Verification of Communication Protocols Using Abstract Interpretation of FIFO Queues

Tristan Le Gall, Bertrand Jeannet, and Thierry Jéron

IRISA/INRIA Rennes, Campus de Beaulieu,
35042 Rennes cedex, France

Abstract. We address the verification of communication protocols or distributed systems that can be modeled by Communicating Finite State Machines (CFSMs), i.e. a set of sequential machines communicating via unbounded FIFO channels. Unlike recent related works based on acceleration techniques, we propose to apply the Abstract Interpretation approach to such systems, which consists in using approximated representations of sets of configurations. We show that the use of regular languages together with an extrapolation operator provides a simple and elegant method for the analysis of CFSMs, which is moreover often as accurate as acceleration techniques, and in some cases more expressive. Last, when the system has several queues, our method can be implemented either as an attribute-independent analysis or as a more precise (but also more costly) attribute-dependent analysis.

1 Introduction

Communicating Finite State Machines (CFSMs) [1, 2] is a simple model to describe distributed systems exchanging messages over an asynchronous network. This model consists of finite state processes that exchange messages via unbounded FIFO queues. Indeed, unbounded queues provide a useful abstraction that simplifies the semantics of specification languages, and frees the protocol designer from implementation details related to buffering policies and limitations. As a consequence, it is used to define the semantics of standardized protocol specification languages such as SDL and Estelle [3]. Despite its simplicity, the CFSM model cannot be easily verified: reachability is undecidable for CFSM [1], since unbounded queues can be used to simulate the tape of a Turing Machine.

Analysis of communicating systems. Two fundamental approaches have been followed for the analysis of communicating systems in general. One consists of eliminating the need for analyzing FIFO queues contents by adopting a partial order semantics or a so-called *true concurrency* model: when one process sends a message to another process, one just records the information that the emission precedes the reception. The seminal work about event structures [4] leads later to scenario-based models like (High-level) Message Sequence Charts [5, 6] incorporated in UML. The second approach, on which this paper focuses, consists in considering a model with explicit FIFO queues, namely the CFSM model described above, and in analyzing their possible contents during the execution of the system.

M. Johnson and V. Vene (Eds.): AMAST 2006, LNCS 4019, pp. 204–219, 2006.

The undecidability of the reachability of CFSM [1] does not prevent any verification attempt, but requires to give up with at least one of the following properties of an ideal method: an ideal method should indeed be (*i*) general (*i.e.* address any CFSM system), (*ii*) always terminate, and (*iii*) deliver exact results. Two main directions have mainly been explored so far: the first one abandon property (*i*) by simplifying the model or considering only a subclass of it, whereas the second one prefer to abandon property (*ii*) by looking only for efficient semi-algorithms that may not terminate but deliver exact results "often enough". *Lossy channels systems* illustrate both directions. They are CFSMs where the channels can lose messages at any time. Those systems are easier to verify than perfect channels systems [7]: the reachability problem is decidable, but there is no effective algorithm to compute the reachability set. However, an on-the-fly analysis semi-algorithm based on *simple regular expressions* is given in [8]. This algorithm can "accelerate" loops, that is, it is able to compute the effect of any *meta transition* (loops in the control transition systems). The termination problem remains because the number of loops is potentially infinite. This *acceleration* approach has been generalized to standard CFSMs systems (*cf.* section 3), leading to various semi-algorithms applying the acceleration principle on different representations for queues contents.

We propose here an alternative tradeoff to face the undecidability problem, which is to keep generality and termination (properties (*i*) and (*ii*)) and to give up with the exactness of the results (property (*iii*)). When analyzing CFSMs, this consists in replacing in dataflow equations, sets of FIFO channel configurations by abstract properties belonging to a lattice. Such a transformation results in conservative approximations: we will be able to prove a safety property, or the non-reachability of a state, but not to prove that a property is false or that a state is effectively reachable. The abstractions we propose in this paper are all based on regular languages, which exhibit among nice properties the closure under all Boolean operations, and a canonical representation with deterministic and minimized finite automata.

Contributions. We show in this paper that our abstract-interpretation based method presents several advantages: it is arguably technically less involving than acceleration-based techniques, it often returns exact results on cases where the acceleration techniques terminate, and relevant information in the other cases where the acceleration techniques do not terminate and do not provide any result, either because the control structure of the system is too intricate, or because the reachable set cannot be represented with the chosen representation. Our method can also be seen as complementary to acceleration techniques when they fail. Last, although acceleration techniques have been applied to other infinite datatypes (counters [9], etc), it is not clear whether they can be easily *combined*, whereas general methods are available for combining different abstract domains.

Outline. We introduce in section 2 the model of communicating finite state machines, and the analysis problem we address, namely reachability analysis. We discuss the related works in section 3. We then explain our approach for the reachability analysis of CFSMs in the case of one FIFO channel (section 4).

In section 5 we generalize it to the case of several FIFO channels. We implemented our method and we present in section 6 a few case studies on which we experimented it, and we compare it with other techniques.

