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 \( GV \cap LV = \emptyset \);
- \( S \) 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 \([\text{guard}]\text{evt}\{\text{prog}\}\) where \( \text{guard} \) is an optional guard condition constituted by variables in \( GV \) and \( LV \); \( \text{evt} \) is either an event name, a channel input/output or the special tick event (which denotes 1-unit time elapsing); and \( \text{prog} \) 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 \( (C, init, \rightarrow) \) such that

\( C \) contains finitely many configurations of the form \( (\sigma_g, \sigma_l, s) \) such that \( \sigma_g \) is

---

\(^3\) Asynchronous channels can be mimicked using shared variables.
the valuation of $GV$ and $\sigma_l$ is the valuation of $LV$ and $s \in S$ is a control state; $\text{init}_c = (\text{init}_g, \text{init}_l, \text{init})$ where $\text{init}_g$ is the initial valuation of $GV$ and $\text{init}_l$ is the initial valuation of $LV$; and $\rightarrow$ is defined as follows: for any $(\sigma_g, \sigma_l, s)$, if $(s, [\text{guard}] e \in \{\text{prog}\}, s') \in T$, then $(\sigma_g, \sigma_l, s) \xrightarrow{e} (\sigma'_g, \sigma'_l, s')$ if the following holds: $\text{guard}$ is true given $\sigma_g$ and $\sigma_l$; $e$ is not a synchronous channel input/output; and $\text{prog}$ updates $\sigma_g$ and $\sigma_l$ to be $\sigma'_g$ and $\sigma'_l$ 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 $\text{tick}$. 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 $V$ and $V'$ (which represent system configurations before and after a transition respectively), transitions of TFSMs can be encoded as BDDs constituted by $V$ and $V'$. An encoded transition is of the form: $g \land e \land t$ such that $g$ (over $V$) is the encoded guard condition; $e$ is the encoded event and $t$ (over $V$ and $V'$) is an encoded transaction.

The BDD encoding of a TFSM, referred to as a BDD machine, is a tuple $\mathcal{B} = (\overline{V}, \overline{V'}, \overline{Init}, \overline{Trans}, \overline{Out}, \overline{In}, \overline{Tick})$. $\overline{V}$ is a set of unprimed Boolean variables encoding global variables, event names and channel names, which are fixed for the whole system before encoding. $\overline{V'}$ is a set of variables encoding local variables and local control states; $\overline{Init}$ is a formula over $\overline{V}$ and $\overline{V'}$ encoding the initial valuation of the variables. $\overline{Trans}$ is a set of encoded transitions excluding tick-transitions which are OR-implicit. In other words, $\overline{Trans}$ is equivalent to a logical disjunction of all its element. $\overline{Out}$ ($\overline{In}$) is a set of Or-implicit encoded transitions labeled with synchronous channel output (input). Note that transitions in $\overline{Out}$ and $\overline{In}$ are to be matched by corresponding input/output from the environment and are thus separated from the rest of the transitions. And $\overline{Tick}$ is also a set of tick-transitions which indicate a time unit elapses. To encode transition, each variable $x$ in $\overline{V}$ or in $\overline{V'}$ has another copy called $x'$ which denotes the variable
variables. For example, if the variable 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 $B = (\overline{V}, \overline{\varphi}, \text{Init}, \text{Trans}, \text{Out}, \text{In}, \text{Tick})$ of a TFSM $\mathcal{M} = (GV, LV, S, \text{init}, \text{Act}, Ch, T)$ where

- $\overline{V} = V' \cup \{event_0, ..., event_n\}$ 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 $\text{Act}$ and channel names in $Ch$ such that $m = \lceil \log_2(|\text{Act} \cup Ch|) \rceil$. Let $event$ is the variable whose value is represented by these $m$ boolean variables, and for each action or channel name $a \in \text{Act} \cup Ch$, $a.index \in \{0..|\text{Act} \cup Ch| - 1\}$ 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.

- $\overline{\varphi} = v' \cup \{state_0, ..., state_n\}$ 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 $n = \lceil \log_2(|S|) \rceil$. Let $state$ is the variable whose value is represented by these $n$ boolean variables, and for each state $s \in S$, $s.index \in \{0..|S| - 1\}$ is the unique index of that state.

