Technical Report: BDD-based Discrete Analysis of Timed Systems

Nguyen Truong Khanh[2], Jun Sun[1], Yang Liu[2], Jin Song Dong[2] and Yan Liu[2]

[1] Information System Technology and Design
Singapore University of Technology and Design
sunjun@sutd.edu.sg
[2] School of Computing
National University of Singapore
{truongkhanh,liuyang,dongjs,yanliu}@comp.nus.edu.sg

Abstract. Complex timed systems are often composed of many components at multiple levels of hierarchy. *Timed finite-state machines* (TFSMs) were proposed to model timed system components, which are designed to capture useful system features like different ways of communication among system components. In this report, we will present a short introduction about TFSMs and a rich set of system composition functions accordingly based on TFSMs. Then we will explain how to encode a TFSM as a BDD and how to generate BDD encoding of these functions without constructing the composed TFSM.

1 Timed Finite-State Machines

Definition 1. *A TFSM is a tuple $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that GV is a set of finite-domain shared variables; LV is a set of finite-domain local variables such that $GV \cap LV = \varnothing$; S is a finite set of control states; $init \in S$ is the initial state; Act is the alphabet; Ch is a set of synchronous channels[3]; and T is a labeled transition relation. A transition label is of the form $[guard]evt\{prog\}$ where guard is an optional guard condition constituted by variables in GV and LV; evt is either an event name, a channel input/output or the special tick event (which denotes 1-unit time elapsing); and prog is an optional transaction, i.e., a sequential program which updates global/local variables.*

A transaction (which may contain program constructs like *if-then-else* or *while-do*) associated with a transition is to be executed atomically. A non-atomic operation is thus to be broken into multiple transitions. TFSM supports many system features. For instance, TFSM may communicate with each other through shared variables GV, multi-way event synchronization (common events in parallel composition are synchronized) or pair-wise channel communication.

The semantics of $\mathcal{M}$ is a labeled transition system $(C, init_c, \rightarrow)$ such that C contains finitely many configurations of the form (σ_g, σ_l, s) such that σ_g is

[3] Asynchronous channels can be mimicked using shared variables.

the valuation of GV and σ_l is the valuation of LV and $s \in S$ is a control state; $init_c = (init_g, init_l, init)$ where $init_g$ is the initial valuation of GV and $init_l$ is the initial valuation of LV; and $\rightarrow$ is defined as follows: for any (σ_g, σ_l, s), if $(s, [guard]e\{prog\}, s') \in T$, then $(\sigma_g, \sigma_l, s) \xrightarrow{e} (\sigma_g', \sigma_l', s')$ if the following holds: *guard* is true given σ_g and σ_l; e is not a synchronous channel input/output; and *prog* updates σ_g and σ_l to be σ_g' and σ_l' respectively. Notice that synchronous input/output cannot occur on its own. Rather, it must be jointly performed by different TFSMs which execute concurrently. Furthermore, $\rightarrow$ contains transitions labeled with events to be synchronized, which later will be synchronized with corresponding transitions from other TFSMs. We remark that timing constraints are captured explicitly by allowing/disallowing transitions labeled with *tick*. For instance, an urgent state is a state which disallows ticks.

2 System Models and BDD Encoding

A timed system may be built from the bottom up by gradually composing system components. We propose to model system components using timed finite-state machines (TFSM), which are designed to capture a variety of system features. In this following, we introduce TFSM and system compositions based on TFSM. Furthermore, we show abstractly how to generate BDD encoding of TFSM in a compositional way.

2.1 Encoding a TFSM

TFSM can be encoded in BDD following the standard approach. That is, a BDD can be used to encode symbolically the system configuration including valuation of global and local variables as well as the control states. Using two sequences of Boolean variables $\overrightarrow{x}$ and $\overrightarrow{x}'$ (which represent system configurations before and after a transition respectively), transitions of TFSMs can be encoded as BDDs constituted by $\overrightarrow{x}$ and $\overrightarrow{x}'$. An encoded transition is of the form: $g \wedge e \wedge t$ such that g (over $\overrightarrow{x}$) is the encoded guard condition; e is the encoded event and t (over $\overrightarrow{x}$ and $\overrightarrow{x}'$) is an encoded transaction.

The BDD encoding of a TFSM, referred to as a *BDD machine*, is a tuple $\mathcal{B} = (\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In, Tick)$. $\overrightarrow{V}$ is a set of unprimed Boolean variables encoding global variables, event names and channel names, which are fixed for the whole system before encoding. $\overrightarrow{v}$ is a set of variables encoding local variables and local control states; *Init* is a formula over $\overrightarrow{V}$ and $\overrightarrow{v}$ encoding the initial valuation of the variables. *Trans* is a set of encoded transitions *excluding tick-transitions* which are OR-implicit. In other words, *Trans* is equivalent to a logical disjunction of all its element. *Out* (*In*) is a set of Or-implicit encoded transitions labeled with synchronous channel output (input). Note that transitions in *Out* and *In* are to be matched by corresponding input/output from the environment and are thus separated from the rest of the transitions. And *Tick* is also a set of tick-transitions which indicate a time unit elapses. To encode transition, each variable x in $\overrightarrow{V}$ or in $\overrightarrow{v}$ has another copy called x' which denotes the variable x's value after the transition. Similarly the boolean formula f' is the formula created by replacing any variable x in f with the variable x'.

Before giving explanation how to encode a TFSM, we will briefly describe how to support integer variable in BDD. An integer variable a between min_a and max_a in GV or LV is encoded by a set of $\lceil log_2(max_a - min_a + 1)\rceil$ boolean variables. For example, if the variable a in the range [0..3] then we need 2 boolean variables $\{a_0, a_1\}$ to encode the value of variable a, suppose a_0 is the most significant bit.