2 Finite Automata and Communicating Finite State Machines

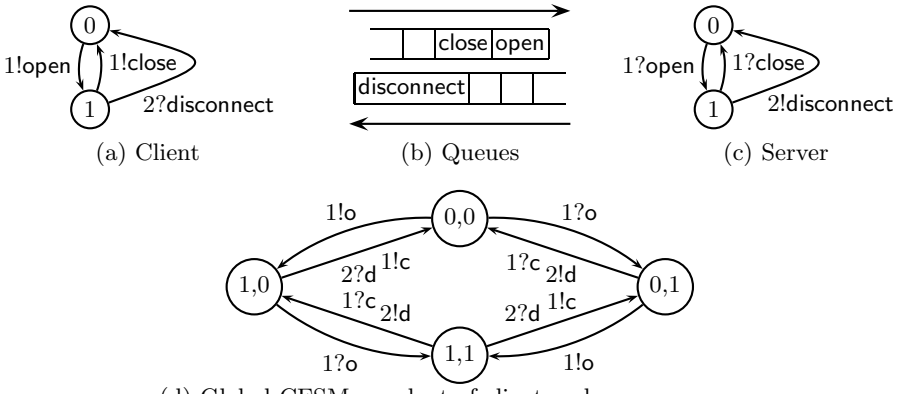
Finite automata. A finite automaton is a 5-tuple $\mathcal{M} = (Q, \Sigma, Q_0, Q_f, \rightarrow)$ where Q is a finite set of states, Σ a finite alphabet, $Q_0, Q_f \subseteq Q$ are the sets of initial and final states, and $\rightarrow \subseteq Q \times \Sigma \times Q$ is the transition relation. The relation $\rightarrow$ is extended on words as the smallest relation $\Rightarrow \subseteq Q \times \Sigma^* \times Q$ satisfying: (i) $\forall q \in Q \ : \ q \stackrel{\epsilon}{\Rightarrow} q$ and (ii) if $q \stackrel{a}{\rightarrow} q'$ and $q' \stackrel{w}{\Rightarrow} q''$, then $q \stackrel{a \cdot w}{\Rightarrow} q''$. $\mathcal{M}$ is deterministic if $Q_0 = \{q_0\}$ and if $\rightarrow$ defines a function $Q \times \Sigma \rightarrow Q$. A word $w \in \Sigma^*$ is *accepted by* $\mathcal{M}$ if $\exists q_0 \in Q_0, \exists q_f \in Q_f : q_0 \stackrel{w}{\Rightarrow} q_f$. The language $L(\mathcal{M})$ accepted by $\mathcal{M}$ is the set of accepted words. Conversely, given a regular language $L \in \wp(\Sigma^*)$, the unique (up to isomorphism) minimal deterministic automaton (MDA) accepting L is denoted by $\mathcal{M}(L)$. The set of regular languages on alphabet Σ is denoted by $\mathcal{R}(\Sigma)$. Given an automaton $\mathcal{M} = (Q, \Sigma, Q_0, Q_f, \rightarrow)$ and an equivalence relation $\simeq$ on its states, $\mathcal{M}/\simeq = (Q/\simeq, \Sigma, \widetilde{Q_0}, \widetilde{Q_f}, \widetilde{\rightarrow})$ denotes the *quotient automaton* defined in the usual way : the states of $\mathcal{M}/\simeq$ are the equivalence classes of $\simeq$, $\overline{q} \in Q/\simeq$ is an initial (resp. final) state if one state of this equivalence class is an initial (resp. final) state of $\mathcal{M}$, and $(\overline{q}, a, \overline{q'}) \in \widetilde{\rightarrow}$ if $\exists q \in \overline{q}, \exists q' \in \overline{q'}, (q, a, q') \in \rightarrow$. For any equivalence relation $\simeq$, we have $L(\mathcal{M}) \subseteq L(\mathcal{M}/\simeq)$.

Definition 1 (CFSM). *A Communicating Finite State Machine is given by a tuple* (C, Σ, c_0, Δ) *where:*

- C *is a finite set of locations (control states)*
- $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \cdots \cup \Sigma_n$ *is a finite alphabet of messages, where* Σ_i *denotes the alphabet of messages that can be stored in queue* i*;*
- $c_0 \in C$ *is the initial location;*
- $\Delta \subseteq C \times A \times C$ *is a finite set of transitions, where* $A = \bigcup_i \{i\} \times \{!, ?\} \times \Sigma_i$ *is the set of actions. An action can be*
  - *either an output* $i!m$*: "the message* m *is sent through the queue* i*";*
  - *or an input* $i?m$*: "the message* m *is received from the queue* i*".*

In the examples, we define CFSMs in terms of an asynchronous product of finite state machines (FSMs) reading and writing on queues.

Example 1. The connexion/deconnexion protocol between two machines is the following (Fig. 1): the client can open a session by sending the message open to the server. Once a session is open, the client may close it on its own by sending the message close or on the demand of the server if it receives the message disconnect. The server can read the request messages open and close, and ask for a session closure.



(a) Client (b) Queues (c) Server

(d) Global CFSM: product of client and server processes

Fig. 1. The connexion/deconnexion protocol

Semantics. The semantics of a CFSM (C, Σ, c_0, Δ) is given as a labelled transition system (LTS) $\langle Q, Q_0, A, \rightarrow \rangle$ where

- $Q = C \times \Sigma_1^* \times \cdots \times \Sigma_n^*$ is the set of states;
- $Q_0 = \{\langle c_0, \varepsilon, \ldots, \varepsilon \rangle\}$ is the set of the initial states;
- A is the alphabet of actions (*cf.* Def. 1).
- $\rightarrow$ is defined by the two rules:

$$\frac{(c_1, i!m, c_2) \in \Delta \qquad w_i' = w_i \cdot m}{\langle c_1, w_1, \ldots, w_i, \ldots, w_n \rangle \rightarrow \langle c_2, w_1, \ldots, w_i', \ldots, w_n \rangle}$$

$$\frac{(c_1, i?m, c_2) \in \Delta \qquad w_i = m.w_i'}{\langle c_1, w_1, \ldots, w_i, \ldots, w_n \rangle \rightarrow \langle c_2, w_1, \ldots, w_i', \ldots, w_n \rangle}$$

A global state of a CFSM is thus a tuple $\langle c, w_1, \ldots, w_n \rangle \in C \times \Sigma_1^* \times \cdots \times \Sigma_n^*$ where c is the current location and w_i is a finite word on Σ_i representing the content of queue i. At the beginning, all queues are empty, so the *initial state* is $\langle c_0, \varepsilon, \ldots, \varepsilon \rangle$. The reflexive transitive closure $\rightarrow^*$ is defined as usual. A state $\langle c, w_1, \ldots, w_n \rangle$ is *reachable* if $\langle c_0, \varepsilon, \ldots, \varepsilon \rangle \rightarrow^* \langle c, w_1, \ldots, w_n \rangle$. The *reachability set* is the set of all states that are reachable. Computing this set is the purpose of the reachability analysis. We can achieve this computation by solving a fix-point equation, as shown in the next paragrah.