- $\text{Init} = (state = \text{init.index})$

- $\text{Trans} = \bigvee (state = s_0.index \land g_{bdd} \land event = e.index \land prog_{bdd} \land state' = s'_0.index)$ for all transition $(s, \{g\}e\{\text{prog}\}, s', s) \in T$ 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 [1] for details.

- $\text{Out} = \bigvee (state = s_0.index \land g_{bdd} \land event = e.index \land prog_{bdd} \land state' = s'_0.index)$ for all transition $(s, \{g\}e\{\text{prog}\}, s') \in T$ such that $e$ is a synchronous channel output.

- $\text{In} = \bigvee (state = s_0.index \land g_{bdd} \land event = e.index \land prog_{bdd} \land state' = s'_0.index)$ for all transition $(s, \{g\}e\{\text{prog}\}, s') \in T$ such that $e$ is a synchronous channel input.

- $\text{Tick} = \bigvee (state = s_0.index \land g_{bdd} \land event = e.index \land prog_{bdd} \land state' = s'_0.index)$ for all transition $(s, \{g\}e\{\text{prog}\}, s') \in T$ 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 $M_i = (GV, LV, S, init, Act, Ch, T_i)$ where $i \in \{0, 1\}$ and two BDD machines $B_i = (\overline{V}, \overline{\overline{V}}, Init, Trans_i, Out_i, In_i, Tick_i)$ which encode $M_i$ respectively. $\overline{V}_0$ and $\overline{V}_1$ are disjoint and $\overline{V}$ is always shared.

**Untime Function**

*Event Prefix* The event prefix $e \rightarrow M_0$ describes a TFSM $M$ which is ready to engage the action $e$, afterward it will pass the control the TFSM $M_0$. It remains in the initial state until the action is taken. $M = (GV, LV, S, init, Act, Ch, T)$ 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 $M$ is $(\overline{V}, \overline{\overline{V}}, Init, Trans, Out, In)$ such that $\overline{V} = \overline{V}_0 \cup \{done\}$ where $done$ is a fresh Boolean variable to manage whether the action $e$ happens and then behave as $M_0$: $init = \neg done$. $Trans$ contains following transitions

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

*In* contains following transitions

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

*Out* contains following transitions

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

*Tick* contains following transitions

- $\neg done \land event = tick \land \neg done'$
- $done \land g_0 \land e_0 \land t_0 \land done'$ where $g_0 \land e_0 \land t_0$ 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 $M_0$ and $M_1$ is a TFSM $M = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1; \ S = ((S_0 \cup done) \times (S_1 \cup done)); \ init = (init_0, init_1); \ Act = Act_0 \cup Act_1; \ Ch = Ch_0 \cup Ch_1; \ T$ 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 $e_0 = e_1 = tick$, $(s_0, s_1), [g_0 \land g_1]tick\{prog_0 \land prog_1\}, (s_0', s_1') \in T$;
- if $e_0 \neq tick$, $(s_0, s_1), [g_0]e_0\{prog_0\}, (s_0', done) \in T$ for all $s \in S_0 \cup \{done\}$;
- 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 $e_0 = \text{tick}$, $((s_0, \text{done}), [g_0]t\text{ick}\{\text{prog}_0\}, (s'_0, \text{done})) \in T$;

- if $e_1 = \text{tick}$, $((\text{done}, s_1), [g_1]t\text{ick}\{\text{prog}_1\}, (\text{done}, s'_1)) \in T$;

The BDD machine of $M$ is $(\overline{V}, \overline{\nu}, \text{Init}, \text{Trans}, \text{Out}, \text{In})$ such that $\overline{\nu} = \overline{\nu}_0 \cup \overline{\nu}_1 \cup \{\text{choice}\}$ where $\text{choice} \in \{-1, 0, 1\}$ is a new variable, $\text{choice} = -1$ means the choice is not resolved, $\text{choice} = 0$ means $M_0$ is selected, and $\text{choice} = 1$ means $M_1$ is selected: $\text{Init} = \text{Init}_0 \land \text{Init}_1 \land \text{choice} = -1$. $\text{Trans}$ contains following transitions

- $(\text{choice} = -1 \lor \text{choice} = i) \land g_i \land e_i \land t_i \land \text{choice}' = i$ where $g_i \land e_i \land t_i$, $i \in \{0, 1\}$, is a transition in $\text{Trans}_i$

