12th IFAC/IEEE Workshop on Discrete Event Systems
Cachan, France. May 14-16, 2014

An Algorithm for Compositional Nonblocking Verification of Extended Finite-State Machines

Sahar Mohajerani [*] **Robi Malik** [**] **Martin Fabian** [*]

[*] *Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden, (e-mail: {mohajera,fabian}@chalmers.se)*
[**] *Department of Computer Science, University of Waikato, Hamilton, New Zealand, (e-mail: robi@waikato.ac.nz)*

Abstract: This paper describes an approach for *compositional nonblocking verification* of discrete event systems modelled as *extended finite-state machines* (EFSM). Previous results about finite-state machines in lock-step synchronisation are generalised and applied to EFSMs communicating via shared variables. This gives rise to an EFSM-based conflict check algorithm that composes EFSMs gradually and *partially unfolds* variables as needed. At each step, components are simplified using conflict-equivalence preserving abstraction. The algorithm has been implemented in the discrete event systems tool Supremica. The paper presents experimental results for the verification of two scalable manufacturing system models, and shows that the EFSM-based algorithm verifies some large models faster than previously used methods.

Keywords: Discrete event systems, extended finite-state machines, compositional verification, nonblocking, abstraction.

1. INTRODUCTION

Many discrete event systems are safety-critical, where failures can result in huge financial losses or even human fatalities. Logical correctness is a crucial property of these systems, and formal verification is an important part of guaranteeing it. This paper focuses on the verification of the *nonblocking* property (Ramadge and Wonham, 1989).

Formal verification requires a formal model, and *finite-state machines (FSM)* are widely used to represent discrete event systems (Ramadge and Wonham, 1989). FSMs describe the behaviour of a system using *states* and *transitions* between these states. Yet, data driven systems are more naturally modelled as *extended finite-state machines (EFSM)*, which communicate through bounded discrete variables. EFSMs have been similarly defined by several researchers (Chen and Lin, 2000; Yang and Gohari, 2005; Sköldstam *et al.*, 2007; Zhaoa *et al.*, 2012; Teixeira *et al.*, 2013).

While variables simplify the modelling of discrete event systems, the verification of large systems remains a challenge due to the *state-space explosion* problem. Verification must take all possible variable values into account, which can result in a large state space. To overcome state space explosion, various approaches including symbolic model checking (Baier and Katoen, 2008; McMillan, 1993) and abstraction (Graf and Steffen, 1990; Dams *et al.*, 1994) have been proposed. Another method is compositional verification using *conflict equivalence*, which has shown impressive results for large FSM models (Flordal and Malik, 2009; Su *et al.*, 2010; Malik and Leduc, 2013).

To apply FSM-based compositional methods to systems modelled as EFSMs, the model is first converted to a set of FSMs (Sköldstam *et al.*, 2007). While the conversion preserves the modular structure, making it possible to apply FSM-based methods directly, it has the drawback of significantly increasing the number of events. In some cases, the conversion takes longer than the verification.

Recently, an adaptation of compositional verification to EFSM models was proposed (Mohajerani *et al.*, 2013*a*), which removes the need to convert EFSMs to FSMs. In that work, *symbolic observation equivalence* is used as the only abstraction method. While observation equivalence reduces the state space significantly, it is not the best possible equivalence for nonblocking verification (Malik *et al.*, 2006). Several conflict-preserving abstraction rules for FSMs are known (Flordal and Malik, 2009; Malik and Leduc, 2013) that extend beyond observation equivalence.

This paper applies the compositional framework (Flordal and Malik, 2009) to systems modelled as EFSMs communicating via shared variables. In addition to *partial unfolding*, which removes variables from the system, and *selfloop removal*, which removes transitions, the paper provides a general framework to apply every conflict-preserving FSM abstraction rule (Flordal and Malik, 2009; Malik and Leduc, 2013) directly to EFSMs. The proposed EFSM-based compositional algorithm is implemented in Supremica (Åkesson *et al.*, 2006), and has been used successfully to verify two scalable manufacturing systems.

This paper is structured as follows. Sect. 2 introduces the notation and concepts for EFSMs. Sect. 3 presents the idea of compositional nonblocking verification of EFSMs and shows how an EFSM can be simplified without converting it to an FSM. Afterwards, Sect. 4 describes the algorithm for compositional nonblocking verification of EFSM systems, Sect. 5 presents the experimental results, and Sect. 6 adds some concluding remarks.