Forward collecting semantics and reachability analysis of a CFSM. The forward collecting semantics defines the semantics of a system in terms of its reachable set. A set of states $X \in \wp(Q) = \wp(C \times \Sigma_1^* \times \cdots \times \Sigma_n^*)$ can be viewed as a map $X : C \rightarrow \wp(\Sigma_1^* \times \cdots \times \Sigma_n^*)$ associating a control state c with a language $X(c)$ representing all possible contents of the queues when being in the control state c. The forward semantics of actions $[\![a]\!] : \wp(\Sigma_1^* \times \cdots \times \Sigma_n^*) \rightarrow \wp(\Sigma_1^* \times \cdots \times \Sigma_n^*)$ is defined as:

$$[\![i!m]\!](L) = \{\langle w_1, \ldots, w_i \cdot m, \ldots, w_n \rangle | \langle w_1, \ldots, w_i, \ldots, w_n \rangle \in L\} \quad (1)$$

$$[\![i?m]\!](L) = \{\langle w_1, \ldots, w_i, \ldots, w_n \rangle | \langle w_1, \ldots, m \cdot w_i, \ldots, w_n \rangle \in L\} \quad (2)$$

$[\![i!m]\!]$ (resp. $[\![i?m]\!]$) associates to a set of queues contents the possible queues contents after the output (resp. the input) of the message m on the queue i, according to the operational semantics of CFSM. Using the inductive definition of reachability — a state is reachable either because it is initial, or because it is the immediate successor of a reachable state —, the reachability set RS is defined as the least solution of the fixpoint equation

$$\forall c \in C,\ X(c) = Q_0(c) \cup \bigcup_{(c',a,c)\in\Delta} [\![a]\!](X(c')) \tag{3}$$

where Q_0 is the initial set of states. Although there is no general algorithm that can compute exactly such a reachability set [1], a number of semi-algorithms that compute the reachability set in some cases have been designed and are described in the next section.

3 Related Works

Semi-algorithms based on the acceleration paradigm. The acceleration paradigm is a popular paradigm for infinite state systems, which we describe in the specific case of CFSM. Eq. (3) is difficult to solve in presence of cycles in the control graph, because iterative solving using Kleene's theorem will not converge. Now, assuming a canonical representation $\mathcal{L}$ for queue contents, given a loop $\theta \stackrel{\triangle}{=} c = c_0 \stackrel{a_1}{\to} c_1 \stackrel{a_2}{\to} \ldots \stackrel{a_k}{\to} c_k = c$ and a language $L \in \mathcal{L}$, we may compute in a single step the effect of the loop θ, i.e. finding a language $[\![\theta^*]\!](L) \in \mathcal{L}$ representing the set of states that can be reached from a state in L following the loop θ an arbitrary number of times. Then, when exploring the state space, we can substitute the entire loop by the single *meta-transition* θ^*. However, even if each loop may be accelerated, we still have to explore an infinite transition system since there is an infinite number of loops. We may exploit some termination conditions [10] or use heuristics that lead to semi-algorithms: for example, we may "flatten" the transition system and find a proper exploration order [9]. In the cases of systems with FIFO channels, this technique has been applied with different kind of representations, depicted in Tab. 1. Usually only forward analysis has been studied. Observe that when several channels are involved in a loop, with some representations, the acceleration is not always possible. [11] provides a detailed comparison of the cited references.

Algorithms based on transducer iterations. Instead of extrapolating sequences of values, one may also extrapolate the full relations $L_{i+1} = R(L_i)$ linking two successive terms, represented as a regular transducer R (in this case, the full state is encoded as a regular word). The computation of the transducer R^* allows the computation of the reachability set. This *regular model-checking* paradigm [14] has mainly been applied to networks of finite state machines. A method to compute the transducer R^* is given in [15], but will not work for any CFSM. [16, 17] define extrapolation operators to compute an over-approximation of R^*, but has experimented them only on one lossy FIFO system [17].

Table 1. Acceleration techniques on CFSMs

queue	representation and typical example	attr.[a] dependent	acceleration with [b] single/several queue	ref.
lossy	SRE[1] : $\sum(a+\epsilon)+(a_1+\ldots+a_m)^*$	no	always / always	[8]
perfect	SLRE[2]: $\sum a_1a_2(b_1b_2)^*a_3(b_3)^*(b_4)^*\ldots$	no	always / sometimes	[11]
perfect	QDD[3] : n-dim regular expression	yes	always / sometimes	[12]
perfect	CQDD[4]: $\sum a_1^{p_1}a_2^{p_2}x_1^{q_1}x_2^{q_2} \mid p_1+2q_1 \leq p_2+q_2$	yes	always / always	[13]

[a] yes if one expression for all queues, no if one expression for each queue
[b] ability to exactly compute the effect of meta-transition
[1] Simple Regular Expressions [2] SemiLinear Regular Expressions
[3] Queue Decision Diagrams [4] Constrained QDD, using Presburger formulas

Decidable subclasses of CFSMs. Reachability has been shown decidable for *monogeneous* [18], *linear* [19] or half-duplex [20] CFSMs. Allowing the channels to be lossy makes also the problem decidable [21, 7]. A recent research direction focuses on probabilisitic lossy channels [22].

Approximated techniques. Besides techniques based on the generation of finite abstract models that are then model-checked,

abstractions have also been experimented on FIFO queues using the classical dataflow analysis framework, hence restricting to lattices of properties satisfying the ascending chain condition (*i.e.* there is no infinite ascending chain). For instance, [23] proposes an analysis that infers the emptiness property and the possible values of the first element to be read in queues. [24] proposes a "widening operator" for decreasing sequences of regular languages, in the same spirit as [16]. However it does not guarantee the convergence of the sequence.

4 Analyzing Systems with Only One Queue

In this section we consider the simple case of CFSMs with a single FIFO queue, on which we describe our method based on abstract interpretation [25].

With a single queue, the concrete state-space has the structure $C \to \wp(\Sigma^*)$, and it will be abstracted by the set $C \to A$, where A is an abstract lattice equipped with a meaning or concretization function $\gamma : A \to \wp(\Sigma^*)$ (*i.e.* γ is monotone and $\gamma(\bot) = \emptyset$). We will consider for A the set of regular languages $\mathcal{R}(\Sigma)$ over Σ, with $\gamma : \mathcal{R}(\Sigma) \to \wp(\Sigma^*)$ being the identity. This simple solution presents several interesting properties:

- $\mathcal{R}(\Sigma)$ is closed under union, intersection, negation and semantic transformers $[\![!m]\!]$ (corresp. to concatenation) and $[\![?m]\!]$ (corresp. to the derivative operator of [26]). Moreover, $Q_0 = \{\langle c_0, \epsilon\rangle\}$ is regular, so that all operators involved in Eq. (3) can be transposed to $\mathcal{R}(\Sigma)$ without loss of information.
- From a computational point of view, regular languages have as a standard canonical representation the minimal deterministic automaton (MDA) recognizing them.