$\text{In}$ contains following transitions

- $(\text{choice} = -1 \lor \text{choice} = i) \land g_i \land e_i \land t_i \land \text{choice}' = i$ where $g_i \land e_i \land t_i$, $i \in \{0, 1\}$, is a transition in $\text{In}_i$

$\text{Out}$ contains following transitions

- $(\text{choice} = -1 \lor \text{choice} = i) \land g_i \land e_i \land t_i \land \text{choice}' = i$ where $g_i \land e_i \land t_i$, $i \in \{0, 1\}$, is a transition in $\text{Out}_i$

$\text{Tick}$ contains the following transitions

- $\text{choice} = -1 \land (g_0 \land e_0 \land t_0) \land (g_1 \land e_1 \land t_1) \land \text{choice}' = -1$ where $g_0 \land e_0 \land t_0$ is a transition in $\text{Tick}_0$, and $g_1 \land e_1 \land t_1$ is a transition in $\text{Tick}_1$

- $\text{choice} = i \land g_i \land e_i \land t_i \land \text{choice}' = i$ where $g_i \land e_i \land t_i$, $i \in \{0, 1\}$, is a transition in $\text{Tick}_i$

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 $M_0$ and $M_1$ is TFSM $M = (GV, LV, S, \text{init}, \text{Act}, \text{Ch}, T)$ such that $LV = LV_0 \cup LV_1$; $S = S_0 \times S_1$; $\text{init} = (\text{init}_0, \text{init}_1)$; $\text{Act} = \text{Act}_0 \cup \text{Act}_1$; $\text{Ch} = \text{Ch}_0 \cup \text{Ch}_1$; $T$ is the minimum transition relation such that for any $(s_0, [g_0]e_0\{\text{prog}_0\}, s'_0) \in T_0$; $(s_1, [g_1]e_1\{\text{prog}_1\}, s'_1) \in T_1$,

- if $e_0 \notin (\text{Act}_0 \cap \text{Act}_1) \cup \{\text{tick}\}$, $((s_0, s_1), [g_0]e_0\{\text{prog}_0\}, (s'_0, s'_1)) \in T$;

- if $e_1 \notin (\text{Act}_0 \cap \text{Act}_1) \cup \{\text{tick}\}$, $((s_0, s_1), [g_1]e_1\{\text{prog}_1\}, (s'_0, s'_1)) \in T$;

- $((s_0, s_1), [g_0 \land g_1]e_0\{\text{prog}_0; \text{prog}_1\}, (s'_0, s'_1)) \in T$ if $e_0 = e_1$ and $e_0 \in (\text{Act}_0 \cap \text{Act}_1) \cup \{\text{tick}\}$. In order to prevent data race, we assume that $\text{prog}_0$ and $\text{prog}_1$ do not conflict, i.e., update the same variables to different values.

- if $e_0 = \text{chv}$ is an output on channel $ch$ with value $v$; and $e_1 = \text{chv}$ is a matching channel input, $((s_0, s_1), [g_0 \land g_1]ch.v\{\text{prog}_0; \text{prog}_1\}, (s'_0, s'_1)) \in T$;

- if $e_1 = \text{chv}$ is a channel output; and $e_0 = \text{chv}$ is a matching channel input, $((s_0, s_1), [g_0 \land g_1]ch.v\{\text{prog}_0\}, (s'_0, s'_1)) \in T$;
Notice that a channel input/output from $M_t$ may be matched with an output/input from $M_{t-1}$ to form a transition in $T$. It is promoted to $Ch$ at the same time because a channel input/output from $M_t$ may synchronize with another TFSM in the rest of the system. In the contrast, an event in $Act_0 \cap Act_1 \cup \{\text{tick}\}$ must be synchronized by both machines. If $Act_0 \cap Act_1 = \emptyset$, then $M_0$ and $M_1$ communicate only through shared variables or channels, which is often referred to as interleaving.

Let $(\tilde{V}, \tilde{V}, \text{Init}, \text{Trans}, \text{Out}, \text{In}, \text{Tick})$ be the BDD machine encoding the parallel composition of $B_0$ and $B_1$. We have $\tilde{V} = \tilde{V}_0 \cup \tilde{V}_1$; $\text{Init} = \text{Init}_0 \land \text{Init}_1$. $\text{Trans}$ contains three kinds of transitions.

