Analysis of Communicating Automata

Anca Muscholl

LaBRI, University Bordeaux, France

Abstract. This extended abstract is a survey of some of the recent developments in the area of automated verification dedicated to the analysis of communicating automata.

Communicating automata are a fundamental computational model for concurrent systems, where processes cooperate via asynchronous message passing using unbounded channels. They are a popular model for representing and reasoning about communication protocols, and they have been used to define the semantics of standardized specification languages such as SDL. However, from the algorithmic point of view communicating automata are more challenging than other true concurrent models such as Petri nets: indeed, this model is Turing equivalent, in particular it subsumes Post tag systems [20]. Therefore, basic questions arising in formal verification, such as the reachability problem, are intractable.

Solving the reachability problem is actually the first step in tackling the more general *model-checking* problem, that consists in verifying that the model, i.e. the communicating automaton, satisfies a given property, usually described in some logical formalisms such as e.g. temporal logics [17]. In this setting, reachability is used for validating safety properties, stating that no bad state can be reached from the initial state. A more challenging and difficult problem is *synthesis*: here, the aim is to compute a model from a given specification. In this survey we will only report on the *closed synthesis* problem, where the model that is to be computed does not interact with any environment. In contrast, synthesis of *open systems*, i.e. systems that are embedded in an unpredictable environment, is even more intricate. The reason why synthesis is challenging here is that communicating automata are a concurrent model, thus the simplest instance of synthesis (i.e., the closed case) already amounts to compute the distribution of a sequential object, i.e. the specification. In the open case, the situation is even more challenging, since we need to solve some sort of concurrent games. In both cases, the existing techniques are rather sparse.

Starting with the model-checking problem, an important line of research was devoted to identify structural or behavioral restrictions on communicating automata that make them amenable to algorithmic methods. A first example are lossy channel systems, or equivalently, communicating automata where any number of messages can be lost, at any time. Lossy channel systems are a particular instance of well-structured transition systems [1, 9] , so reachability was shown to be decidable [2], albeit of non-primitive recursive complexity [21]. On the other hand, liveness properties were shown to be undecidable [1].

A second line of research aimed at describing the set of reachable configurations of communicating automata, resp. approximations thereof, by some form of extended finite-state automata (called *symbolic representations*). The idea here is to manipulate a possibly infinite set of configurations by means of finite objects, such as finite automata or some suitable extensions.

Whereas both previous approaches emphasize the symbolic representation of the set of reachable configurations, an orthogonal approach based on *partial orders*, has been developed more recently. The partial order approach emphasizes concurrency and, in particular, the partially ordered events executed by a communicating automaton. Here, we are mainly interested e.g. in the reachability of control states, so that the memory (i.e., the channel contents), is handled only implicitly. Notice that this kind of event-based reasoning arises very naturally when communicating automata are viewed as sequential automata synchronizing over communication events.

The partial order approach was successfully applied for obtaining both model-checking algorithms, as well as synthesis algorithms, for so-called *existentially-bounded* finite state communicating automata [13]. The main idea here is to assume unbounded channels, but to consider only executions that can be rescheduled on (uniformly) bounded ones. A simple example illustrating the idea is a pair of processes, a producer and a consumer, where the producer keeps sending messages to the consumer. Since there is no control on the relative speed of these two processes, the channel should be of unlimited size. However, often it suffices to reason on executions where messages can be consumed without delay, i.e. on executions that can be scheduled with a channel of size one.

The partial order approach has been actually inspired by the study of automata and logics over Mazurkiewicz traces ([19], see also [8] for a textbook of the topic). The deepest result in the area of Mazurkiewicz traces is Zielonka's construction of distributed (trace) automata from sequential automata [23, 24]. This sophisticated construction is the building brick for the synthesis of existentially bounded finite-state communicating automata in [10].

1 Basics

Communicating automata follow the simple paradigm of a network of automata cooperating asynchronously over point-to-point, fifo communication channels. They arise naturally as models for peer-to-peer interaction, as occurring e.g. in distributed protocols using asynchronous message passing.