As a consequence, we only have to define a suitable widening operator to ensure convergence of iterative resolution of Eq. (3). Indeed, the lattice $\mathcal{R}(\Sigma)$ does not satisfy the ascending chain condition and is even not complete[1]: it is well-known that the monotone sequence $L_n = \{a^k b^k \mid k \leq n\}$ converges towards a context-free language which is not regular.

Generally speaking, a widening operator is a binary operator $\nabla : A \times A \to A$ satisfying technical conditions (*c.f.* proposition 1) that ensure, in the context of the iterative resolution of a fixpoint equation $X = F(X)$, that the sequence $X_0 = F(\bot), X_{i+1} = X_i \nabla F(X_i)$ converges in a finite number of steps towards a post-fixpoint of F. In general, a widening operator tries to capture and to extrapolate the difference between its two arguments X_i and $F(X_i)$, by making the hypothesis that the difference will be repeated in the sequence X_i, $F(X_i)$, $F(F(X_i))$,.... The main difference with acceleration techniques is that the widening, at least in its basic definition, does not exploit the semantic function F (which is defined by the analyzed system), but is defined solely on abstract values. This is both a weakness — it is then more difficult to make a good or even an exact guess, and a strength — a highly complex function F is not a difficulty, whereas acceleration-based techniques may fail in such cases (non-flat systems, nested loops, ...).

4.1 Widening Operator

In our case, the choice of ∇ is all the more important as all approximations performed by the analysis will depend on its application. Because of the FIFO operations, the widening operator should remain precise for both the begining and the end of the queue. It also should induce intuitive approximations. In [27], a widening operator for regular languages was mentioned. We will adapt this operator to regular languages representing the content of a FIFO channel.

This widening operator will be based on an extensive and idempotent operator $\rho_k : \mathcal{R}(\Sigma) \to \mathcal{R}(\Sigma)$ (*i.e.* $\rho_k(X) \supseteq X$ and $\rho_k \circ \rho_k = \rho_k$), where $k \in \mathbb{N}$ is a parameter. ρ_k will induce a widening operator defined by $X_1 \nabla_k X_2 = \rho_k(X_1 \cup X_2)$. Thus, the proposed widening does not work by extrapolating a difference, but by simplifying the regular languages generated during the iterative resolution. The operator ρ_k is defined on a language L by considering the automaton $\mathcal{M}(L)$ quotiented by a bisimulation up to depth k.

Definition 2 (Bisimulation of depth k). *Let $(Q, \Sigma, Q_0, Q_f, \to)$ be a minimal deterministic automaton and* $\mathrm{col} : Q \to [1..N]$ *a color function defining an equivalence relation $q_1 \approx_{\mathrm{col}} q_2 \Leftrightarrow \mathrm{col}(q_1) = \mathrm{col}(q_2)$. For $k \geq 0$, the smallest bisimulation of depth k finer than $\approx_{\mathrm{col}}$ is defined inductively by: $\forall q_1, q_2 \in Q$,*

$$q_1 \approx_0^{\mathrm{col}} q_2 \text{ iff } q_1 \approx_{\mathrm{col}} q_2$$

$$q_1 \approx_{k+1}^{\mathrm{col}} q_2 \text{ iff } \begin{cases} q_1 \approx_k^{\mathrm{col}} q_2 \\ \forall a \in \Sigma, \forall q_1' \in Q,\ q_1 \xrightarrow{a} q_1' \implies \exists q_2' \in Q \ :\ q_2 \xrightarrow{a} q_2' \wedge q_1' \approx_k^{\mathrm{col}} q_2' \\ \forall a \in \Sigma, \forall q_2' \in Q,\ q_2 \xrightarrow{a} q_2' \implies \exists q_1' \in Q \ :\ q_1 \xrightarrow{a} q_1' \wedge q_1' \approx_k^{\mathrm{col}} q_2' \end{cases}$$

[1] It is precisely because $A = \mathcal{R}(\Sigma)$ is not complete that we cannot define an abstraction function $\alpha : \wp(\Sigma^*) \to \mathcal{R}(\Sigma)$ as it is usually done in abstract interpretation.

In this section, we consider the *standard color function*, which uses $N = 4$ colours for separating initial and final states from other states:

$$\mathrm{col}(q) = 1 \text{ if } q \in Q_0 \cap Q_f,\ 2 \text{ if } q \in Q_f \setminus Q_0,\ 3 \text{ if } q \in Q_0 \setminus Q_f,\ 4 \text{ otherwise} \quad (4)$$

Definition 3 (Operator ρ_k^{col}.). *Given a bisimulation relation $\approx_k^{\mathrm{col}}$ of depth k the operator $\rho_k^{\mathrm{col}} : \mathcal{R}(\Sigma) \to \mathcal{R}(\Sigma)$ is defined by quotienting the MDA of L:*

$$\rho_k^{\mathrm{col}}(L) = \mathcal{L}(\mathcal{M}(L)/\approx_k^{\mathrm{col}})$$

ρ_k^{col} is extensive as being defined by a quotient automaton, and it is idempotent as a consequence of $\approx_k^{\mathrm{col}}$ being a bisimulation relation. As $\approx_{k+1}^{\mathrm{col}} \subseteq \approx_k^{\mathrm{col}}$, we also have $\forall L \in \mathcal{R}(\Sigma) : \rho_{k+1}(L) \subseteq \rho_k(L)$. However, ρ_k is not monotone, as shown by the following example: $a^4 \subseteq a^4 + a^2b$, but $\rho_1(a^4) = a^3a^*$ is not comparable to $\rho_1(a^4 + a^2b) = a^4 + a^2b$.