- local transition: if $g_i \land e_i \land t_i$ is a transition in $Trans_i$ and $e_i$ is an event which is not to be synchronized (i.e., $e \not\in (Act_0 \cap Act_1) \cup \{\text{tick}\}$), $Trans$ contains a transition $g_i \land e_i \land t_i \land (\tilde{V}_i = \tilde{V}_i')$, where $(\tilde{V}_i = \tilde{V}_i')$ denotes that the local variables of $B_{i-1}$ are unchanged.

- channel communication: if $g_i \land e_i \land t_i$ is a transition in $Out_i$; and $g_{1-i} \land e_{1-i} \land t_{1-i}$ is a transition in $In_{1-i}$; and $e_i$ and $e_{1-i}$ are matching channel input/output, $Trans$ contains a transition $g_i \land g_{1-i} \land e_i \land t_i \land t_{1-i}$. \[4\]

- barrier synchronization: if $g_i \land e_i \land t_i$ is a transition in $Trans_i$ and $g_{1-i} \land e_{1-i} \land t_{1-i}$ 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 $g_i \land g_{1-i} \land e_i \land t_i \land t_{1-i}$.

$Out/In$ contains a transition $g_i \land e_i \land t_i \land (\tilde{V}_{i-1} = \tilde{V}_{i-1}')$ if $g_i \land e_i \land t_i$ 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 $g_i \land g_{1-i} \land \text{tick} \land t_i \land t_{1-i}$ if $g_i \land \text{tick} \land t_i$ is a transition in $Tick_i$ and $g_{1-i} \land \text{tick} \land t_{1-i}$ 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 $M_0$ and $M_1$ is a TFSM $M = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1$; $S = S_0 \times S_1$; $init = (\text{init}_0, \text{init}_1)$; $Act = Act_0 \cup Act_1$; $Ch = Ch_0 \cup Ch_1$; $T$ is the minimum transition relation such that for any $(s_0, [g_0]e_0\{\text{prog}_0\}, s_0') \in T_0$: $(s_1, [g_1]e_1\{\text{prog}_1\}, s_1') \in T_1$, $s_0 \neq s_1$.

- if $e_0 \neq \text{tick}$, ((s_0, s_1), [g_0]e_0\{\text{prog}_0\}, (s_0', s_1)) \in T;
- if $e_1 \neq \text{tick}$, ((s_0, s_1), [g_1]e_1\{\text{prog}_1\}, (s_0', s_1')) \in T;
- if $e_0 = \text{ch?v}$ is an output on channel $ch$ with value $v$; and $e_1 = \text{ch?v}$ is a matching channel input, ((s_0, s_1), [g_0 \land g_1]ch.v\{\text{prog}_0\} \land \{\text{prog}_1\}, (s_0', s_1')) \in T;
- if $e_1 = \text{ch?v}$ is a channel output; and $e_0 = \text{ch?x}$ is a matching channel input, ((s_0, s_1), [g_0 \land g_1]ch.v\{\text{prog}_0\} \land \{\text{prog}_1\}, (s_0', s_1')) \in T;
- if $e_0 = \text{ch?v} \land e_1 \land \text{cond}$ is a transition in $Trans_i$; and $e_0 \neq \text{ch?x}$ is a matching channel input, ((s_0, s_1), [g_0 \land g_1]e_0\{\text{prog}_0\} \land \{\text{prog}_1\}, (s_0', s_1')) \in T$.

In our encoding, matching synchronous input/output is labeled with the same event.
Let $\mathcal{B} = (\overline{V}, \overline{v}, Init, Trans, Out, In, Tick)$ be the BDD machine encoding of the interleave composition of two components $\mathcal{M}_0$ and $\mathcal{M}_1$ such that $\overline{v} = \overline{v}_0 \cup \overline{v}_1$; $Init = Init_0 \land Init_1$. $Trans$ contains two kinds of transitions.

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

$In$ contains following transitions:

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

$Out$ contains following transitions:

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

$Tick$ contains following transitions:

- $(g_0 \land e_0 \land t_0) \land (g_1 \land e_1 \land t_1)$ where $g_0 \land e_0 \land t_0$, and $g_1 \land e_1 \land t_1$ 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 $\checkmark$ 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 $\mathcal{M} = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 \cup LV_1$; $S = S_0 \cup S_1$; $init = init_0$; $Act = Act_0 \cup Act_1$; $Ch = Ch_0 \cup Ch_1$; $T$ 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 $e_0 \neq \checkmark$, $(s_0, [g_0]e_0\{prog_0\}, s'_0) \in T$;
- $(s_1, [g_1]e_1\{prog_1\}, s'_1) \in T$;
- if $e_0 = \checkmark$, $(s_0, [g_0]e_0\{prog_0\}, init_0) \in T$.