We consider systems described by means of a fixed *communication network*, consisting of a (usually finite) set of concurrent *processes* $\mathcal{P}$, together with a set of *channels* $\mathrm{Ch} \subseteq \{(p, q) \in \mathcal{P}^2 \mid p \neq q\}$, that stand for point-to-point links. Following the classical definition [6], we exclude multiple channels between a pair of processes, as well as self-linking channels. However, this has no severe impact on the kind of results we will present. At best, it has an impact when one aims at classifying networks w.r.t. decidability of various verification questions. In this model, processes act either by point-to-point communication or by local

actions. A send action denoted as $p!q(m)$ means that process p sends a message with content m to process q. A receive action denoted as $p?q(m)$ means that p receives from q a message with content m. Whenever we write $p!q$ and $q?p$, we will assume that $(p,q) \in \mathrm{Ch}$. A local action m on process p is denoted as $\ell_p(m)$. For a given (finite) set M of message contents, resp. local actions, and a process $p \in \mathcal{P}$, we define the set of p-local actions as $\Sigma_p = \{p!q(m), p?q(m), \ell_p(m) \mid m \in M\}$ and set $\Sigma = \bigcup_{p\in\mathcal{P}} \Sigma_p$.

A *communicating automaton* (*CA* for short) is a tuple $\mathcal{A} = \langle (\mathcal{A}_p)_{p\in\mathcal{P}}, F\rangle$ where

- each $\mathcal{A}_p = (S_p, \to_p, s_p^0)$ is a labeled transition system (LTS for short) with state space S_p, transition relation $\to_p \subseteq S_p \times \Sigma_p \times S_p$, and $s_p^0 \in S_p$ as initial state;
- $F \subseteq \prod_{p\in\mathcal{P}} S_p$ is a set of *global* final states.

We denote the product $S := \prod_{p\in\mathcal{P}} S_p$ as set of *global states*.

The behavior of a CA is defined as the behavior of an infinite-state LTS, by considering the possible (local) transitions on the set of configurations of the CA. A *configuration* of the CA $\mathcal{A}$ consists a global state $s \in S$, together with a word $w_{p,q} \in M^*$ for each channel $(p,q) \in \mathrm{Ch}$. We write $C = \langle s = (s_p)_{p\in\mathcal{P}}, w = (w_{p,q})_{(p,q)\in\mathrm{Ch}}\rangle$ for a configuration with global state $s \in S$ and channel contents $w \in (M^*)^{\mathrm{Ch}}$. The set of all configurations of $\mathcal{A}$ is denoted $\mathcal{C}_\mathcal{A}$ (or simply $\mathcal{C}$ when there is no risk of confusion). For any two configurations $C = \langle s, w\rangle, C' = \langle s', w'\rangle$ and any action $a \in \Sigma_p$ of $\mathcal{A}$, we say that C' is a *successor* of C (and write $C \xrightarrow{a} C'$, or simply $C \longrightarrow C'$ or $C' \in \mathrm{post}(C)$, when the action does not matter), if

- $s_p \xrightarrow{a}_p s'_p$ is a p-local transition, and $s'_q = s_q$ for all $q \neq p$,
- Send action: if $a = p!q(m)$, then $w'_{p,q} = w_{p,q}m$ (message m is inserted into the channel from p to q) and $w'_{r,s} = w_{r,s}$ for all $(r,s) \neq (p,q)$ (all other channels are unchanged).
- Receive action: if $a = p?q(m)$, then $w_{q,p} = mw'_{q,p}$ (message m is deleted from the channel from q to p) and $w'_{r,s} = w_{r,s}$ for all $(r,s) \neq (q,p)$ (all other channels are unchanged).
- Local action: if $a = \ell_m$, then $w = w'$.

A *run* of a CA $\mathcal{A}$ is (finite or infinite) sequence of transitions: $\rho = C_0 \xrightarrow{a_0} C_1 \xrightarrow{a_1} C_2 \cdots$, with $C_i \in \mathcal{C}_\mathcal{A}$ configurations and $a_i \in \Sigma$ actions. For a run ρ as above, we also write $C_0 \xrightarrow{*} C_n$.

We define *accepting* runs in the usual way, by referring to the global states. A finite run $\rho = C_0 \xrightarrow{a_0} C_1 \xrightarrow{a_1} \cdots C_n$ is accepting if $C_0 = \langle s^0, \boldsymbol{\varepsilon}\rangle$ and $C_n = \langle f, w\rangle$, where $\boldsymbol{\varepsilon}_{p,q} = \varepsilon$ for all $(p,q) \in \mathrm{Ch}$, $f \in F$ and $w \in (M^*)^{\mathrm{Ch}}$.