Then a *BDD machine* $\mathcal{B} = (\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In, Tick)$ of a TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ where

- $\overrightarrow{V} = V' \cup \{event_0, ..., event_n\}$ where V' is the set of boolean variables to encode global variables in the set GV and $event_0, ..., event_{m-1}$ are m boolean variables to encode the actions in Act and channel names in Ch such that $m = \lceil log_2(\mid Act \cup Ch \mid)\rceil$. Let $event$ is the variable whose value is represented by these m boolean variables, and for each action or channel name $a \in Act \cup Ch$, $a.index \in \{0.. \mid Act \cup Ch \mid -1\}$is the unique index of that action or channel name. For optimal performance, static analysis is conducted to get the number of actions and channel names in advance and these boolean variables describing actions and channel name are fixed before the encoding procedure starts.
- $\overrightarrow{v} = v' \cup \{state_0, ..., state_n\}$ where v' is the set of boolean variables to encode local variables in the set LV and $state_0, ..., state_{n-1}$ are n boolean variables to encode the control states in S such that $n = \lceil log_2(\mid S \mid)\rceil$. Let $state$ is the variable whose value is represented by these n boolean variables, and for each state $s \in S$, $s.index \in \{0.. \mid S \mid -1\}$is the unique index of that state.
- $Init = (state = init.index)$
- $Trans = \bigvee(state = s_0.index \wedge g_{bdd} \wedge event = e.index \wedge prog_{bdd} \wedge state' = s'_0.index)$ for all transition $(s, [g]e\{prog\}, s') \in T$ such that e is not a synchronous channel and not a tick action. Note that for simplicity, we don't describe how we encode guard expression g to g_{bdd} and program block $prog$ to $prog_{bdd}$. Interested readers can refer [] for details.
- $Out = \bigvee(state = s_0.index \wedge g_{bdd} \wedge event = e.index \wedge prog_{bdd} \wedge state' = s'_0.index)$ for all transition $(s, [g]e\{prog\}, s') \in T$ such that e is a synchronous channel output.
- $In = \bigvee(state = s_0.index \wedge g_{bdd} \wedge event = e.index \wedge prog_{bdd} \wedge state' = s'_0.index)$ for all transition $(s, [g]e\{prog\}, s') \in T$ such that e is a synchronous channel input.
- $Tick = \bigvee(state = s_0.index \wedge g_{bdd} \wedge event = e.index \wedge prog_{bdd} \wedge state' = s'_0.index)$ for all transition $(s, [g]e\{prog\}, s') \in T$ such that e is a tick action.

2.2 Composition Encoding

A complicated system may consist of many components at different level of hierarchies. Components at the same level may be composed in a variety of ways according to many behavioral patterns. In the following, we define a commonly used system composition functions and show how to generate encoding of the composition without constructing the composed TFSM. Note that explicitly constructing the composed TFSM could be expensive. In the following, we fix two TFSMs $\mathcal{M}_i = (GV, LV_i, S_i, init_i, Act_i, Ch_i, T_i)$ where $i \in \{0, 1\}$ and two BDD machines $\mathcal{B}_i = (\overrightarrow{V}, \overrightarrow{v}_i, Init_i, Trans_i, Out_i, In_i, Tick_i)$ which encode $\mathcal{M}_i$ respectively. $\overrightarrow{v}_0$ and $\overrightarrow{v}_1$ are disjoint and $\overrightarrow{V}$ is always shared.

Untime Function

Event Prefix The event prefix $e \rightarrow \mathcal{M}_0$ describes a TFSM $\mathcal{M}$ which is ready to engage the action e, afterward it will pass the control the TFSM $\mathcal{M}_0$. It remains in the initial state until the action is taken. $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0$; $S = S_0 \cup \{init\}$; $Act = Act_0 \cup \{e\}$; $Ch = Ch_0$; $T = \{(init, e, init_0), (init, tick, init)\} \cup T_0$

A BDD machine of $\mathcal{M}$ is $(\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In)$ such that $\overrightarrow{v} = \overrightarrow{v}_0 \cup \{done\}$ where $done$ is a fresh Boolean variable to manage whether the action e happens and then behave as $\mathcal{M}_0$; $Init = \neg\, done$. $Trans$ contains following transitions

- $\neg\, done \wedge event = e \wedge done' \wedge Init_0$
- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Trans_0$

In contains following transitions

- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in In_0

Out contains following transitions

- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Out_0

Tick contains following transitions

- $\neg\, done \wedge event = tick \wedge \neg\, done'$
- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Tick_0$

Unconditional Choice The unconditional choice offers a choice between TFSMs, which is only resolved right after a TFSM engages the first visible event. Moreover in the context of time, the choice is preserved when TFSMs evolve. An unconditional choice between $\mathcal{M}_0$ and $\mathcal{M}_1$ is a TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1$; $S = ((S_0 \cup done) \times (S_1 \cup done))$; $init = (init_0, init_1)$; $Act = Act_0 \cup Act_1$; $Ch = Ch_0 \cup Ch_1$; and T is the minimum transition relation defined as follows. Notice that we introduce a special state $done$ which denotes the state of one component after the other component is chosen. For any $(s_0, [g_0]e_0\{prog_0\}, s_0') \in T_0$; any $(s_1, [g_1]e_1\{prog_1\}, s_1') \in T_1$,

- if $e_0 = e_1 = tick$, $((s_0, s_1), [g_0 \wedge g_1]tick\{prog_0;\ prog_1\}, (s_0', s_1')) \in T$;
- if $e_0 \neq tick$, $((s_0, s), [g_0]e_0\{prog_0\}, (s_0', done)) \in T$ for all $s \in S_1 \cup \{done\}$;
- if $e_1 \neq tick$, $((s, s_1), [g_1]e_1\{prog_1\}, (done, s_1')) \in T$ for all $s \in S_0 \cup \{done\}$;

– if $e_0 = tick$, $((s_0, done), [g_0]tick\{prog_0\}, (s_0', done)) \in T$;
– if $e_1 = tick$, $((done, s_1), [g_1]tick\{prog_1\}, (done, s_1')) \in T$;