Definition 4 (Widening operator ∇_k^{col}). *Given an integer $k \geq 0$ and a color function* col*, we define a binary operator $\nabla_k^{\mathrm{col}} : \mathcal{R}(\Sigma) \times \mathcal{R}(\Sigma) \to \mathcal{R}(\Sigma)$:*

$$L_1 \nabla_k^{\mathrm{col}} L_2 \stackrel{\triangle}{=} \rho_k^{\mathrm{col}}(L_1 \cup L_2)$$

Proposition 1. *∇_k^{col} is a widening operator for $\mathcal{R}(\Sigma)$ in the sense of [25]:*

1. *$L_1 \cup L_2 \subseteq L_1 \nabla_k^{\mathrm{col}} L_2$;*
2. *For any increasing chain $(L_0 \subseteq L_1 \subseteq \ldots)$, the increasing chain defined by $L'_0 = L_0,\ L'_{i+1} = L'_i \nabla_k^{\mathrm{col}} L_{i+1}$ is not strictly increasing (it stabilizes after a finite number of steps).*

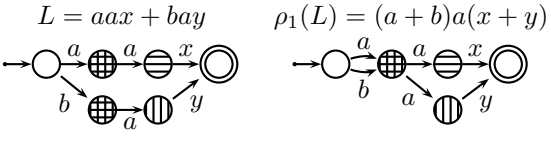
This property ensures the global correctness of our analysis [25].

Proof. 1. The language recognized by a quotient automaton is a superset of the language of the initial automaton. 2. Given a deterministic automaton $(Q, \Sigma, Q_0, Q_f, \to)$ and a color function $\mathrm{col} : Q \to [1..N]$, we have $|Q/\approx_k^{\mathrm{col}}| \leq N^{|\Sigma|^{k+1}} \times 2^{|\Sigma|^k}$ (proved in [28]). Thus the set $\{\rho_k^{\mathrm{col}}(L) \,|\, L \in \mathcal{R}(\Sigma)\}$ is finite.

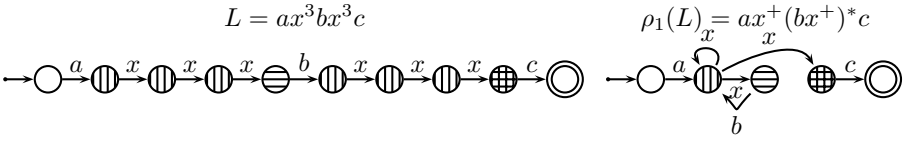
4.2 Effects of the Widening Operator

We analyze here in detail the effect of the extensive operator ρ_k on a language, using the color function of Eq. (4).

Sum of languages: If $L = L_1 \cup L_2$, the widening operator may merge some subwords of L_1 with subwords of L_2. For instance, $\rho_1(aax + bay) = (a + b)a(x + y)$; we thus lose the property " we have an 'a' at the beginning of the queue iff we have an 'x' at the end".



Repetition: an important effect of ρ_k is to introduce Kleene closures in regular expressions. We have $\rho_k(a^n) = \left|\begin{matrix} a^{k+2}a^* & \text{if } k < n-2 \\ a^n & \text{otherwise} \end{matrix}\right|$: the repetition of a letter beyond some number is thus abstracted by an unbounded repetition. The same happens for the repetition of bounded-length words: for $n \geq 3$, $\rho_k((a_1 \dots a_k)^n) = (a_1 \dots a_k)(a_1 \dots a_k)^*$. If the system allows arbitrarily-long channel contents, this approximation can guess the limit of the fix-point computation. If a letter is repeated at different places, the two Kleene stars may be merged: for instance $\rho_1(ax^3bx^3c) = ax^+(bx^+)^*c$, instead of the (preferable) ax^+bx^+c:



One can improve the widening for the two previous situations, by considering a color function col2 which also separates states according to the set of letters already encountered from the initial states. One has $\rho_1^{\text{col2}}(L) = ax^+bx^+c$. This allows to propagate non-local properties in the FIFO queue.

Suffixes and prefixes: we have the following properties:

Proposition 2. *[28] L and $\rho_k(L)$ have the same set of prefixes of length 1 and the same set of suffixes of length less or equal to k.*

Thus, the k last messages written in a queue are not abstracted. As a consequence, we wait enough before trying to capture some regularity with the operator ρ_k. Notice than one can improve the result for prefixes by combining forward with *backward* bisimulation relations.

Surprisingly, this simple widening has not yet been experimented for the analysis of CFSMs. Our contribution here is to adapt for FIFO queues the widening mentioned in [27], by choosing an appropriate color function, and to demonstrate its practical relevance in this context (*c.f.* section 6).

4.3 Complexity of the Analysis

The operations we perform on finite automata are polynomial and rather efficient in practice. The complexity of our analysis depends also on the number of steps of the fixpoint computation. This number is quite small on the examples of section 6 (≤ 12, with $\rho_{k\leq 2}$), but the theoretical bound is exponential in the size of the alphabet and double-exponential in k. We conjecture than even on larger examples, the practical complexity remains much below this bound.

5 Systems with Several Queues

We now come back to the general case where several queues are to be analyzed. In this case, we must choose whereas we analyse each queue independently, using the

method of the previous section, or we analyse all the queues together. In the first case, according to the classification of [29], we obtain an *attribute-independent* analysis based on a *non-relational* abstraction, because properties on different queues are not inter-related. In the second case, we obtain an *attribute-dependent* analysis based on a *relational* abstraction, in which one can represent properties like "queue 1 contains 'a' messages iff queue 2 contains 'b' messages". We propose both solutions.

Concrete representation. In the previous section, a configuration was a word. Now a configuration is defined by n words $w_1, \ldots, w_n$ which can be represented:

1. as a vector of words $\langle w_1, \ldots, w_n \rangle$
2. as a single word $w_1 \sharp \ldots \sharp w_n$ obtained by concatenation and the addition of a separation letter $\sharp$
3. or as an "interleaved" word $w_1^0 \ldots w_n^0 w_1^1 \ldots w_n^1 \ldots$

The third representation is used for representing sets of unbounded integer vectors with NDDs [30], but it is not suited to the FIFO operations. We will consider the two first representations that naturally define two different analyses.

5.1 Non-relational Abstraction

Here we adopt the view of a configuration as a vector of words, and we abstract each component independently: we take

$$A^{nr} = \mathcal{R}(\Sigma_1) \times \cdots \times \mathcal{R}(\Sigma_n)$$

as an abstract lattice, ordered component-wise. The meaning function $\gamma^{nr} : A^{nr} \to \wp(\Sigma_1^* \times \cdots \times \Sigma_n^*)$ is defined by

$$\gamma^{nr}(\langle L_1, \ldots, L_n \rangle) = \gamma(L_1) \times \cdots \times \gamma(L_n)$$

The widening ∇_k of section 4 is extended to A^{nr} component-wise: $\langle L_1, \ldots, L_n \rangle \nabla_k \langle L'_1, \ldots, L'_n \rangle = \langle L_1 \nabla_k L'_1, \ldots, L_n \nabla_k L'_n \rangle$, which defines a proper widening operator. Sending or receiving a message on the queue i consists in modifying the component i of the abstract value. In this lattice, the least upper bound ("the union") is no longer exact, because of the cartesian product. For example, the upper bound of the values $\langle a, x \rangle$ and $\langle b, y \rangle$ is the language $\langle a{+}b, x{+}y \rangle$. Hence, the loss of information is no longer only due to the widening operator.