2. PRELIMINARIES

2.1 Finite-State Machines

The standard means to model discrete event systems (Ramadge and Wonham, 1989) are *finite-state machines (FSM)*, which synchronise on shared *events* (Hoare, 1985). Events are taken from a finite alphabet Σ. The special *silent event* $\tau \notin \Sigma$ is not included in Σ unless explicitly mentioned using the notation $\Sigma_\tau = \Sigma \cup \{\tau\}$. Further, Σ^* is the set of all finite *traces* of events from Σ, including the *empty trace* ε. The concatenation of two traces $s, t \in \Sigma^*$ is written as st.

Definition 1. A *finite-state machine (FSM)* is a tuple $G = \langle \Sigma, Q, \rightarrow, Q^\circ, Q^\omega \rangle$, where Σ is a finite set of events, Q is a finite set of *states*, $\rightarrow \subseteq Q \times \Sigma_\tau \times Q$ is the *state transition relation*, $Q^\circ \subseteq Q$ is the set of *initial states*, and $Q^\omega \subseteq Q$ is the set of *marked states*.

The transition relation is written in infix notation $x \xrightarrow{\sigma} y$, and is extended to traces in Σ_τ^* by $x \xrightarrow{\varepsilon} x$ for all $x \in Q$, and $x \xrightarrow{s\sigma} z$ if $x \xrightarrow{s} y$ and $y \xrightarrow{\sigma} z$ for some $y \in Q$. The transition relation is also defined for state sets $X, Y \subseteq Q$, for example $X \xrightarrow{s} y$ means $x \xrightarrow{s} y$ for some $x \in X$.

Definition 2. Let $G_1 = \langle \Sigma_1, Q_1, \rightarrow_1, Q_1^\circ, Q_1^\omega \rangle$ and $G_2 = \langle \Sigma_2, Q_2, \rightarrow_2, Q_2^\circ, Q_2^\omega \rangle$ be two FSMs. The *synchronous composition* of G_1 and G_2 is

$$G_1 \,\|\, G_2 = \langle \Sigma_1 \cup \Sigma_2, Q_1 \times Q_2, \rightarrow, Q_1^\circ \times Q_2^\circ, Q_1^\omega \times Q_2^\omega \rangle \quad (1)$$

where

$$(x_1, x_2) \xrightarrow{\sigma} (y_1, y_2) \text{ if } \sigma \in \Sigma_1 \cap \Sigma_2,\ x_1 \xrightarrow{\sigma}_1 y_1,\ x_2 \xrightarrow{\sigma}_2 y_2\,;$$
$$(x_1, x_2) \xrightarrow{\sigma} (y_1, x_2) \text{ if } \sigma \in (\Sigma_1 \setminus \Sigma_2) \cup \{\tau\},\ x_1 \xrightarrow{\sigma}_1 y_1\,;$$
$$(x_1, x_2) \xrightarrow{\sigma} (x_1, y_2) \text{ if } \sigma \in (\Sigma_2 \setminus \Sigma_1) \cup \{\tau\},\ x_2 \xrightarrow{\sigma}_2 y_2\,.$$

This paper concerns verification of the *nonblocking* property, which is commonly used in supervisory control theory of discrete event systems (Ramadge and Wonham, 1989).

Definition 3. An FSM $G = \langle \Sigma, Q, \rightarrow, Q^\circ, Q^\omega \rangle$ is *nonblocking* if, for every $s \in \Sigma_\tau^*$ and every $x \in Q$ such that $Q^\circ \xrightarrow{s} x$, there exists $t \in \Sigma_\tau^*$ such that $x \xrightarrow{t} Q^\omega$.

2.2 Extended Finite-State Machines

Extended finite-state machines (EFSM) are similar to conventional finite-state machines (FSM), but augmented with *updates* associated to the transitions (Chen and Lin, 2000; Sköldstam *et al.*, 2007). Updates are predicates containing variables.

A *variable* v is an entity associated with a finite domain $\mathrm{dom}(v)$ and an initial value $v^\circ \in \mathrm{dom}(v)$. Let $V = \{v_0, \ldots, v_n\}$ be the set of variables with domain $\mathrm{dom}(V) = \mathrm{dom}(v_0) \times \cdots \times \mathrm{dom}(v_n)$. An element of $\mathrm{dom}(V)$ is also called *valuation* and denoted by $\hat{v} = (\hat{v}_0, \ldots, \hat{v}_n)$ with $\hat{v}_i \in \mathrm{dom}(v_i)$. The *initial valuation* is $\hat{v}^\circ = (v_0^\circ, \ldots v_n^\circ)$. A second set of variables, called *next-state variables* and denoted by $V' = \{\, v' \mid v \in V \,\}$ with $\mathrm{dom}(V') = \mathrm{dom}(V)$, is used to describe how variables are updated by transitions.