The BDD machine of $\mathcal{M}$ is $(\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In)$ such that $\overrightarrow{v} = \overrightarrow{v}_0 \cup \overrightarrow{v}_1 \cup \{choice\}$ where $choice \in \{-1, 0, 1\}$ is a new variable, $choice = -1$ means the choice is not resolved, $choice = 0$ means $\mathcal{M}_0$ is selected, and $choice = 1$ means $\mathcal{M}_1$ is selected; $Init = Init_0 \wedge Init_1 \wedge choice = -1$. $Trans$ contains following transitions

– $(choice = -1 \vee choice = i) \wedge g_i \wedge e_i \wedge t_i \wedge choice' = i$ where $g_i \wedge e_i \wedge t_i$, $i \in \{0, 1\}$, is a transition in $Trans_i$

In contains following transitions

– $(choice = -1 \vee choice = i) \wedge g_i \wedge e_i \wedge t_i \wedge choice' = i$ where $g_i \wedge e_i \wedge t_i$, $i \in \{0, 1\}$, is a transition in In_i

Out contains following transitions

– $(choice = -1 \vee choice = i) \wedge g_i \wedge e_i \wedge t_i \wedge choice' = i$ where $g_i \wedge e_i \wedge t_i$, $i \in \{0, 1\}$, is a transition in Out_i

$Tick$ contains the following transitions

– $choice = -1 \wedge (g_0 \wedge e_0 \wedge t_0) \wedge (g_1 \wedge e_1 \wedge t_1) \wedge choice' = -1$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Tick_0$, and $g_1 \wedge e_1 \wedge t_1$ is a transition in $Tick_1$
– $choice = i \wedge g_i \wedge e_i \wedge t_i \wedge choice' = i$ where $g_i \wedge e_i \wedge t_i$, $i \in \{0, 1\}$, is a transition in $Tick_i$

Parallel Composition If TFSMs are running in parallel, they need to synchronize on common actions, but they can dependently perform actions besides this intersection. In addition it is required that time progresses at the same rate in both TFSMs; therefore they must synchronize on timed transitions. The parallel composition of $\mathcal{M}_0$ and $\mathcal{M}_1$ is a TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1$; $S = S_0 \times S_1$; $init = (init_0, init_1)$; $Act = Act_0 \cup Act_1$; $Ch = Ch_0 \cup Ch_1$; T is the minimum transition relation such that for any $(s_0, [g_0]e_0\{prog_0\}, s_0') \in T_0$; $(s_1, [g_1]e_1\{prog_1\}, s_1') \in T_1$,

– if $e_0 \notin (Act_0 \cap Act_1) \cup \{tick\}$, $((s_0, s_1), [g_0]e_0\{prog_0\}, (s_0', s_1)) \in T$;
– if $e_1 \notin (Act_0 \cap Act_1) \cup \{tick\}$, $((s_0, s_1), [g_1]e_1\{prog_1\}, (s_0, s_1')) \in T$;
– $((s_0, s_1), [g_0 \wedge g_1]e_0\{prog_0;\ prog_1\}, (s_0', s_1')) \in T$ if $e_0 = e_1$ and $e_0 \in (Act_0 \cap Act_1) \cup \{tick\}$. In order to prevent data race, we assume that $prog_0$ and $prog_1$ do not conflict, i.e., update the same variables to different values.
– if $e_0 = ch!v$ is an output on channel ch with value v; and $e_1 = ch?x$ is a matching channel input, $((s_0, s_1), [g_0 \wedge g_1]ch.v\{prog_0;\ prog_1\}, (s_0', s_1')) \in T$;
– if $e_1 = ch!v$ is a channel output; and $e_0 = ch?x$ is a matching channel input, $((s_0, s_1), [g_0 \wedge g_1]ch.v\{prog_1;\ prog_0\}, (s_0', s_1')) \in T$;

Notice that a channel input/output from $\mathcal{M}_i$ may be matched with an output/input from $\mathcal{M}_{1-i}$ to form a transition in T. It is promoted to Ch at the same time because a channel input/output from $\mathcal{M}_i$ may synchronize with another TFSM in the rest of the system. In the contrast, an event in $Act_0 \cap Act_1 \cup \{tick\}$ must be synchronized by both machines. If $Act_0 \cap Act_1 = \varnothing$, then $\mathcal{M}_0$ and $\mathcal{M}_1$ communicate only through shared variables or channels, which is often referred to as interleaving.

Let $(\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In, Tick)$ be the BDD machine encoding the parallel composition of $\mathcal{B}_0$ and $\mathcal{B}_1$. We have $\overrightarrow{v} = \overrightarrow{v}_0 \cup \overrightarrow{v}_1$; $Init = Init_0 \wedge Init_1$. $Trans$ contains three kinds of transitions.

- local transition: if $g_i \wedge e_i \wedge t_i$ is a transition in $Trans_i$ and e_i is an event which is not to be synchronized (i.e., $e \notin (Act_0 \cap Act_1) \cup \{tick\}$), $Trans$ contains a transition $g_i \wedge e_i \wedge t_i \wedge (\overrightarrow{v}_{1-i} = \overrightarrow{v}'_{1-i})$, where $(\overrightarrow{v}_{1-i} = \overrightarrow{v}'_{1-i})$ denotes that the local variables of $\mathcal{B}_{1-i}$ are unchanged.
- channel communication: if $g_i \wedge e_i \wedge t_i$ is a transition in Out_i; and $g_{1-i} \wedge e_{1-i} \wedge t_{1-i}$ is a transition in In_{1-i}; and e_i and e_{1-i} are matching channel input/output, $Trans$ contains a transition $g_i \wedge g_{1-i} \wedge e_i \wedge t_i \wedge t_{1-i}$[4].
- barrier synchronization: if $g_i \wedge e_i \wedge t_i$ is a transition in $Trans_i$ and $g_{1-i} \wedge e_i \wedge t_{1-i}$ is a transition in $Trans_{1-i}$ and $e_i \in (Act_0 \cap Act_1)$ is a synchronization barrier and t_i and t_{1-i} do not conflict, $Trans$ contains transition $g_i \wedge g_{1-i} \wedge e_i \wedge t_i \wedge t_{1-i}$.

