Theorem 6.4). We now prove the complexity upper bounds for these two problems.

Let us first focus on the $\mathcal{QDD}$ symbolic representation. Following the ideas of [4], we encode, in linear time, any vector $\mathsf{r} \in REG(M)^C$ of regular expressions as a single regular expression ρ_r over the alphabet $C \times M$, as follows: assuming $C = \{c_1, c_2, \cdots c_n\}$, we define $\rho_\mathsf{r} = \rho_{c_1} \cdot \rho_{c_2} \cdots \rho_{c_n}$ where each ρ_{c_i} is obtained from $\mathsf{r}[c]$ by replacing each letter $m \in M$ by (c_i, m). Observe that for any $\chi, \chi' \in \mathcal{P}_f(REG(M)^C)$ we have:

$$
\begin{array}{rll}
[\![\chi]\!] \subseteq [\![\chi']\!] & \text{iff} & \displaystyle\bigcup_{\mathsf{r}\in\chi} \prod_{c\in C} [\![\mathsf{r}[c]]\!] \;\subseteq\; \bigcup_{\mathsf{r}'\in\chi'} \prod_{c\in C} [\![\mathsf{r}'[c]]\!] \\
& \text{iff} & \displaystyle\bigcup_{\mathsf{r}\in\chi} [\![\rho_\mathsf{r}]\!] \;\subseteq\; \bigcup_{\mathsf{r}'\in\chi'} [\![\rho_{\mathsf{r}'}]\!] \\
& \text{iff} & \displaystyle\left[\!\!\left[\sum_{\mathsf{r}\in\chi} \rho_\mathsf{r}\right]\!\!\right] \;\subseteq\; \left[\!\!\left[\sum_{\mathsf{r}'\in\chi'} \rho_{\mathsf{r}'}\right]\!\!\right]
\end{array}
$$

Thus, we reduce in linear time the equivalence problem for $\mathcal{QDD}$ symbolic channel contents to the equivalence problem for regular expressions, which is known to be belong to PSPACE.

We finally analyse the case of $\mathcal{SLRE}$ symbolic channel contents. We will reuse the previous encoding of C-indexed vectors of regular expressions. Given a vector $\mathsf{r} \in SLRE(M)^C$ of SLREs, the regular expression $\rho_\mathsf{r} \in REG(C \times M)$ is not necessarily a SLRE, but it clearly denotes a flat language. Since flat languages (aka bounded languages) are closed under union, we reduce in linear time the equivalence problem for $\mathcal{SLRE}$ symbolic channel contents to the equivalence problem for regular expressions denoting bounded languages. This conludes the proof, since inequivalence between regular expression denoting *bounded* languages is in NP [27]. ■

Let us remark that the complexity lower bounds of the previous theorem still hold in the single channel case.

TABLE 2
Comparison of the accelerations

	one channel	multiple channels
$\mathcal{SLRE}$	$\nabla_{\mathcal{SLRE}}$: sound, complete	$\nabla_{\mathcal{SLRE}}$: sound, non complete $\overline{\nabla}_{\mathcal{SLRE}}$: non sound, complete
$\mathcal{QDD}$	$\nabla_{\mathcal{QDD}}$: sound, complete	$\nabla_{\mathcal{QDD}}$: sound, non complete
$\mathcal{CQDD}$	$\nabla_{\mathcal{CQDD}}$: sound, complete	$\nabla_{\mathcal{CQDD}}$: sound, complete

Compared to QDDs and CQDDs, SLREs seem to have more practical relevance, since they are *usually enough* to represent sets of FIFO contents and to iterate loops (see Table 2), and they have a *better complexity*. Starting from a single initial state of FIFO automaton with one channel, the generic `SymbolicTree` algorithm applied to these three abstractions/accelerations will give the same result. For FIFO automata with multiple channels, $\nabla_{\mathcal{SLRE}}$ and $\nabla_{\mathcal{QDD}}$ applied to an $\mathcal{SLRE}$ abstract state produce the same regular languages.