The BDD machine of $\mathcal{M}_0$; $\mathcal{M}_1$ is $(\overline{V}, \overline{v}, Init, Trans, Out, In)$ such that $\overline{v} = \overline{v}_0 \cup \overline{v}_1 \cup \{terminated\}$ where $terminated$ is a fresh Boolean variable to manage whether $\mathcal{M}_0$ terminates; $Init = Init_0 \land \neg terminated$. $Trans$ contains following transitions

- $\neg terminated \land g_0 \land e_0 \land t_0 \land event \neq \checkmark \land \neg terminated'$ where $g_0 \land e_0 \land t_0$ is a transition in $Trans_0$
- $\neg terminated \land g_0 \land event = \tau \land t_0 \land terminated' \land Init_1$ where $g_0 \land e_0 \land t_0$ is a transition in $Trans_0$. Note that as the composition, we replace the $\checkmark$ action of $\mathcal{M}_0$ with the internal action $\tau$.
- $terminated \land g_1 \land e_1 \land t_1 \land terminated'$ where $g_1 \land e_1 \land t_1$ is a transition in $Trans_1$

$In$ contains following transitions
\[ \neg \text{terminated} \land g_0 \land e_0 \land t_0 \land \neg \text{terminated}' \text{ where } g_0 \land e_0 \land t_0 \text{ is a transition in } \text{In}_0 \]

\[ \text{terminated} \land g_1 \land e_1 \land t_1 \land \text{terminated}' \text{ where } g_1 \land e_1 \land t_1 \text{ is a transition in } \text{In}_1 \]

\text{Out} \text{ contains following transitions}

\[ \neg \text{terminated} \land g_0 \land e_0 \land t_0 \land \neg \text{terminated}' \text{ where } g_0 \land e_0 \land t_0 \text{ is a transition in } \text{Out}_0 \]

\[ \text{terminated} \land g_1 \land e_1 \land t_1 \land \text{terminated}' \text{ where } g_1 \land e_1 \land t_1 \text{ is a transition in } \text{Out}_1 \]

\text{Tick} \text{ contains following transitions}

\[ \neg \text{terminated} \land g_0 \land e_0 \land t_0 \land \neg \text{terminated}' \text{ where } g_0 \land e_0 \land t_0 \text{ is a transition in } \text{Tick}_0 \]

\[ \text{terminated} \land g_1 \land e_1 \land t_1 \land \text{terminated}' \text{ where } g_1 \land e_1 \land t_1 \text{ is a transition in } \text{Tick}_1 \]

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

- if \( e_0 \neq \text{tick} \), \((s_0, s_1), [g_0] e_0 \{ \text{prog}_0 \}, (s'_0, s_1)) \in T; \]
- if \( e_1 \neq \text{tick} \), \((s_0, s_1), [g_1] e_1 \{ \text{prog}_1 \}, (\text{done}, s'_1)) \in T; \]
- \((s_0, s_1), [g_0 \land g_1] e_0 \{ \text{prog}_0 \land \text{prog}_1 \}, (s'_0, s'_1)) \in T \text{ if } e_0 = e_1 = \text{tick} \]

A BDD machine of \( \mathcal{M}_0 \Delta \mathcal{M}_1 \) is \((\overline{V}, \overline{V}, \text{Init}, \text{Trans}, \text{Out}, \text{In})\) such that \( \overline{V} = \overline{V}_0 \cup \overline{V}_1 \cup \{ \text{interrupted} \} \) where \( \text{interrupted} \) is a fresh Boolean variable to manage whether \( \mathcal{M}_1 \) interrupts \( \mathcal{M}_0 \); \( \text{Init} = \text{Init}_0 \land \text{Init}_1 \lor \neg \text{interrupted} \). \text{Trans} contains following transitions

\[ \neg \text{interrupted} \land g_0 \land e_0 \land t_0 \land \neg \text{interrupted}' \land (\overline{V}_1 = \overline{V}'_1) \text{ where } g_0 \land e_0 \land t_0 \text{ is a transition in } \text{Trans}_0 \]

\[ g_1 \land e_1 \land t_1 \land \text{interrupted}' \text{ where } g_1 \land e_1 \land t_1 \text{ is a transition in } \text{Trans}_1 \]