Out/In contains a transition $g_i \wedge e_i \wedge t_i \wedge (\overrightarrow{v}_{1-i} = \overrightarrow{v}'_{1-i})$ if $g_i \wedge e_i \wedge t_i$ is a transition in Out_i/In_i respectively. Transitions in Out, In cannot occur on its own, but could be paired with matching input/output from a TFSM running in parallel later. Lastly, $Tick$ contains transition $g_i \wedge g_{1-i} \wedge tick \wedge t_i \wedge t_{1-i}$ if $g_i \wedge tick \wedge t_i$ is a transition in $Tick_i$ and $g_{1-i} \wedge tick \wedge t_{1-i}$ is in $Tick_{1-i}$.

Interleave Composition The semantic of interleaving composition is entirely similar to the one of parallel composition except that common actions are not required to be synchronized. In other words, actions is performed independently by exactly one TFSM, while the other TFSMs make no progress at all. The interleave composition of $\mathcal{M}_0$ and $\mathcal{M}_1$ is a TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1$; $S = S_0 \times S_1$; $init = (init_0, init_1)$; $Act = Act_0 \cup Act_1$; $Ch = Ch_0 \cup Ch_1$; T is the minimum transition relation such that for any $(s_0, [g_0]e_0\{prog_0\}, s_0') \in T_0$; $(s_1, [g_1]e_1\{prog_1\}, s_1') \in T_1$,

- if $e_0 \neq tick$, $((s_0, s_1), [g_0]e_0\{prog_0\}, (s_0', s_1)) \in T$;
- if $e_1 \neq tick$, $((s_0, s_1), [g_1]e_1\{prog_1\}, (s_0, s_1')) \in T$;
- if $e_0 = ch!v$ is an output on channel ch with value v; and $e_1 = ch?x$ is a matching channel input, $((s_0, s_1), [g_0 \wedge g_1]ch.v\{prog_0;\ prog_1\}, (s_0', s_1')) \in T$;
- if $e_1 = ch!v$ is a channel output; and $e_0 = ch?x$ is a matching channel input, $((s_0, s_1), [g_0 \wedge g_1]ch.v\{prog_1;\ prog_0\}, (s_0', s_1')) \in T$;
- $((s_0, s_1), [g_0 \wedge g_1]e_0\{prog_0;\ prog_1\}, (s_0', s_1')) \in T$ if $e_0 = e_1 = tick$

[4] In our encoding, matching synchronous input/ouput is labeled with the same event.

Let $\mathcal{B} = (\overrightarrow{V}, \overrightarrow{v}, \mathit{Init}, \mathit{Trans}, \mathit{Out}, \mathit{In}, \mathit{Tick})$ be the BDD machine encoding of the interleave composition of two components $\mathcal{M}_0$ and $\mathcal{M}_1$ such that $\overrightarrow{v} = \overrightarrow{v}_0 \cup \overrightarrow{v}_1$; $\mathit{Init} = \mathit{Init}_0 \wedge \mathit{Init}_1$. *Trans* contains two kinds of transitions.

- Local transitions: $g_i \wedge e_i \wedge t_i \wedge (\overrightarrow{v}_{1-i} = \overrightarrow{v}'_{1-i})$ where $g_i \wedge e_i \wedge t_i$, $i \in \{0, 1\}$, is a transition in Trans_i
- Synchronous channel communication: $g_i \wedge e_i \wedge t_i \wedge g_{1-i} \wedge e_{1-i} \wedge t_{1-i}$ where $g_i \wedge e_i \wedge t_i$, and $g_{1-i} \wedge e_{1-i} \wedge t_{1-i}$ are transitions in In_i and Out_{1-i}, $i \in \{0, 1\}$respectively

In contains following transitions:

- $g_i \wedge e_i \wedge t_i \wedge (\overrightarrow{v}_{1-i} = \overrightarrow{v}'_{1-i})$ where $g_i \wedge e_i \wedge t_i$ is a transition in In_i

Out contains following transitions:

- $g_i \wedge e_i \wedge t_i \wedge (\overrightarrow{v}_{1-i} = \overrightarrow{v}'_{1-i})$ where $g_i \wedge e_i \wedge t_i$ is a transition in Out_i

Tick contains following transitions:

- $(g_0 \wedge e_0 \wedge t_0) \wedge (g_1 \wedge e_1 \wedge t_1)$ where $g_0 \wedge e_0 \wedge t_0$, and $g_1 \wedge e_1 \wedge t_1$ are tick transitions in Tick_0, and Tick_1 respectively.

Sequential Composition Sequential composition allows to pass the control to a second process after the first process terminates successfully. When the first process terminates, its ✓ action becomes internal to the sequential composition, because the sequential composition should not indicate that it has terminated until the second process does. The sequential composition of $\mathcal{M}_0;\ \mathcal{M}_1$ is a TFSM $\mathcal{M} = (GV, LV, S, \mathit{init}, \mathit{Act}, \mathit{Ch}, T)$ such that $LV = LV_0 \cup LV_1$; $S = S_0 \cup S_1$; $\mathit{init} = \mathit{init}_0$; $\mathit{Act} = \mathit{Act}_0 \cup \mathit{Act}_1$; $\mathit{Ch} = \mathit{Ch}_0 \cup \mathit{Ch}_1$; T is the minimum transition relation such that for any $(s_0, [g_0]e_0\{\mathit{prog}_0\}, s'_0) \in T_0$; $(s_1, [g_1]e_1\{\mathit{prog}_1\}, s'_1) \in T_1$,

- if $e_0 \neq \checkmark$, $(s_0, [g_0]e_0\{\mathit{prog}_0\}, s'_0) \in T$;
- $(s_1, [g_1]e_1\{\mathit{prog}_1\}, s'_1) \in T$;
- if $e_0 = \checkmark$, $(s_0, [g_0]e_0\{\mathit{prog}_0\}, \mathit{init}_1) \in T$

