Systems of of Communicating Finite State Machines and related properties

1. Introduction

The material of this section is taken from Chapter 1 of the PhD Thesis "Improving state exploration techniques for the automatic verification of concurrent systems", University of Ottawa, by Hans van der Schoot.

Sophisticated computer and information systems have become of the essence in todays society,and they are being deployed at an ever increasing rate. Most contemporary computing systems are typically composed of entities that operate concurrently and cooperate through communication. Examples of such concurrent systemsare computer and communication networks and protocols,operating systems, asynchronous circuits and many other embedded systems with application areas like process control, telephony and air traffic control to just name a few.

The correct design of concurrent systems is known to be a problem of considerable depth. One major source of difficulties lies in the fact that the functionality of these systems tends to be very large and complex. Traditionally, a sequential system (or program) is transformational and can bethought of as a function: given an input, it may produce an output. The specification of all input-output pairs defines the precise meaning of the system. A concurrent system is yet hard to describein this way as it operates within an environment over an indefinite period of time. Its functional behavior is defined by the many ongoing interactions with its environment, and these interactions often exhibit complex interdependencies. For this reason, it is difficult to adequately specify, understand and predict the behavior of concurrent systems and, hence, to assess whether they meettheir requirements.

Another source of difficulties lies in the distributed nature of concurrent systems. As the constituent entities of a concurrent system are dispersed over different locations, they must also interact with each other in order to realize the functionality of the system as a whole. An important example is found in communication protocols, where protocol entities interact according to strictrules. Indeed, the mere purpose of a communication protocol is to govern the orderly exchange of messages among communicating entities. Both the design of communication protocols and the assessment of their correctness are certainly delicate tasks, and rigorous automated analysis methods are required to support these tasks.

Verification of concurrent systems, and of communication protocols in particular corresponds to the act of proving (or disproving) formally that a system design meets its expected properties, which can range from several types of general consistency requirements to more specific functional requirements asserted in, for instance, a logical language. What is strictly not meant is testing (unless it is exhaustive), or any other method which may indicate that a system design is probably correct. In order to prove that a system satisfies some property, all possible executionsof the system must be checked to determine whether each and every one of them complies to the property. As such, verification is thus the means to guarantee the correctness of the design of a concurrent system or communication protocols.

Throughout the past years or so, various formal models have been proposed and studied to facilitate the specification and validation of (designs of) concurrent systems. These models differ in their expressiveness in terms of specification and in their tractability in terms of validation (i.e.verification and testing). However, most of the models have in common that their individual semantics renders a translation of the pure syntactic description of a concurrent system into some kind of a transition system, consisting of a set of states, a designated initial state, and a (labeled) transition relation among these states. This transition system represents the behavior of the concurrent system as a whole, i.e. the joint behavior of all the concurrent entities in the system.

A model particularly suited for specifying communication protocols is the communicating finite state machine(CFSM) model. In the CFSM model, a protocol is specified as a collection of processes (i.e. the protocol entities) that exchange messages over error-free simplex channels. Each process is modeled as a finite state machine (FSM) and each simplex channel is a FIFO queue. A (global) state of the protocol consists of a state for each FSM and a content for each simplex channel. A state transition can occur only when some process is ready to either send a message to one of its output channels, or receive a message from one of its input channels. The CFSM model is well-defined, elegant and rather easy to understand. These features make it attractive for both academia and industry. Indeed, the CFSM model has become a widely established means for specifying, verifying and testing communication protocols. Furthermore, it underlies two standardized specification languages, namely Estelle and SDL.

One of the most prevalent techniques for the verification of protocols, and concurrent systems in general, is state space exploration. State (space) exploration, which is widely known also as reachability analysis, amounts to exploring in a systematic manner the complete state space of asystem, i.e. all states and transitions of the system that can be reached from a given initial state. Many different types of system properties can be verified by reachability analysis. It was originally proposed for verifying so-called logical correctness properties of protocols specified in the CFSMmodel, namely freedom of deadlocks, non-executable transitions (cf. dead code in a computerprogram), unspecified receptions, and buffer overflows or unbounded channel growth. These are general correctness properties that concern concurrent systems at large, albeit that unspecified receptions and buffer overflows or unbounded channel growth are particular to models featuring some form of asynchronous message passing.