5.2 Relational Abstraction

If we adopt instead the view of a configuration as a concatenated word, we obtain the QDD representation of [12], to which we apply the principles of section 4:

$$A^r = \mathcal{R}(\Sigma \cup \{\sharp\}) \tag{5}$$

$$\gamma^r(X) = \{\langle w_1, \ldots, w_n \rangle \in \Sigma_1^* \times \ldots \times \Sigma_n^* \mid w_1 \sharp \ldots \sharp w_n \in X\} \tag{6}$$

We implicitly restrict A^r to sets of concatenated words of the form described above. The only difference with [12] is the use of widening instead of acceleration. This representation allows to represent relations or dependencies between queues. For instance the language L of Fig. 2 encodes the relation "the queue 1 starts with an 'a' iff the queue 2 contains an 'x' ".

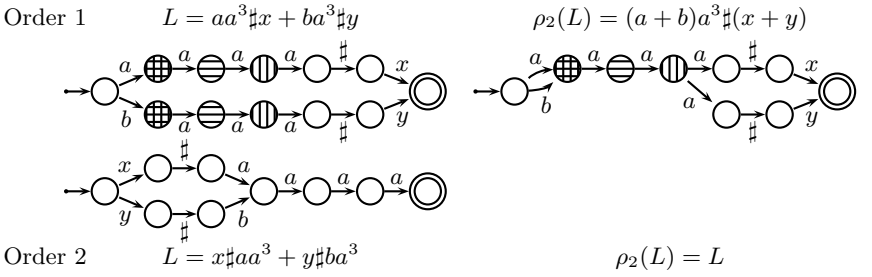


Fig. 2. Widening and ordering of queues in concatenated words

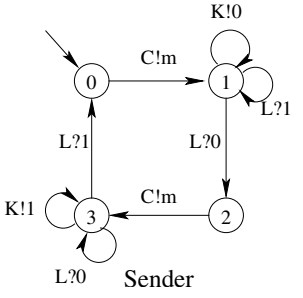
Operations. The union, intersection and inclusion test operations are the natural extensions of their counterpart for an automaton representing a single queue. However, we have to adapt the operations $[\![i!m]\!]$, $[\![i?m]\!]$ and ∇_k. As each word recognized by a MDA $M = \mathcal{M}(L)$ with $L \in \mathcal{R}(\Sigma \cup \{\sharp\})$ is a concatenated word separated by $\sharp$ letters, each state $q \in Q$ of M can be associated to one queue-content by a function $\eta : Q \to [1..n]$, and can be characterized as initial, and/or final for this queue [12, 28]. Given such a partition, the operations $[\![i!m]\!]$ and $[\![i?m]\!]$ are easily implemented. Concerning the widening operator, it should avoid to merge the different queue contents, and preserve the invariant that each word has $n-1$ $\sharp$ letters. We thus adapt the standard color function, which uses now $N = 4n$ colours:

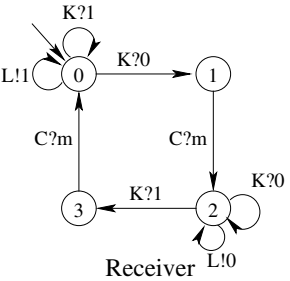
$$\mathrm{col}(q) = \begin{cases} 4 * \eta(q) - 3 & \text{if } q \text{ is both an initial and a final state for the queue } \eta(q) \\ 4 * \eta(q) - 2 & \text{if } q \text{ is a final (but not initial) state for the queue } \eta(q) \\ 4 * \eta(q) - 1 & \text{if } q \text{ is an initial (but not final) state for the queue } \eta(q) \\ 4 * \eta(q) & \text{otherwise} \end{cases} \tag{7}$$

Impact of the ordering. A natural question arises: to which extent is our relational analysis dependent on the chosen ordering for queues ? All the exact operations, which do not lose information, do not depend on it. However, the widening is dependent on the ordering of queues, as shown by the example of Fig. 2. Consequently, our analysis depends on the ordering. A widening operator which would be independent of the ordering would have been more satisfactory, but we did not find out yet such a widening operator, with good properties w.r.t. precision and efficiency (see the discussion in [28]).

6 Experiments and Comparisons

The approach we followed for the analysis was to sacrifice exactness for universality of the analyzed model and convergence guarantee. Of course such an approach is relevant only if the approximations introduced are not too strong, and if they still allow to obtain interesting results. In order to perform this experimental evaluation, we implemented both the non-relational and the relational abstractions, and we connected them to a generic fixpoint calculator, that takes





Sender/ Receiver	Contents K#L#C
0/0	$1^*\#1^*\#\varepsilon$
0/1	$\emptyset$
0/2	$\emptyset$
0/3	$\emptyset$
1/0	$1^*0^*\#1^*\#m$
1/1	$0^*\#1^*\#m$
1/2	$0^*\#1^*0^*\#\varepsilon$
1/3	$\emptyset$
2/0	$\emptyset$
2/1	$\emptyset$
2/2	$0^*\#0^*\#\varepsilon$
2/3	$\emptyset$
3/0	$1^*\#0^*1^*\#\varepsilon$
3/1	$\emptyset$
3/2	$0^*1^*\#0^*\#m$
3/3	$1^*\#0^*\#m$

Fig. 3. The alternating bit protocol

care of the iterative resolution of fixpoint equations and applies widening following the principles of [31]. All our experiments used the ∇_1 widening operator based on the standard color function, and returned their result in less than 1 sec. on a 2 GHz Intel™ Pentium computer. The fixpoint was obtained in 7 to 12 iteration steps, depending on the examples.