The *reachability set* of a CA $\mathcal{A}$, denoted $\mathrm{Reach}(\mathcal{A})$, is the set

$$\mathrm{Reach}(\mathcal{A}) := \{w \in (M^*)^{\mathrm{Ch}} \mid C_0 \xrightarrow{*} \langle f, w\rangle \text{ for some } f \in F\}\,.$$

The *language* of a CA $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$, is the set

$$\mathcal{L}(\mathcal{A}) = \{a_0 a_1 \cdots a_{n-1} \mid C_0 \xrightarrow{a_0} C_1 \xrightarrow{a_1} C_2 \cdots \xrightarrow{a_{n-1}} C_n \text{ is an accepting run}\}.$$

Remark 1. Notice that we did not impose in the definition of a communicating automaton $\mathcal{A} = \langle (A_p)_{p\in\mathcal{P}}, F\rangle$ any restriction on the local LTS $\mathcal{A}_p$. In general, we might be interested in various kinds of (possibly infinite-state) LTS, such as pushdown automata. However, a basic kind of CA is obtained by requiring that every $\mathcal{A}_p$ is a finite-state automaton, and then we denote $\mathcal{A}$ as *communicating finite-state machine* (CFM for short). Most of the research done in the past 15 years on CAs focused on CFMs, and we will concentrate on them in the next sections.

2 Symbolic representations

The basic idea when using symbolic representations is to approximate in a finitary way behavior of a CFM. Often, such a computation is intended to capture the effect of iterating single loops. This leads to define *meta-transitions* and to use them to accelerate the usual fixpoint computation defining $\text{Reach}_{\mathcal{A}}$:

$$X := \{C_0\}, \quad X := X \cup \text{post}(X)$$

Queue-content Decision Diagrams (QDD for short) were proposed in [3] for describing (possibly infinite) sets of configurations of CFM by finite automata. With such representations, one can answer to various questions such as boundedness of channels or reachability. But of course, the method only offers a semi-algorithm for the reachability problem.

Let us assume that our network has k channels, with a fixed order on Ch. The channel contents $w = (w_i)_{i=1}^k$ of a configuration $\langle s, w\rangle$ of the CFM can be represented by a word $w_1\#w_2\#\cdots w_k\#$ (assuming that $\# \notin M$). A QDD is then a finite automaton reading words from $(M^*\#)^k$. Computing the effect of a loop means iterating a sequence $\sigma \in S^*$, that leads from a global state $s \in S$ back to s. The general idea is to start with $s, \sigma, \mathcal{B}$, where $\mathcal{B}$ is a QDD, and to compute the effect of σ^* on the set of configurations $\{\langle s, (w_i)_{i=1}^k\rangle \mid w_1\#w_2\#\cdots w_k\# \in \mathcal{L}(\mathcal{B})\}$. The paper [4] characterizes those sequences σ that preserve regularity, i.e., QDD representations, as *non-counting* sequences. This roughly means that such loops cannot send on two different channels. This paper also suggests a semi-algorithm for model-checking LTL properties on runs of CFM, where atomic propositions refer to control states.

The paper [5] goes beyond regular representations, introducing *Constrained Queue-content Decision Diagrams* (CQDD for short). CQDDs are restricted finite automata, extended by linear constraints on the frequency of transitions in a run. The main result of [5] is that the CQDD representation is preserved by the iteration of arbitrary loops.

3 Faulty channels

Assuming that channels are imperfect, at least two types of faults may seem natural for real systems. *Lossy* machines are such that channels can loose an arbitrary number of messages, at any time. For machines with *insertion errors*, new messages can be inserted in channels, at any time. Although these two models have different flavor, the techniques used to manipulate them are quite similar, so that we will only consider lossy CFMs in the following.

Lossy CFMs (or *lossy channel systems*) represent a special instance of a more general class of infinite-state systems, known as *well-structured transition systems* (WSTS for short), [2, 9]. The basic idea behind a WSTS $\langle \mathcal{S}, \longrightarrow \rangle$ with state space $\mathcal{S}$ is to use a *well quasi-order* (wqo for short) on $\mathcal{S}$ in order to manipulate certain infinite subsets of $\mathcal{S}$ symbolically. A wqo $\preceq$ on $\mathcal{S}$ is a well-founded preorder with no infinite anti-chain. What makes a transition system $\langle \mathcal{S}, \longrightarrow \rangle$ a WSTS is *monotonicity*: for every $s' \in \mathrm{post}(s)$ and every $s_1 \in \mathcal{S}$ with $s \preceq s_1$, it is required that some $s'_1 \in \mathrm{post}(s_1)$ exists such that $s' \preceq s'_1$.