The BDD machine of $\mathcal{M}_0;\ \mathcal{M}_1$ is $(\overrightarrow{V}, \overrightarrow{v}, \mathit{Init}, \mathit{Trans}, \mathit{Out}, \mathit{In})$ such that $\overrightarrow{v} = \overrightarrow{v}_0 \cup \overrightarrow{v}_1 \cup \{\mathit{terminated}\}$ where *terminated* is a fresh Boolean variable to manage whether $\mathcal{M}_0$ terminates; $\mathit{Init} = \mathit{Init}_0 \wedge \neg\, \mathit{terminated}$. *Trans* contains following transitions

- $\neg\, \mathit{terminated} \wedge g_0 \wedge e_0 \wedge t_0 \wedge \mathit{event} \neq \checkmark \wedge \neg\, \mathit{terminated}'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Trans_0
- $\neg\, \mathit{termniated} \wedge g_0 \wedge \mathit{event} = \tau \wedge t_0 \wedge \mathit{terminated}' \wedge \mathit{Init}_1$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Trans_0. Note that as the composition, we replace the ✓ action of $\mathcal{M}_0$ with the internal action τ.
- $\mathit{terminated} \wedge g_1 \wedge e_1 \wedge t_1 \wedge \mathit{termniated}'$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in Trans_1

In contains following transitions

- $\neg$ *terminated* $\wedge\ g_0 \wedge e_0 \wedge t_0 \wedge \neg$ *terminated*$'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in In_0
- *terminated* $\wedge\ g_1 \wedge e_1 \wedge t_1 \wedge$ *termninated*$'$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in In_1

Out contains following transitions

- $\neg$ *terminated* $\wedge\ g_0 \wedge e_0 \wedge t_0 \wedge \neg$ *terminated*$'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Out_0
- *terminated* $\wedge\ g_1 \wedge e_1 \wedge t_1 \wedge$ *termninated*$'$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in Out_1

Tick contains following transitions

- $\neg$ *terminated* $\wedge\ g_0 \wedge e_0 \wedge t_0 \wedge \neg$ *terminated*$'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Tick_0$
- *terminated* $\wedge\ g_1 \wedge e_1 \wedge t_1 \wedge$ *termninated*$'$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in $Tick_1$

Interrupt The interrupt construction $\mathcal{M}_0 \Delta \mathcal{M}_1$ allows the first TFSM to execute; however the second TFSM can interrupt at any time by an action from TFSM $\mathcal{M}_1$. Different from sequential composition, in the interrupt construction, both $\mathcal{M}_0$, and $\mathcal{M}_1$ must evolve together. The interrupt construction of $\mathcal{M}_0 \Delta \mathcal{M}_1$ is a TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1$; $S = ((S_0 \cup done) \times S_1)$; $init = (init_0, init_1)$; $Act = Act_0 \cup Act_1$; $Ch = Ch_0 \cup Ch_1$; and T is the minimum transition relation defined as follows. Notice that we introduce a special state *done* which denotes the interrupted state of the first component. For any $(s_0, [g_0]e_0\{prog_0\}, s_0') \in T_0$; any $(s_1, [g_1]e_1\{prog_1\}, s_1') \in T_1$,

- if $e_0 \neq tick$, $((s_0, s_1), [g_0]e_0\{prog_0\}, (s_0', s_1)) \in T$;
- if $e_1 \neq tick$, $((s_0, s_1), [g_1]e_1\{prog_1\}, (done, s_1')) \in T$;
- $((done, s_1), [g_1]e_1\{prog_1\}, (done, s_1')) \in T$;
- $((s_0, s_1), [g_0 \wedge g_1]e_0\{prog_0;\ prog_1\}, (s_0', s_1')) \in T$ if $e_0 = e_1 = tick$

A BDD machine of $\mathcal{M}_0 \Delta \mathcal{M}_1$ is $(\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In)$ such that $\overrightarrow{v} = \overrightarrow{v}_0 \cup \overrightarrow{v}_1 \cup \{interrupted\}$ where *interrupted* is a fresh Boolean variable to manage whether $\mathcal{M}_1$ interrupts $\mathcal{M}_0$; $Init = Init_0 \wedge Init_1 \wedge \neg\, interrupted$. *Trans* contains following transitions

- $\neg\, interrupted \wedge g_0 \wedge e_0 \wedge t_0 \wedge \neg\, interrupted' \wedge (\overrightarrow{v}_1 = \overrightarrow{v}_1')$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Trans_0$
- $g_1 \wedge e_1 \wedge t_1 \wedge interrupted'$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in $Trans_1$

In contains following transitions

- $\neg\, interrupted \wedge g_0 \wedge e_0 \wedge t_0 \wedge \neg\, interrupted' \wedge (\overrightarrow{v}_1 = \overrightarrow{v}_1')$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in In_0
- $g_1 \wedge e_1 \wedge t_1 \wedge interrupted'$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in In_1

Out contains following transitions

- $\neg$ *interrupted* $\wedge\, g_0 \wedge e_0 \wedge t_0 \wedge \neg$ *interrupted*$' \wedge (\overrightarrow{v}_1 = \overrightarrow{v}'_1)$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Out_0
- $g_1 \wedge e_1 \wedge t_1 \wedge$ *interrupted*$'$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in Out_1

Tick contains following transitions

- $\neg$ *interrupted* $\wedge (g_0 \wedge e_0 \wedge t_0) \wedge (g_1 \wedge e_1 \wedge t_1) \wedge \neg$ *interrupted*$'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Tick_0$, and $g_1 \wedge e_1 \wedge t_1$ is a transition in $Tick_1$
- *interrupted* $\wedge\, g_1 \wedge e_1 \wedge t_1 \wedge$ *interrupted*$'$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in $Tick_1$

Channel Out Let a is a channel name of type T, and *exps* is a particular value of type T, then then channel output $a?exps \rightarrow \mathcal{M}_0$ describes a TFSM which can output *exps* along channel a and subsequently behaves as $\mathcal{M}_0$. However this output can not occur on its own but must be synchronized with a corresponding channel input from other components. The TFSM of $a?exps \rightarrow \mathcal{M}_0$ is a new TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1$; $S = S_0 \cup \{init\}$; $Act = Act_0 \cup \{a?exps\}$; $Ch = Ch_0 \cup \{a\}$; $T = \{(init, a?exps, init_0), (init, tick, init)\} \cup T_0$; and $Ch = Ch_0$.