Reachability analysis can further be employed for the verification of individual, functional correctness properties of concurrent systems and protocols, like temporal safety and liveness properties. This has emanated in the past decade from the development of model-checking methods for various temporal logics.

2. Systems of Communicating Finite State Machines

In this section we recall (partly following [1, 2, 3]) the definitions of communicating finite state machine (CFSM) and systems of CFSMs. Throughout the section we assume given a countably infinite set $\mathbf{P}_{\mathfrak{U}}$ of role (participant) names (ranged over by $\mathtt{p}, \mathtt{q}, \mathtt{r}, \mathtt{s}, \mathtt{A}, \mathtt{B}, \mathtt{H}, \mathtt{I}, \ldots$) and a countably infinite alphabet $\mathbb{A}_{\mathfrak{U}}$ (ranged over by $a, b, c, \ldots$) of messages.

Definition 2.1 (CFSM). *Let $\mathbf{P}$ and $\mathbb{A}$ be finite subsets of $\mathbf{P}_{\mathfrak{U}}$ and $\mathbb{A}_{\mathfrak{U}}$ respectively.*

i) The set $C_{\mathbf{P}}$ of channels *over $\mathbf{P}$ is defined by*
$$C_{\mathbf{P}} = \{\mathtt{pq} \mid \mathtt{p}, \mathtt{q} \in \mathbf{P}, \mathtt{p} \neq \mathtt{q}\}$$

ii) The set $Act_{\mathbf{P},\mathbb{A}}$ of actions *over $\mathbf{P}$ and $\mathbb{A}$ is defined by*
$$Act_{\mathbf{P},\mathbb{A}} = C_{\mathbf{P}} \times \{!, ?\} \times \mathbb{A}$$

iii) A $\mathtt{p}$-communicating finite-state machine over $\mathbf{P}$ and $\mathbb{A}$, *where $\mathtt{p} \in \mathbf{P}$, is a finite transition system given by a tuple*
$$M = (Q, q_0, \mathbb{A}, \delta)$$
where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\delta \subseteq Q \times Act_{\mathbf{P},\mathbb{A}} \times Q$ is a set of transitions such that
$$(q, l, q') \in \delta \implies sbj(l) = \mathtt{p}$$
where $sbj : Act_{\mathbf{P},\mathbb{A}} \longrightarrow \mathbf{P}$ is defined by $\quad sbj(\mathtt{rs}!a) = \mathtt{r} \qquad sbj(\mathtt{rs}?a) = \mathtt{s}$

Notice that the above definition of a CFSM is generic w.r.t. the underlying sets $\mathbf{P}$ of roles and $\mathbb{A}$ of messages. This is actually necessary only when we deal with an arbitrary number of open systems that can be *composed.* We shall write C and Act instead of $C_{\mathbf{P}}$ and $Act_{\mathbf{P},\mathbb{A}}$ when no ambiguity can arise. We assume $l, l', \ldots$ to range over Act; $\varphi, \varphi', \ldots$ to range over Act^* (the set of finite words over Act), and $w, w', \ldots$ to range over $\mathbb{A}^*$ (the set of finite words over $\mathbb{A}$). $\varepsilon\,(\notin \mathbb{A} \cup Act)$ denotes the empty word and $\mid v \mid$ the lenght of a word $v \in Act^* \cup \mathbb{A}^*$. Given a word v with prefix v', i.e. such that $v = v' \cdot v''$ for a certain v'', we define $v \setminus v' = v''$. Moreover, given a word v with 'a' as last element, i.e. $v = v' \cdot a$ for a certain v', we define $\mathsf{init}(v) = v'$ and $\mathsf{last}(v) = a$.

The transitions of a CFSM are labelled by actions; a label $\mathtt{sr}!a$ represents the asynchronous sending of message a from machine $\mathtt{s}$ to $\mathtt{r}$ through channel $\mathtt{sr}$

and, dually, $\texttt{sr}?a$ represents the reception (consumption) of a by $\texttt{r}$ from channel $\texttt{sr}$.

We write $\mathcal{L}(M) \subseteq Act^*$ for the language over Act accepted by the automaton corresponding to machine M, where each state of M is an accepting state. A state $q \in Q$ with no outgoing transition is *final*; q is a *sending* (resp. *receiving*) state if it is not final and all outgoing transitions are labelled with sending (resp. receiving) actions; q is a *mixed* state if there are at least two outgoing transitions: one labelled with a sending action and the other one labelled with a receiving action.