Two basic properties are crucial for WSTS. The first one is that every *upward-closed*[1] subset $X \subseteq \mathcal{S}$ can be described by a *finite* set of minimal elements. The second property is that the predecessor relation preserves upward-closed sets. That is, $\mathrm{pre}(X) := \{x \mid x \longrightarrow y \text{ for some } y \in X\}$ is upward-closed whenever X is upward-closed. As a consequence, reachability of upward-closed sets X can be decided by a backward algorithm, that computes in a fixpoint manner $\mathrm{pre}^*(X)$. Intersecting the result with the set of initial configurations solves the reachability problem.

For lossy CFMs, the choice for a wqo is very natural. One starts with the subword ordering: for two words $x, y \in M^*$, let $x \preceq y$ if $x = x_1 \cdots x_n$ and $y = y_0 x_1 y_1 \cdots y_{n-1} x_n y_n$ for some $y_i \in M^*$. This wqo easily extends to M^k and then to configurations of the CFM. For two configurations $C = \langle s, w \rangle$, $C' = \langle s', w' \rangle$ we set $C \preceq C'$ if $s = s'$ and $w \preceq w'$.

This technique allows to decide e.g. control-state reachability for lossy CFMs. More complex properties, such as repeated reachability of a control state, are undecidable [2]. For deciding termination from an initial configuration, a different technique is employed, based on the computation of a finite reachability tree (forward algorithm). However, the more general problem of termination from *any* initial configuration, is undecidable [18]. From a different perspective, a recent paper [7] considered mixed architectures, where some channels are lossy and others are error-free, and characterized architectures with a decidable reachability question.

4 Partial order approach

An early line of work considered *universally bounded* CFMs. This property amounts to say that there exists a uniform bound B such that every run can be

[1] X is upward-closed if $X = \{y \mid x \preceq y \text{ for some } x \in X\}$.

executed with channels of size B, no matter how events are scheduled. Equivalently, the number of transitory messages is at most B, at any time. Since the size of the communication channels is fixed uniformly, this constraint turns a CFM into a finite state device. Checking that a CFM is universally bounded is undecidable, in general. However if the bound B is given, and if the CFM $\mathcal{A}$ is deadlock-free, then we can check in polynomial space whether $\mathcal{A}$ is universally B-bounded [11].

Being universally bounded leads immediately to a decision procedure for the model-checking problem, since we can transform the CFM into an (exponentially larger) finite automaton.

An even more important result concerns closed synthesis. Suppose that we are given a regular language $L \subseteq \Sigma^*$, that satisfies the following properties for some $B > 0$ (notice that these properties are decidable for a given B):

1. For every prefix w of a word from L, and every $(p, q) \in \mathrm{Ch}$, we have $|w|_{p!q} - |w|_{q?p} \leq B$.
2. Every word in L can induce a fifo run, which leads to a configuration where all channels are empty.
3. Whenever $w \in L$, we can swap adjacent actions in w that (1) do not belong to the same process and (2) do not form a matching send/receive pair, and the resulting word is still in L.

The main result of [14], later extended to infinite runs in [16], is a construction for transforming a regular language satisfying the three properties above into a universally B-bounded CFM. As mentioned previously, the challenge for such constructions is to distribute a sequential object, e.g. the finite automaton describing L. The techniques make heavy use of Mazurkiewicz trace theory and, in particular, of Zielonka's construction for distributed automata.

The drawback of models with universally bounded channels is the limited expressive power. Intuitively, universal channel bounds require message acknowledgments, which can be difficult to impose in general. For instance, basic protocols of producer-consumer type (such as e.g. the USB protocol) are not universally bounded, since the communication is one-way and does not allow acknowledgments. Therefore, a relaxation of this restriction on channels was proposed in [13, 10]. The idea is to require an *existential bound* on channels. This means roughly that every CFM run must have *some* scheduling of events that respects the given channel bound (other schedules might exceed the bound). In other words, runs can be executed with bounded channels, provided that we schedule the events fairly. For instance, in a producer-consumer setting, the scheduling alternates between producer and consumer actions. This requirement is perfectly legitimate in practice, since real life protocols must be executable with limited communication channels. When a channel overflow happens, then the sender stops temporarily until some message is consumed from the queue.