7. CONCLUSION

The goal of this paper was to give a general setting for the fast computation of $post^*$ (and pre^* by duality). The contributions of the paper are two fold: the investigation of SLREs as a symbolic representation, and the use of abstractions as a uniform mechanism for explaining symbolic representations and accelerations. In particular, we have been able to investigate the relation between SLREs and other formalisms such as QDDs and CQDDs.

The two algorithms defined in Section 4 become *implementable* when the abstract inclusion $\sqsubseteq$ is decidable, and when the three total functions $post_{\mathcal{A}}$, $\sqcup$ and ∇ are computable. This is the case for the $\mathcal{SLRE}$ based abstraction (and also for the $\mathcal{QDD}$ and $\mathcal{CQDD}$ based abstractions), hence our abstraction $\mathcal{A}^{\mathcal{SLRE}}_{LTS(F)}$ and its two associated accelerations $\nabla_{\mathcal{SLRE}}, \overline{\nabla}_{\mathcal{SLRE}}$ are effective.

Pressburger formulas are usually used to symbolically represent counter values. Comon et al. showed in [12] that the effect of arbitrary iteration of a loop in a multiple counters automaton can be described by a Pressburger formula. Thus, an abstraction based on Pressburger formulas, along with a sound and complete acceleration can be defined for multiple counters automata.

Our general abstraction and acceleration framework can also capture the notions of well structured transition systems [16, 2, 15, 19]. In [16, 15], an acceleration is defined by means of limits (or lubs) in order to compute a finite coverability tree. No acceleration is used in [2, 19] since the abstract domain satisfies the ascending chain condition.

We expect that these notions would carry over to accelerations of parametrized systems [6] and to combinations of symbolic representations — topics we propose to consider in the near future.

ACKNOWLEDGMENT

We thank Peter Habermehl and Richard Mayr [25] for pointing out an error in [35], which led, in a previous version of this paper, to an incorrect statement of Theorem 6.4. They also observed that the proof of NP-hardness of the inequivalence problem between SLREs is contained in [34]. We are also grateful to the anonymous referee for many relevant comments.

REFERENCES

1. P. Abdulla, A. Annichini, and A. Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. *Lecture Notes in Computer Science*, 1579:208–222, 1999.
2. P. A. Abdulla, K. Čerāns, B. Jonsson, and T. Yih-Kuen. General decidability theorems for infinite-state systems. In *Proc. 11th IEEE Symp. Logic in Computer Science (LICS'96), New Brunswick, NJ, USA, July 1996*, pages 313–321, 1996.
3. L. Berman. The complexity of logical theories. *Theoretical Computer Science*, 11:71–77, 1980.
4. B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs. In *Proc. 4th Int. Symp. Static Analysis, Paris, France, September 1997*, volume 1302, pages 172–186. Springer-Verlag, 1997.
5. A. Bouajjani and P. Habermehl. Symbolic reachability analysis of FIFO channel systems with nonregular sets of configurations. In *Proc. 24th Int. Coll. Automata, Languages, and Programming (ICALP'97), Bologna, Italy, July 1997*, volume 1256 of *Lecture Notes in Computer Science*, pages 560–570. Springer-Verlag, 1997.
6. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *Proc. 12th Int. Conf. Computer Aided Verification (CAV'2000), Chicago, USA, July 2000*, volume 1855 of *Lecture Notes in Computer Science*. Springer, 2000. To appear.
7. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, April 1983.
8. J. A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, October 1964.
9. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.
10. G. Cécé. Vérification, analyse et approximations symboliques des automates communicants. Thèse ENS de Cachan, January 1998.
11. H. Comon and V. Cortier. Flatness is not a weakness. In *Proc. 14th Int. Workshop Computer Science Logic (CSL'2000), Fischbachau, Germany, Aug. 2000*, Lecture Notes in Computer Science. Springer, 2000. To appear.
12. H. Comon and Y. Jurski. Multiple counters automata, safety analysis and Presburger arithmetic. In *Proc. 10th Int. Conf. Computer Aided Verification (CAV'98), Vancouver, BC, Canada, June-July 1998*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1998.
13. H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *Proc. 10th Int. Conf. Concurrency Theory (CONCUR'99), Eindhoven, The Netherlands, Aug. 1999*, volume 1664 of *Lecture Notes in Computer Science*, pages 242–257. Springer, 1999.
14. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual Symposium on Principles of Programming Languages*, pages 238–252. ACM SIGACT and SIGPLAN, ACM Press, 1977.
15. E. A. Emerson and K. S. Namjoshi. On model checking for non-deterministic infinite-state systems. In *Proc. 13th IEEE Symp. Logic in Computer Science (LICS'98), Indianapolis, IN, USA, June 1998*, pages 70–80, 1998.
16. A. Finkel. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–179, 1990.
17. A. Finkel and O. Marcé. A minimal symbolic coverability graph for infinite-state comunicating automata. Technical report, LIFAC, 1996.
18. A. Finkel and L. Rosier. A survey on the decidability questions for classes of FIFO nets. In *Advances in Petri Nets 1988*, pages 106–132, Berlin, June 1988. Springer.