A CFSM $M = (Q, q_0, \mathbb{A}, \delta)$ is:

a) *deterministic* if for all states $q \in Q$ and all actions l: $(q, l, q'), (q, l, q'') \in \delta$ imply $q' = q''$;

b) *?-deterministic* (resp. *!-deterministic*) if for all states $q \in Q$ and all actions $(q, \texttt{rs}?a, q'), (q, \texttt{pq}?a, q'') \in \delta$ (resp. $(q, \texttt{rs}!a, q'), (q, \texttt{pq}!a, q'') \in \delta$) imply $q' = q''$;

c) *?!-deterministic* if it is both ?-deterministic and !-deterministic.

The notion of ?!-deterministic machine is more demanding than in usual CFSM settings. Note that a ?!-deterministic CFSM is also deterministic, but the converse does not hold (since the channel names are abstracted away in the definition of ?!-determinism).

Definition 2.2 (Communicating system and configuration)**.** *Let* $\mathbf{P}$ *and* $\mathbb{A}$ *be as in Def. 2.1.*

i) A communicating system (CS) over $\mathbf{P}$ and $\mathbb{A}$ *is a tuple* $S = (M_{\texttt{p}})_{\texttt{p}\in\mathbf{P}}$ *where for each* $\texttt{p} \in \mathbf{P}$*,* $M_{\texttt{p}} = (Q_{\texttt{p}}, q_{0\texttt{p}}, \mathbb{A}, \delta_{\texttt{p}})$ *is a* $\texttt{p}$*-CFSM over* $\mathbf{P}$ *and* $\mathbb{A}$*.*

ii) A configuration *of a system* S *is a pair* $s = (\vec{q}, \vec{w})$ *where*

- $\vec{q} = (q_{\texttt{p}})_{\texttt{p}\in\mathbf{P}}$ *with* $q_{\texttt{p}} \in Q_{\texttt{p}}$,
- $\vec{w} = (w_{\texttt{pq}})_{\texttt{pq}\in C}$ *with* $w_{\texttt{pq}} \in \mathbb{A}^*$.

The component $\vec{q}$ *is the control state of the system and* $q_{\texttt{p}} \in Q_{\texttt{p}}$ *is the local state of machine* $M_{\texttt{p}}$*. The component* $\vec{w}$ *represents the state of the channels of the system and* $w_{\texttt{pq}} \in \mathbb{A}^*$ *is the state of the channel for messages sent from* $\texttt{p}$ *to* $\texttt{q}$*. The initial configuration of* S *is* $s_0 = (\vec{q_0}, \vec{\varepsilon})$ *with* $\vec{q_0} = (q_{0_{\texttt{p}}})_{\texttt{p}\in\mathbf{P}}$*.*

Definition 2.3 (Reachable configuration)**.** *Let* S *be a communicating system over* $\mathbf{P}$ *and* $\mathbb{A}$*, and let* $s = (\vec{q}, \vec{w})$ *and* $s' = (\vec{q'}, \vec{w'})$ *be two configurations of* S*. Configuration* s' is reachable from s by firing a transition *with action* l*, written* $s \xrightarrow{l} s'$*, if there is* $a \in \mathbb{A}$ *such that one of the following conditions holds:*

1. $l = \texttt{sr}!a$ *and* $(q_{\texttt{s}}, l, q'_{\texttt{s}}) \in \delta_{\texttt{s}}$ *and* *a) for all* $\mathtt{p} \neq \mathtt{s}:\ q'_{\mathtt{p}} = q_{\mathtt{p}}$ *and*

b) $w'_{\mathtt{sr}} = w_{\mathtt{sr}} \cdot a$ *and for all* $\mathtt{pq} \neq \mathtt{sr}:\ w'_{\mathtt{pq}} = w_{\mathtt{pq}}$*;*

2. $l = \mathtt{sr}?a$ *and* $(q_{\mathtt{r}}, l, q'_{\mathtt{r}}) \in \delta_{\mathtt{r}}$ *and*

a) for all $\mathtt{p} \neq \mathtt{r}:\ q'_{\mathtt{p}} = q_{\mathtt{p}}$ *and*

b) $w_{\mathtt{sr}} = a \cdot w'_{\mathtt{sr}}$ *and for all* $\mathtt{pq} \neq \mathtt{sr}:\ w'_{\mathtt{pq}} = w_{\mathtt{pq}}$*.*

We write $s \longrightarrow s'$ *if there exists* l *such that* $s \xrightarrow{l} s'$*. As usual, we denote the reflexive and transitive closure of* $\longrightarrow$ *by* $\longrightarrow^*$*. The set of* reachable configurations *of* S *is* $RS(S) = \{s \mid s_0 \longrightarrow^* s\}$*.*

According to the last definition, communication happens via buffered channels following the FIFO principle.