Let $\mathcal{B} = (\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In, Tick)$ be the BDD machine encoding of $a?exps \rightarrow \mathcal{M}_0$. We have $\overrightarrow{v} = \overrightarrow{v}_0 \cup \{done\}$; $Init = \neg\, done$. *Trans* contains the following transitions

- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Trans_0$

In contains the following transitions

- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in In_0

Out contains the following transitions

- $\neg\, done \wedge (count_a < L) \wedge [\bigwedge_{i=1..exps.count}(a[top_a][i]' = exps[i])] \wedge (size_a[top_a]' = exps.count) \wedge (count'_a = count_a + 1) \wedge top_a = (top_a + 1)\%L \wedge done' \wedge Init_0$ where $count_a$ is the number of the elements in the channel buffer, top_a is the position to put new element int the buffer, L is the buffer length of the channel a, and $size_a$ is an array to manage the number of the messages in the buffer. The guard of the channel out transition includes *done* is false, and the channel buffer is not full. After the channel in transition, elements from the expression *exps* is pushed to the buffer. The size of the expression is also updated to $size_a[top_a]$. Moreover the channel buffer updates its size $count_a$ and tail position top_a . *done* is set false to constrain the channel out transition to happen once and then pass the control to the process P_0.
- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Out_0

Tick contains the following transitions

- $\neg\, done \wedge event = tick \wedge \neg\, done'$
- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Tick_0$

Channel In A channel in is similar to channel out, but is ready to accept any value x of type T along channel a. The TFSM of $a!exps \rightarrow \mathcal{M}_0$ is a new TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1$; $S = S_0 \cup \{init\}$; $Act = Act_0 \cup \{a!exps\}$; $Ch = Ch_0$; $T = \{(init, a!exps, init_0), (init, tick, init)\} \cup T_0$; and $Ch = Ch_0 \cup \{a\}$.

Let $\mathcal{B} = (\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In, Tick)$ be the BDD machine encoding of $[b]a!exps \rightarrow P_1$. We have $\overrightarrow{v} = \overrightarrow{v}_0 \cup \{done\}$; $Init = \neg\, done$. *Trans* contains the following transitions

- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Trans_0$

In contains the following transitions

- $\neg\, done \wedge b \wedge (count_a > 0) \wedge (size_a[(top_a - count_a)\%L] = exps.count) \wedge [\bigwedge_{i=1..exps.count}(exps[i]' = a[(top_a - count_a)\%L][i])] \wedge (count'_a = count_a - 1) \wedge done' \wedge Init_0$. The guard of the channel in transition includes *done* is false, the guard condition b is satisfied, the channel buffer is not empty and the size of the message in the top of the buffer is equal to the size of the channel in expression. After the transition, variable in the channel in expression is updated with the element in the channel buffer and the buffer also updates its size.
- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in In_0

Out contains the following transitions

- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Out_0

Tick contains the following transitions

- $\neg\, done \wedge event = tick \wedge \neg\, done'$
- $done \wedge g_0 \wedge e_0 \wedge t_0 \wedge done'$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Tick_0$

Time Function

Delay A TFSM $Wait[t]$ exactly delays for a period of t time units then terminates. $Wait[t]$ is a TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ where $LV = \varnothing$; $S = \{s_i \mid 0 \le i \le t+1\}$, $init = s_0$, $Act = \{\checkmark\}$, $Ch = \varnothing$ and T contains following transitions

- $(s_i, tick, s_{i+1})$ where $0 \le i \le t-1$
- $(s_t, tick, s_t)$
- $(s_t, \checkmark, s_{t+1})$

Because the $Wait[t]$ is a simple TFSM which is not composed by other TFSMs, the BDD machine encoding of $Wait[t]$ is achieved by directly encoding its TFSM.

Timeout The timeout operator $\mathcal{M}_0 timeout[t]\mathcal{M}_1$ offers a time sensitive choice between $\mathcal{M}_0$, and $\mathcal{M}_1$. Initially the control belongs to the TFSM $\mathcal{M}_0$. If $\mathcal{M}_0$ performs any visible action, then the timeout is resolved in favor of $\mathcal{M}_0$ and $\mathcal{M}_1$ is discarded. However if after t time units, $\mathcal{M}_0$ does not engage any visible action, the control is passed to $\mathcal{M}_1$ and $\mathcal{M}_0$ is discarded. The timeout construction of $\mathcal{M}_0 timeout[t]\mathcal{M}_1$ is a TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1$; $S = S_0 \cup S_1 \cup \{state_i \mid 1 \leq i \leq t\}$; $init = init_0$; $Act = Act_0 \cup Act_1$; $Ch = Ch_0 \cup Ch_1$; and T is the minimum transition relation defined as follows. Notice that we introduce t states to remember the time passage while the $\mathcal{M}_0$ delays its first visible action. For any $(s_0, [g_0]e_0\{prog_0\}, s_0') \in T_0$; any $(s_1, [g_1]e_1\{prog_1\}, s_1') \in T_1$, T contains below transitions