For channel systems with existential bounds, the fundamental Kleene-Büchi equivalence of automata, logics and regular expressions was shown to hold in [10]. Automata means here CFMs, whereas logics refers to monadic second-order logics over partial orders. Regular expressions refer to a visual formalism

that is very popular for early design of communication protocols, namely *message sequence charts*. Regarding model-checking, the complexity for existentially-bounded CFMs remains the same as in the case of universal bounds [13].

5 Conclusion and outlook

This survey focused on two basic questions related to the automated verification of communicating automata, namely *model-checking* and *(closed) synthesis*. We presented three different approaches that allow to tackle these questions, usually offering some approximation of the behavior of this type of automata. We did not mention results specific to the ITU standard of *message sequence charts*, see [12] for some further references on this topic.

Concerning further work on communicating automata, let us mention some recent development. The extension of communicating automata by local push-downs received recently attention justified by questions on the analysis of multithreaded programs or distributed software. Of course, the model is highly undecidable but still, methods like context-bounded model-checking or suitable restrictions on the network provide reasonable practical settings where reachability is decidable, [22, 15].

References

1. P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General decidability theorems for infinite state systems. In *LICS'96*, pages 313–323, 1996.
2. P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.
3. B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In *CAV'96*, volume 1102 of *LNCS*, pages 1–12. Springer, 1996.
4. B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs. In *SAS'97*, volume 1302 of *LNCS*. Springer, 1997.
5. A. Bouajjani and P. Habermehl. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. *Theor. Comp. Science*, 221(1-2):211–250, 1999.
6. D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
7. P. Chambart and P. Schnoebelen. Mixing lossy and perfect fifo channels. In *CONCUR'08*, volume 5201 of *LNCS*, pages 340–355. Springer, 2008.
8. V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.
9. A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.
10. B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. Comput.*, 204(6):920–956, 2006.
11. B. Genest, D. Kuske, and A. Muscholl. On communicating automata with bounded channels. *Fundam. Inform.*, 80(1-3):147–167, 2007.

12. B. Genest, A. Muscholl, and D. Peled. Message sequence charts. In *Lectures on Concurrency and Petri Nets*, number 3098 in LNCS, pages 537–558. Springer, 2003.
13. B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state high-level MSCs: Model-checking and realizability. *J. Comput. Syst. Sci.*, 72(4):617–647, 2006.
14. J. G. Henriksen, M. Mukund, K. N. Kumar, M. Sohoni, and P. Thiagarajan. A theory of regular MSC languages. *Information and Computation*, 202(1):1–38, 2005.
15. A. Heußner, J. Leroux, A. Muscholl, and G. Sutre. Reachability analysis of communicating pushdown systems. In *FoSSaCS'10*, LNCS. Springer, 2010.
16. D. Kuske. Regular sets of infinite message sequence charts. *Information and Computation*, 187:80–109, 2003.
17. Z. Manna and A. Pnueli. Verification of the concurrent programs: the temporal framework. In R.Boyer and J.Moore, editors, *The Correctness Problem in Computer Scince*, pages 215–273. Academic Press, 1981.
18. R. Mayr. Undecidable problems in unreliable computations. *Theor. Comp. Science*, 297(1-3):337–354, 2003.
19. A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.
20. E. Post. Formal reductions of the combinatorial decision problem. *American Journal of Mathematics*, 65(2):197–215, 1943.
21. P. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Inf. Process. Lett.*, 83(5):251–261, 2002.
22. S. L. Torre, P. Madhusudan, and G. Parlato. Context-bounded analysis of concurrent queue systems. In *TACAS'08*, volume 4963 of *LNCS*, pages 299–314. Springer, 2008.
23. W. Zielonka. Notes on finite asynchronous automata. *RAIRO - Informatique Théorique et Applications*, 21:99–135, 1987.
24. W. Zielonka. Asynchronous automata. In G. Rozenberg and V. Diekert, editors, *Book of Traces*, pages 175–217. World Scientific, Singapore, 1995.