Definition 2.4 (Communication properties)**.** *Let* S *be a communicating system, and let* $s = (\vec{q}, \vec{w})$ *be a configuration of* S*.*

i) s *is a* deadlock configuration *if*

$$\vec{w} = \vec{\varepsilon} \quad \wedge \quad \forall \mathtt{p} \in \mathbf{P}.\ q_{\mathtt{p}} \textit{ is a receiving state}$$

i.e. all buffers are empty, but all machines are waiting for a message. We say that S *is* deadlock-free *whenever, for any* $s \in RS(S)$*,* s *is not a deadlock configuration.*

ii) s *is an* orphan message configuration *if*

$$(\forall \mathtt{p} \in \mathbf{P}.\ q_{\mathtt{p}} \textit{ is final}) \quad \wedge \quad \vec{w} \neq \vec{\varepsilon}$$

i.e. each machine is in a final state, but there is still at least one non-empty buffer. We say that S *is* orphan message-free *whenever, for any* $s \in RS(S)$*,* s *is not an orphan message configuration.*

iii) s *is an* unspecified reception configuration *if*

a) $\exists \mathtt{r} \in \mathbf{P}.\ q_{\mathtt{r}}$ *is a receiving state ; and*

b) $\forall \mathtt{s} \in \mathbf{P}.[\ (q_{\mathtt{r}}, \mathtt{sr}?a, q'_{\mathtt{r}}) \in \delta_{\mathtt{r}} \implies (|w_{\mathtt{sr}}| > 0 \quad \wedge \quad w_{\mathtt{sr}} \notin \mathbb{A}^* \cdot a)\ ]$.

i.e. there is a receiving state $q_{\mathtt{r}}$ *which is prevented from receiving any message from any of its buffers. (In other words, in each channel* $\mathtt{sr}$ *from which role* $\mathtt{r}$ *could consume there is a message which cannot be received by* $\mathtt{r}$ *in state* $q_{\mathtt{r}}$*.) We say that* S *is* reception error-free *whenever, for any* $s \in RS(S)$*,* s *is not an unspecified reception configuration.*

iv) S *satisfies the* progress property *if for all* $s = (\vec{q}, \vec{w}) \in RS(S)$*, either there exists* s' *such that* $s \longrightarrow s'$ *or* $(\forall \mathtt{p} \in \mathbf{P}.\ q_{\mathtt{p}}$ *is final).*

Note that the progress property iv) implies deadlock-freeness. The other properties are orthogonal to each other.

The above definitions of communication properties are the same as the properties considered in [2], though our formulation of progress is slightly simpler but equivalent to the one in [2]. The notions of orphan message and unspecified reception are also the same in [3]. The same notions of deadlock and unspecified reception are given in [1] and inspired by [4]. The deadlock notions in [4] and [3] coincide with [1] and [2] if the local CFSMs have no final states. Otherwise deadlock in [3] is weaker than deadlock above. A still weaker notion of deadlock configuration, and hence a stronger notion of deadlock-freeness, has been suggested in [5]. To distinguish it from the notion above, we call it *strong deadlock-freeness.*

Definition 2.5 (Strong deadlock-freeness)**.** *Let S be a communicating system, and let $s = (\vec{q}, \vec{w})$ be a configuration of S.*
s is a weak deadlock configuration *if $s \not\longrightarrow$ and either*

a) $\exists \mathtt{r} \in \mathbf{P}$ *such that* $q_{\mathtt{r}} \xrightarrow{\mathtt{rs}?a} q'_{\mathtt{r}}$ *, or*

b) $\vec{w} \neq \vec{\varepsilon}$

i.e. s is stuck and at the same time either a machine is still waiting for a message or there is a message waiting in a buffer which cannot be consumed (or both). We say that S is strongly deadlock-free *whenever, for any $s \in RS(S)$, s is not a weak deadlock configuration.*

As it is natural to expect, a stuck configuration made of final states and empty buffers is not a weak deadlock configuration. But any orphan message configuration is a weak deadlock configuration.