The Alternating Bit Protocol (ABP) is a data-transmission protocol, between a sender S and a receiver R. S transmits some data package m through a FIFO channel C and R and S exchange some information (one-bit messages) through two channels K and L (Fig. 3). We performed a relational analysis of the CFSM modeling this protocol (Fig. 3). It shows that some control states are not reachable and that there is at most one message in data channel C. As in [12, 32], we obtain the exact result. Notice that in this case, a simpler non-relational analysis delivers the same results.

The connexion/deconnexion protocol, defined in Example 1, demonstrates the usefulness of a relational analysis:

Relational Analysis

Client/ Server	Queue 1 # Queue 2
0/0	$(co)^*(oc)^*\#\varepsilon \ + \ c(oc)^*\#d$
1/0	$(co)^*(oc)^*o\#\varepsilon \ + \ (co)^*\#d$
0/1	$c(oc)^*\#\varepsilon$
1/1	$(co)^*\#\varepsilon$

Non-Relational Analysis

Client/ Server	Queue 1	Q.2
0/0	$o^* + (o^*c)^+(\varepsilon + o^+ + o^+c)$	d^*
1/0	$(o^*c)^*o^+$	d^*
0/1	$o^* + (o^*c)^+(\varepsilon + o^+ + o^+c)$	d^*
1/1	$o^+ + o^*(co^+)^+$	d^*

The result given by the relational analysis happens to be the exact reachability set, unlike the non-relational one. The non-relational analysis misses the fact that there is at most one d in the second queue, which induces many approximations.

A non-regular example. Our abstraction can deal with cases where the reachability set is not regular. Let us consider the CFSM depicted in Fig. 4. Each

process can send a message a or c, and a synchronisation is guaranteed by the messages b and d.

In location (0/0), the content of the queues will be $a^n\sharp\varepsilon\sharp c^n\sharp\varepsilon$ with $n \geq 0$. This set is non-regular, and thus cannot be represented by a regular expression. Our method will find an over-approximation of the exact reachability set. In location (0/0) the queue-content we found is represented by the language :

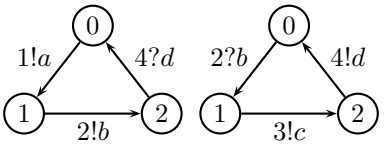


Fig. 4. A non-regular protocol

$$L_{(0/0)} = \varepsilon\sharp\varepsilon\sharp\varepsilon\sharp\varepsilon + a\sharp\varepsilon\sharp c\sharp\varepsilon + aaa^*\sharp\varepsilon\sharp ccc^*\sharp\varepsilon$$

This example shows that our method may give a good over-approximation of a non-regular reachability set.

A protocol with nested loops is depicted in Fig. 5, which is an abstraction of systems exchanging frames composed of several packets.

The sender first sends a start message, then sends any number of a messages and ends the frame with an end message. The receiver can read any message at any time.

Our analysis shows that, when the sender is in location 0, the content of the queue is :

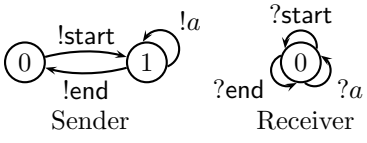


Fig. 5. Nested loop

$$L_0 = \varepsilon + (s + \varepsilon)a^*e(sa^*e)^*$$

Here the ability of representing regular expressions with nested Kleene closures is important; in this case we even obtain the exact reachability set.

Comparison with acceleration techniques. In Tab. 2 we compare the techniques mentioned in Tab. 1 with our non-relational and relational analysis, on the 4 previous examples. We did not consider the method of [8], which assumes lossy channels.

- *yes* means that the reachability analysis gives the exact result.
- *no* means that the reachability analysis does not terminate.
- *approx* means that the reachability analysis gives an over-approximation of the reachability set.

The only case where our relational method gives less satisfactory result than another method, which is also the only case where the result is not exact, is the Non-Regular protocol. On this protocol, the CQDD method can compute the exact reachability set $\bigcup_{n\geq 0} a^n\sharp\epsilon\sharp c^n\sharp\epsilon$, whereas we approximate it, using ∇_k, by $\bigcup_{0\leq n\leq k+2} a^n\sharp\epsilon\sharp c^n\sharp\epsilon \ \cup \ a^{k+2}a^*\sharp\epsilon\sharp c^{k+2}c^*\sharp\epsilon$, which is not so bad. On the other hand, none of the other methods delivers results for all the examples.

This comparison is experimental, and should be completed in the future with larger examples. However, it is very difficult to *prove* the superiority of an analysis that uses a widening operator, as pointed out by [33]. From a theoretical point of view, we can make two statements. First, we can partially order the *expressiveness of the representations* (which does not necessarily induce a corresponding

Table 2. Comparison of acceleration techniques with our 2 analysis

Example	Acceleration techniques			Regular languages with widening	
	SLRE [11]	QDD [12]	CQDD [13]	non-relational	relational
(1) ABP protocol	yes	yes	yes	yes	yes
(2) Conn./deconn.	approx[a]	yes	yes	approx[a]	yes
(3) Non-regular	no[a,b,c]	no[b,c]	yes	approx[a,c]	approx[c]
(4) Nested loops	no[c]	yes	no[c]	yes	yes

[a] non-relational representation [b] counting loops [12] that cannot be accelerated
[c] exact set not representable

ordering of the analyses in terms of accuracy). Following Tab. 1, we have that SLRE is the less expressive, QDD and our relational method are equivalent, and are uncomparable to CQDD. Second, proposition 2 implies a (modest) partial completeness result: if in a CFSM the length of the FIFO queues is bounded by l, then taking $k \geq n \cdot l$ for the widening ∇_k lead to exact results.

7 Conclusion

In this paper, we showed how to perform reachability analysis of CFSMs using an Abstract Interpretation approach and the notion of relational/non-relational analysis [29]. Our method can be applied to any CFSM and always terminates. It is technically simple, based on standard Abstract Interpretation technique and well-known concepts like regular languages and bisimulation of depth k. Despite of its simplicity, that we see as a strength, our method is often as accurate as acceleration techniques on standard examples, and it can deal with counting loops [12]. It is however unable to certify by itself whether the obtained result is exact or not (which is a limitation common to abstract interpretation techniques). Last but not least, we think that our approach is more amenable to the combination of FIFO channels with other unbounded datatypes, like counters, in the spirit of [34]. Indeed, it seems very difficult to accelerate loops where FIFO operations are guarded by numerical tests on counters and where counters are conversely updated depending on the FIFO queues contents.