For example, let x be a variable with domain $\mathrm{dom}(x) = \{0, \ldots, 5\}$ and initial value $x^\circ = 0$. A transition with update $x' = x + 1$ changes the variable x by adding 1 to its current value, if it currently is less than 5. Otherwise (if $x = 5$) the transition is disabled and no updates are performed. Another possibility is to write the formula $x' = \min(x+1, 5)$, in which case the transition remains enabled when $x = 5$. The update $x = 3$ disables a transition unless $x = 3$ in the current state, and the value of x in the next state is not changed. Differently, the update $x' = 3$ always enables its transition, and the value of x in the next state is forced to be 3.

The set of all update predicates using variables in V and V' is denoted by Π_V. For an update $p \in \Pi_V$, the term $\mathrm{vars}(p)$ denotes the set of all variables that occur in p, and $\mathrm{vars}'(p)$ denotes the set of all variables modified by p. For example, if $p \equiv x' = y+1$ then $\mathrm{vars}(p) = \{x, y\}$, and $\mathrm{vars}'(p) = \{x\}$. An update p with $\mathrm{vars}'(p) = \emptyset$ is called a *pure guard*. Its execution leaves all variables unchanged.

Definition 4. An *extended finite-state machine (EFSM)* is a tuple $E = \langle V, Q, \rightarrow, Q^\circ, Q^\omega \rangle$, where V is a finite set of variables, Q is a finite set of *locations*, $\rightarrow \subseteq Q \times \Pi_V \times Q$ is the *conditional transition relation*, $Q^\circ \subseteq Q$ is the set of *initial locations*, and $Q^\omega \subseteq Q$ is the set of *marked locations*.

The expression $x \xrightarrow{p} y$ denotes the presence of a transition in E, from location x to location y with update $p \in \Pi_V$. On the occurrence of such a transition, the EFSM changes its location from x to y while updating the variables in $\mathrm{vars}'(p)$ in accordance with p; variables not contained in $\mathrm{vars}'(p)$ remain unchanged.

Usually, reactive systems are modelled as several components interacting with each other. An *EFSM system* is a collection of interacting EFSMs,

$$\mathcal{E} = \{E_1, \ldots, E_n\}\ . \quad (2)$$

The behaviour of such a system is expressed using *interleaving* semantics (Baier and Katoen, 2008). Synchronisation is achieved indirectly through the variables.

Definition 5. Given two EFSMs $E = \langle V_E, Q_E, \rightarrow_E, Q_E^\circ, Q_E^\omega \rangle$ and $F = \langle V_F, Q_F, \rightarrow_F, Q_F^\circ, Q_F^\omega \rangle$ the *composition* of E and F is

$$E \,\|\, F = \langle V_E \cup V_F, Q_E \times Q_F, \rightarrow, Q_E^\circ \times Q_F^\circ, Q_E^\omega \times Q_F^\omega \rangle\ , \quad (3)$$

where

- $(x_E, x_F) \xrightarrow{p_E} (y_E, x_F)$ if $x_E \xrightarrow{p_E}_E y_E$;
- $(x_E, x_F) \xrightarrow{p_F} (x_E, y_F)$ if $x_F \xrightarrow{p_F}_F y_F$.

To apply the nonblocking property to EFSMs, they are interpreted as FSMs. The standard approach to do this is *flattening*, which introduces states for all combinations of locations and variable values (Baier and Katoen, 2008).

Definition 6. Let $E = \langle V, Q, \rightarrow, Q^\circ, Q^\omega \rangle$ be an EFSM. The *monolithic flattened* FSM of E is $U(E) = \langle \emptyset, Q_U, \rightarrow_U, Q_U^\circ, Q_U^\omega \rangle$ where

- $Q_U = Q \times \mathrm{dom}(V)$,
- $(x, \hat{v}) \xrightarrow{\tau}_U (y, \hat{w})$ if $x \xrightarrow{p} y$ and $p(\hat{v}, \hat{w}) = \text{true}$,
- $Q_U^\circ = Q^\circ \times \{\hat{v}^\circ\}$
- $Q_U^\omega = Q^\omega \times \mathrm{dom}(V)$.

The domain of variables and the initial valuation $\hat{v}_0$ are determined by variables as they are properties of variables. The variable values ensure the correct sequencing of transitions in the flattened FSM. The flattened FSM of an EFSM system $\mathcal{E} = \{E_1, \ldots, E_n\}$ is $U(\mathcal{E}) = U(E_1 \,\|\, \cdots \,\|\, E_n)$. Using these definitions, the nonblocking property is also defined for EFSMs and EFSM systems.