- $(init_0, tick, state_1)$
- $(state_i, tick, state_{i+1})$ where $1 \leq i \leq t-1$
- $(state_t, \tau, init_1)$. The timeout occurs and the control is passed to $\mathcal{M}_1$.
- $(s, [g_0]e_0\{prog_0\}, s_0')$ where e_0 is a visible action and $s \in init_0 \cup \{state_1, ..., state_t\}$. We are copying the first visible action to the new t states $state_i$ to allow it happens within t time units.
- any transition from T_1, $(s_1, [g_1]e_1\{prog_1\}, s_1')$
- $(s_0, [g_0]e_0\{prog_0\}, s_0')$ where $s_0 \neq init_0$

Let $\mathcal{B} = (\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In, Tick)$ be the BDD machine encoding of $P_0 timeout[t]P_1$. We have $\overrightarrow{v} = \overrightarrow{v}_0 \cup \overrightarrow{v}_1 \cup \{clk\}$ where $-1 \leq clk \leq t+1$ records the number time units elapsed so far, $clk = -1$ indicates that an visible event of P_0 is engaged, and $clk = t+1$ indicates that t time units elapse and P_1 takes the control, and $Init = Init_0 \wedge clk = 0$; and $Trans$ contains following transitions:

- $clk \leq t \wedge g_0 \wedge e_0 \wedge t_0 \wedge [(event = \tau \wedge clk' = clk) \vee (event \neq \tau \wedge clk' = -1)]$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Trans_0$
- $clk = t \wedge event = \tau \wedge clk' = t+1 \wedge Init_1'$
- $clk = t+1 \wedge g_1 \wedge e_1 \wedge t_1 \wedge clk' = t+1$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in $Trans_1$

Out, *In* are defined like *Trans*. *In* contains following channel transitions:

- $clk \leq t \wedge g_0 \wedge e_0 \wedge t_0 \wedge clk' = -1$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in In_0
- $clk = t+1 \wedge g_1 \wedge e_1 \wedge t_1 \wedge clk' = t+1$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in In_1

Out contains following channel transitions:

- $clk \leq t \wedge g_0 \wedge e_0 \wedge t_0 \wedge clk' = -1$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Out_0
- $clk = t+1 \wedge g_1 \wedge e_1 \wedge t_1 \wedge clk' = t+1$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in Out_1

Tick includes below transitions.

- $g_0 \wedge e_0 \wedge t_0 \wedge (clk \geq 0 \wedge clk < t \wedge clk' = clk+1) \vee (clk = -1 \wedge clk' = -1)$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Tick_0$
- $clk = t+1 \wedge g_1 \wedge e_1 \wedge t_1 \wedge clk' = t+1$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in $Tick_1$

Time Interrupt The TFSM $\mathcal{M}_0 interrupt[t]\mathcal{M}_0$ behaves as P_0 until t time units elapse and then switches to $\mathcal{M}_1$. There is no need for $\mathcal{M}_1$ to execute concurrently with $\mathcal{M}_0$ because it is not invoked after t time units. It is not trivial to generate the TFSM of the time interrupt without the presence of a new variable because we need to count the number of happening tick transitions which can occurs at any time. Therefore for the time interrupt, a new variable is presented. The time interrupt of $\mathcal{M}_0 interrupt[t]\mathcal{M}_1$ is a TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1 \cup \{clk\}$ where $clk \in \{-1..t+1\}$ is a new variable to count the time passage which is initialized with 0; $clk = -1$ to discard $\mathcal{M}_1$ after $\mathcal{M}_0$ terminates and $clk = t+1$ to discard $\mathcal{M}_0$ after $\mathcal{M}_1$ interrupts, $S = S_0 \cup S_1$; $init = init_0$; $Act = Act_0 \cup Act_1$; $Ch = Ch_0 \cup Ch_1$; and T is the minimum transition relation defined as follows. For any $(s_0, [g_0]e_0\{prog_0\}, s_0') \in T_0$; any $(s_1, [g_1]e_1\{prog_1\}, s_1') \in T_1$, T contains below transitions

- $(s_0, [g_0]e_0\{prog_0\}, s_0')$ if $e_0 \neq tick \wedge e_0 \neq \checkmark$
- $(s_0, [g_0]e_0\{prog_0;\ clk = -1\}, s_0')$ if $e_0 \neq tick \wedge e_0 = \checkmark$
- $(s_0, [g_0]e_0\{prog_0;\ if\{0 \leq clk < t\}clk++elseif\{clk = -1\}clk = -1\}, s_0')$ if $e_0 = tick$
- $(s, [clk = t]\tau\{clk = t+1\}, init_1)$ for all $s \in S_0$
- $(s_1, [g_1 \wedge clk = t+1]e_1\{prog_1\}, s_1')$

Let $\mathcal{B} = (\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In, Tick)$ be the BDD machine encoding $P_0 interrupt[t]P_1$. We have $\overrightarrow{v} = \overrightarrow{v}_0 \cup \overrightarrow{v}_1 \cup \{clk\}$, $-1 \leq clk \leq t+1$ and $Init = Init_0 \wedge clk = 0$; and $Trans$ contains following transitions:

- $clk \leq t \wedge g_0 \wedge e_0 \wedge t_0 \wedge [(event = \checkmark \wedge clk' = -1) \vee (event \neq \checkmark \wedge clk' = clk)]$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Trans_0$
- $clk = t \wedge event = \tau \wedge clk' = t+1 \wedge Init_1'$
- $clk = t+1 \wedge g_1 \wedge e_1 \wedge t_1 \wedge clk' = t+1$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in $Trans_1$

In contains following transitions:

- $clk \leq t \wedge g_0 \wedge e_0 \wedge t_0 \wedge clk' = clk$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in In_0
- $clk = t+1 \wedge g_1 \wedge e_1 \wedge t_1 \wedge clk' = t+1$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in In_1

Out contains following transitions:

- $clk \leq t \wedge g_0 \wedge e_0 \wedge t_0 \wedge clk' = clk$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Out_0
- $clk = t+1 \wedge g_1 \wedge e_1 \wedge t_1 \wedge clk' = t+1$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in Out_1

Tick includes below transitions:

- $g_0 \wedge e_0 \wedge t_0 \wedge [(0 \leq clk < t \wedge clk' = clk+1) \vee (clk = -1 \wedge clk' = -1)]$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Tick_0$
- $clk = t+1 \wedge g_1 \wedge e_1 \wedge t_1 \wedge clk' = t+1$ where $g_1 \wedge e_1 \wedge t_1$ is a transition in $Tick_1$

Deadline A timed system requirement may put an bound on the execution time of a component, i.e., a component must terminate before certain time units. A TFSM $\mathcal{M}_0$ with a deadline d is a TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0$; $S = S_1 \times \{0, 1, \cdots, d\}$ where the number is the number of time unit that has elapsed; $init = (init_0, 0)$; $Act = Act_0$; $Ch = Ch_0$; and T is the minimum transition relation such that:

- for any $(s, [g]e\{prog\}, s') \in T_0$ and $e \neq tick$, $((s, d_1), [g]e\{prog\}, (s', d_1)) \in T$ for all $d_1 \in \{0, 1, \cdots, d\}$.
- for any $(s, [g]tick\{prog\}, s') \in T_0$, $((s, d_1), [g]tick\{prog\}, (s', d_1 + 1)) \in T$ for all $d_1 \in \{0, 1, \cdots, d - 1\}$.

Let $\mathcal{B} = (\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In, Tick)$ be the BDD machine encoding of $P_0\,deadline[t]$ where $\overrightarrow{v} = \overrightarrow{v}_0 \cup \{clk\}$, $-1 \leq clk \leq t$ records the number of elapsed time units so far, $clk = -1$ when the deadline is resolved; $Init = Init_0 \wedge clk = 0$; and *Trans* includes below transitions:

- $clk \leq t \wedge g_0 \wedge e_0 \wedge t_0 \wedge [(event \neq \checkmark \wedge clk' = clk) \vee (event = \checkmark \wedge clk' = -1)]$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Trans_0$

In includes below transitions:

- $clk \leq t \wedge g_0 \wedge e_0 \wedge t_0 \wedge clk' = clk$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in In_0

Out includes below transitions:

- $clk \leq t \wedge g_0 \wedge e_0 \wedge t_0 \wedge clk' = clk$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Out_0

Tick includes below transitions:

- $[(0 \leq clk < t \wedge clk' = clk + 1) \vee (clk = -1 \wedge clk' = -1)] \wedge g_0 \wedge e_0 \wedge t_0$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Tick_0$

Within: The *Within* operator on the TFSM $\mathcal{M}_0$ forces it to make an observable move within the given time frame. The Within operator of $\mathcal{M}_0 within[t]$ is a TFSM $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0$; $S = S_0 \cup \{state_i \mid 1 \leq i \leq t\}$; $init = init_0$; $Act = Act_0$; $Ch = Ch_0$; and T is the minimum transition relation defined as follows. For any $(s_0, [g_0]e_0\{prog_0\}, s_0') \in T_0$, T contains below transitions

- $(init_0, tick, state_1)$
- $(state_i, tick, state_{i+1})$ where $1 \leq i \leq t - 1$
- $(s, [g_0]e_0\{prog_0\}, s_0')$ where e_0 is a visible action and $s \in init_0 \cup \{state_1, ..., state_t\}$. We are copying the first visible action to the new t states $state_i$ to allow it happens within t time units.
- $(s_0, [g_0]e_0\{prog_0\}, s_0')$ where $s_0 \neq init_0$

Let $\mathcal{B} = (\overrightarrow{V}, \overrightarrow{v}, Init, Trans, Out, In, Tick)$ be the BDD machine encoding of $\mathcal{M}_0 within[t]$ where $\overrightarrow{v} = \overrightarrow{v}_0 \cup \{clk\}$, $-1 \leq clk \leq t$ records the number elapsed time units so far and $clk = -1$ indicates an visible action just happens and $Init = Init_0 \wedge clk = 0$; and *Trans* includes below transitions

– $clk \leq t \wedge g_0 \wedge e_0 \wedge t_0 \wedge [(event \neq \tau \wedge clk' = -1) \vee (event = \tau \wedge clk' = clk)]$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Trans_0$

In includes below transitions

– $clk < t \wedge g_0 \wedge e_0 \wedge t_0 \wedge clk' = -1$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in In_0

Out includes below transitions

– $clk < t \wedge g_0 \wedge e_0 \wedge t_0 \wedge clk' = -1$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in Out_0

Tick includes below transitions:

– $g_0 \wedge e_0 \wedge t_0 \wedge [(clk \geq 0 \wedge clk < t \wedge clk' = clk+1) \vee (clk = -1 \wedge clk' = -1)]$ where $g_0 \wedge e_0 \wedge t_0$ is a transition in $Tick_0$

3 Conclusion

In this report, we present two ways of encoding a TFSM. The first way is directly encoding it as a TFSM to BDD. The second way is encoding its components and then combining them by using composition functions. Each way has its own pros and cons. The former does not need to create new variables but extend the set of control states S and transition function T. This may get a smaller BDD in two aspects. First having more variables makes the BDD more complex. Second, we may utilize some redundant values of some variables when extending the value range. For example, suppose the event prefix $e \rightarrow \mathcal{M}_0$ where S_0 has 3 states. If we use the second way to encode it, we will use 2 boolean variable to encode the set of states S_0 then one more variable is introduced when using event prefix composition function. However if we use the first way, then a new TFSM is generated based on $\mathcal{M}_0$ which has 4 states. Therefore we also use only 2 boolean variables to encode the control state set, compared to the first way using 3 boolean variables to encode 2 separate variables. However the first way may have trouble when generating the TFSM for parallel or interleaving composition when the control set becomes complex, in the form of Cartesian product of other control set products. The latter approach offers a clear advantage in these cases. Therefore the first way should be used instead of the second way until parallel or interleave composition is needed. In our work, some static analysis is run to find the largest components which can be represented as a TFSM. After these TFSMs are encoded as the first way, composition functions are used to compose these components.