For CFSMs, our method is a good alternative to acceleration based techniques. The two approaches may actually be seen as complementary. Typically, one can first try to get the exact reachability set using acceleration techniques and then apply our method in case of failure. A more interesting combination consists in using acceleration techniques to add meta-transitions in the original model, when possible, and to apply our method to the augmented system.

In the future we plan to explore two directions: the first one is to combine the abstraction for FIFO queues with abstractions for numerical variables, in order to attack the verification of more realistic models. The second one is to consider CFSM with infinite alphabets. This is required for the many protocols that use "tokens" to uniquely identify different frames. These tokens are typically assumed to belong to an infinite set.

Acknowledgments

We thank the anonymous AMAST referees for their careful comments and suggestions.

References

1. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM **30** (1983) 323–342
2. Bochmann, G.V.: Finite state description of communication protocols. IEEE Computer Society Press, Los Alamitos, CA, USA (1995)
3. Turner, K.J.: Using Formal Description Techniques: An Introduction to Estelle, Lotos, and SDL. John Wiley & Sons, Inc., New York, NY, USA (1993)
4. Nielsen, M., Plotkin, G., Winskel, G.: Petri nets, event structures and domains, part 1. Theoretical Computer Science **13** (1981)
5. ITU-TS: ITU-TS Recommendation Z.120: Message Sequence Chart (MSC). (1999)
6. Reniers, M., Mauw, S.: High-level message sequence charts. In Cavalli, A., Sarma, A., eds.: Proc. of the 8^{th} SDL Forum. (1997)
7. Cécé, G., Finkel, A., Iyer, S.P.: Unreliable channels are easier to verify than perfect channels. Information and Computation **124** (1996) 20–31
8. Abdulla, P., Bouajjani, A., Jonsson, B.: On-the-fly analysis of systems with unbounded, lossy FIFO channels. In: Computer Aided Verification (CAV '98). Volume 1427 of LNCS. (1998)
9. Bardin, S., Finkel, A., Leroux, J., Petrucci, L.: FAST: Fast Acceleration of Symbolic Transition systems. In: Computer Aided Verification (CAV'03). Volume 2725 of LNCS. (2003)
10. Boigelot, B., Godefroid, P., Willems, B., Wolper, P.: The power of QDDs. In: Static Analysis Symposium (SAS'97). Volume 1302 of LNCS. (1997)
11. Finkel, A., Iyer, S.P., Sutre, G.: Well-abstracted transition systems: application to FIFO automata. Information and Computation **181** (2003) 1–31
12. Boigelot, B., Godefroid, P.: Symbolic verification of communication protocols with infinite state spaces using QDDs. FMSD **14** (1997) 237–255
13. Bouajjani, A., Habermehl, P.: Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. Theor. Comp. Science **221** (1999)
14. Abdulla, P., Jonsson, B., Nilsson, M., Saksena, M.: A survey of regular model checking. In: CONCUR'04. Volume 3170 of LNCS. (2004)
15. Bouajjani, A., Jonsson, B., Nilsson, M., Touili, T.: Regular model checking. In: Computer Aided Verification (CAV'00). Volume 1855 of LNCS. (2000)
16. Boigelot, B., Legay, A., Wolper, P.: Iterating transducers in the large. In: Computer Aided Verification (CAV'03). Volume 2725 of LNCS. (2003)
17. Bouajjani, A., Habermehl, P., Vojnar, T.: Abstract regular model checking. In: Computer Aided Verification (CAV'04). Volume 3114 of LNCS. (2004)
18. Memmi, G., Finkel, A.: An introduction to FIFO nets-monogeneous nets: a subclass of FIFO nets. Theoretical Computer Science **31** (1985)
19. Finkel, A., Rosier, L.: A survey on the decidability questions for classes of FIFO nets. In: Eur. Workshop on Applications and Theory of Petri Nets. Volume 340 of LNCS. (1987)
20. Cécé, G., Finkel, A.: Verification of programs with half-duplex communication. Information and Computation **202** (2005)

21. Abdulla, P., Jonsson, B.: Verifying programs with unreliable channels. Information and Computation **127** (1996)
22. Abdulla, P., Bertrand, N., Rabinovich, A., Schnoebelen, P.: Verification of probabilistic systems with faulty communication. Inf. and Comp. **202** (2005)
23. Peng, W., Puroshothaman, S.: Data flow analysis of communicating finite state machines. ACM Trans. Program. Lang. Syst. **13** (1991) 399–442
24. Lesens, D., Halbwachs, N., Raymond, P.: Automatic verification of parameterized linear networks of processes. In: Principles of Programming Languages (POPL'97), ACM Press (1997)
25. Cousot, P., Cousot, R.: Abstract interpretation and application to logic programs. Journal of Logic Programming **13** (1992)
26. Brzozowski, J.A.: Derivatives of regular expressions. Journal of the ACM **1** (1964)
27. Feret, J.: Abstract interpretation-based static analysis of mobile ambients. In: Static Analysis Symposium (SAS'01). Volume 2126 of LNCS. (2001)
28. Jeannet, B., Jeron, T., Le Gall, T.: Abstracting interpretation of FIFO channels. Technical Report 5784, INRIA (2005)
29. Jones, N., Muchnick, S.: Complexity of flow analysis, inductive assertion synthesis, and a language due to Dijkstra. In Jones, N., Muchnick, S., eds.: Program Flow Analysis: Theory and Applications. Prentice-Hall (1981)
30. Wolper, P., Boigelot, B.: An automata-theoretic approach to Presburger arithmetic constraints. In: Static Analysis Symposium (SAS'95). Volume 983 of LNCS. (1995)
31. Bourdoncle, F.: Efficient chaotic iteration strategies with widenings. In: Int. Conf. on Formal Methods in Progr. and their Applications. Volume 735 of LNCS. (1993)
32. Abdulla, P.A., Annichini, A., Bouajjani, A.: Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In: Tools and Algorithms for Construction and Analysis of Systems (TACAS'99). (1999)
33. Su, Z., Wagner, D.: A class of polynomially solvable range constraints for interval analysis without widenings and narrowings. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04). Volume 2988. (2004)
34. Jeannet, B., Halbwachs, N., Raymond, P.: Dynamic partitioning in analyses of numerical properties. In: Static Analysis Symposium, SAS'99. Volume 1694 of LNCS. (1999)