19. A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere !, 2000. To appear in *Theoretical Computer Science*, available at `http://www.lsv.ens-cachan.fr/Publis/PAPERS/`.

20. A. Finkel and G. Sutre. Decidability of reachability problems for classes of two counters automata. In *Proc. 17th Ann. Symp. Theoretical Aspects of Computer Science (STACS'2000), Lille, France, Feb. 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 346–357. Springer, 2000.

21. L. Fribourg and H. Olsén. A decompositional approach for computing least fixed-points of Datalog programs with Z-counters. *Constraints*, 2(3/4):305–335, 1997.

22. L. Fribourg and H. Olsén. Proving safety properties of infinite state systems by compilation into Presburger arithmetic. In *Proc. 8th Int. Conf. Concurrency Theory (CONCUR'97), Warsaw, Poland, Jul. 1997*, volume 1243 of *Lecture Notes in Computer Science*, pages 213–227. Springer, 1997.

23. S. Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, 1966.

24. M. G. Gouda, E. M. Gurari, T. Lai, and L. E. Rosier. On deadlock detection in systems of communicating finite state machines. *Computers and Artificial Intelligence*, 6(3):209–228, 1987.

25. P. Habermehl and R. Mayr. A note on SLRE, October 2000. Personal communication.

26. Thomas A. Henzinger and Rupak Majumdar. A classification of symbolic transition systems. In *Proc. 17th Ann. Symp. Theoretical Aspects of Computer Science (STACS'2000), Lille, France, Feb. 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 13–34. Springer, 2000.

27. H. B. Hunt III, D. J. Rosenkrantz, and T. G. Szymanski. On the equivalence, containment, and covering problems for the regular and context-free languages. *Journal of Computer and System Sciences*, 12(2):222–268, April 1976.

28. T. Jéron and C. Jard. Testing for unboundedness of fifo channels. *Theoretical Computer Science*, 113(1):93–117, 1993.

29. R. M. Karp and R. E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.

30. G. Memmi and A. Finkel. An introduction to FIFO nets-monogeneous nets: a subclass of FIFO nets. *Theoretical Computer Science*, 35(2-3):191–214, February 1985.

31. J. K. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Proc. of Protocol Specification, Testing and Verification, VII*, 1987.

32. W. Peng and S. Purushothaman. Data flow analysis of communicating finite state machines. *ACM Transactions on Programming Languages and Systems*, 13(3):399–442, July 1991.

33. D. Perrin. Words. In *Combinatorics on Words, ed. M. Lothaire, Encyclopedia of Mathematics and its Applications*, volume 17. Addison-Wesley, 1983.

34. L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proc. 5th ACM Symp. Theory of Computing (STOC'73), Austin, TX, USA, Apr.-May 1973*, pages 1–9, 1973.

35. S. Yu. Regular languages. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 1, pages 41–111. Springer-Verlag, 1997.