It can be shown that the strong deadlock-freeness property is equivalent to the properties of progress and orphan message-freeness. Moreover, progress is just the same as strong deadlock-freeness when alternative (b) of Definition 2.4(i) is omitted.

Proposition 2.6. *Let S be a communicating system.*

i) If S is strongly deadlock-free, then S is deadlock-free.

ii) There is no progress in a reachable configuration $s = (\vec{q}, \vec{w}) \in RS(S)$ if and only if $s \not\longrightarrow$ and

a) $\exists \mathtt{r} \in \mathbf{P}$ *such that* $q_{\mathtt{r}} \xrightarrow{\mathtt{rs}?a} q'_{\mathtt{r}}$.

iii) S is strongly deadlock-free if and only if S is orphan message-free and satisfies the progress property.

Proof. i) is obvious.
ii) $\Rightarrow$: Assume that there is no progress in s. Then $s \not\longrightarrow$ and there exists $\mathtt{r} \in \mathbf{P}$ such that $\mathtt{r}$ is not final. Thus no sending action is possible in $\mathtt{r}$. Therefore there

must exist a transition $q_{\mathtt{r}} \xrightarrow{\mathtt{rs}?a} q'_{\mathtt{r}}$ and thus a) holds.
$\Leftarrow$: The converse direction is clear, since $s \not\rightarrow$ and, by (a), there exists $\mathtt{r} \in \mathbf{P}$ and $q_{\mathtt{r}} \xrightarrow{\mathtt{rs}?a} q'_{\mathtt{r}}$. Thus $q_{\mathtt{r}}$ is not final.
iii) $\Rightarrow$: Assume that there exists an orphan message configuration $s \in RS(S)$. Since for all $\mathtt{p} \in \mathbf{P}, q_{\mathtt{p}}$ is final, we have $s \not\rightarrow$. Moreover, $\vec{w} \neq \vec{\varepsilon}$ since s is an orphan message configuration. Thus s is a weak deadlock configuration. Now assume that there exists a configuration $s \in RS(S)$ with no progress. Then, using (ii), s is a weak deadlock configuration.
$\Leftarrow$: Assume that there is a weak deadlock configuration $s \in RS(S)$. Then $s \not\rightarrow$. If a) holds then, according to (ii), there is no progress in s. If a) does not hold, then $\forall \mathtt{p} \in \mathbf{P}$. $q_{\mathtt{p}}$ is final and, since s is a weak deadlock configuration, b) in Def. 2.5 must hold, i.e. $\vec{w} \neq \vec{\varepsilon}$. In this case s is an orphan message configuration. □

In [4], Brand and Zafiropulo showed that most properties of interest, such as logical correctness properties, are in general undecidable for protocols specified in the CFSM model. Undecidability stems from the potential unboundedness of the channels in the model. Even a protocol consisting of two processes communicating over two channels with unbounded capacity appears as powerful as a Turing machine. In essence, the halting problem reduces to the problem of detecting (logical) design errors in a protocol by using the unbounded channels to simulate the tape of a Turing machine [4].
Despite this negative result, decidability of the verification problem is known for some classes of protocols. Most eminently, reachability analysis decides the verification problem for all those protocols whose channels are bounded.
(*PhD Thesis of* Hans van der Schoot)

References

[1] G. Cécé, A. Finkel, Verification of programs with half-duplex communication, Inf. Comput. 202 (2) (2005) 166–190. `doi:10.1016/j.ic.2005.05.006`.

[2] P. Deniélou, N. Yoshida, Multiparty session types meet communicating automata, in: ESOP'12, 2012, pp. 194–213. `doi:10.1007/978-3-642-28869-2_10`.

[3] J. Lange, E. Tuosto, N. Yoshida, From communicating machines to graphical choreographies, in: POPL 2015, 2015, pp. 221–232. `doi:10.1145/2676726.2676964`.

[4] D. Brand, P. Zafiropulo, On communicating finite-state machines, J. ACM 30 (2) (1983) 323–342. `doi:10.1145/322374.322380`.

[5] E. Tuosto, R. Guanciale, Semantics of global view of choreographies, J. Log. Algebr. Meth. Program. 95 (2018) 17–40. `doi:10.1016/j.jlamp.2017.11.002`.