\text{In} \text{ contains following transitions}

\[ \neg \text{interrupted} \land g_0 \land e_0 \land t_0 \land \neg \text{interrupted}' \land (\overline{V}_1 = \overline{V}'_1) \text{ where } g_0 \land e_0 \land t_0 \text{ is a transition in } \text{In}_0 \]

\[ g_1 \land e_1 \land t_1 \land \text{interrupted}' \text{ where } g_1 \land e_1 \land t_1 \text{ is a transition in } \text{In}_1 \]

\text{Out} \text{ contains following transitions}
The TFSM of output cannot occur on its own but must be synchronized with a corresponding channel output.

Channel Out

Let $a$ be a channel name of type $T$, and $\text{exps}$ is a particular value of type $T$, then then channel output $a?\text{exps} \rightarrow M_0$ describes a TFSM which can output $\text{exps}$ along channel $a$ and subsequently behaves as $M_0$. However this output can not occur on its own but must be synchronized with a corresponding channel input from other components. The TFSM of $a?\text{exps} \rightarrow M_0$ is a new TFSM $M = (GV, LV, S, \text{init}, \text{Act}, Ch, T)$ such that $LV = LV_0 \cup LV_1; S = S_0 \cup \{\text{init}\}; \text{Act} = \text{Act}_0 \cup \{a?\text{exps}\}; Ch = Ch_0 \cup \{a\}; T = \{(\text{init}, a?\text{exps}, \text{init}_0), (\text{init}, \text{tick}, \text{init})\} \cup T_0; \text{ and } Ch = Ch_0.$

Let $B = (V, \overline{V}, \text{Init}, \text{Trans}, \text{Out}, \text{In}, \text{Tick})$ be the BDD machine encoding of $a?\text{exps} \rightarrow M_0$. We have $\overline{V} = \overline{V}_0 \cup \{\text{done}\}; \text{Init} = \neg \text{done}$. $\text{Trans}$ contains the following transitions

- $\neg \text{interrupted} \land g_0 \land e_0 \land t_0 \land \neg \text{interrupted}'$ where $g_0 \land e_0 \land t_0$ is a transition in $\text{Trans}_0$

$\text{In}$ contains the following transitions

- $\neg \text{interrupted} \land (g_0 \land e_0 \land t_0) \land (g_1 \land e_1 \land t_1) \land \neg \text{interrupted}'$ where $g_0 \land e_0 \land t_0$ is a transition in $\text{Tick}_0$, and $g_1 \land e_1 \land t_1$ is a transition in $\text{Tick}_3$
- $\neg \text{interrupted} \land g_1 \land e_1 \land t_1 \land \text{interrupted}'$ where $g_1 \land e_1 \land t_1$ is a transition in $\text{Tick}_3$

$\text{Tick}$ contains the following transitions

- $\neg \text{interrupted} \land \{\text{exps}\}$ where $\text{exps}$ is a channel input from other components. The TFSM of output cannot occur on its own but must be synchronized with a corresponding channel output.

Channel Out

$\text{Out}$ contains the following transitions

- $\neg \text{done} \land (\text{count}_a < L) \land [\overline{\text{A}}_{i=1..\text{exps}.\text{count}}(a[top_a][i]) = \text{exps}[i]) \land (\text{size}_a[top_a]' = \text{exps}.\text{count}) \land (\text{count}_a' = \text{count}_a + 1) \land top_a = (top_a + 1)\%L \land \text{done}' \land \text{Init}_0$

where $\text{count}_a$ 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 $\text{size}_a$ is an array to manage the number of the messages in the buffer. The guard of the channel out transition includes $\text{done}$ is false, and the channel buffer is not full. After the channel in transition, elements from the expression $\text{exps}$ is pushed to the buffer. The size of the expression is also updated to $\text{size}_a[top_a]$. Moreover the channel buffer updates its size $\text{count}_a$ and tail position $top_a$. $\text{done}$ is set false to constrain the channel out transition to happen once and then pass the control to the process $P_0$.

- $\neg \text{done} \land \text{event} = \text{tick} \land \neg \text{done}'$

$\text{Tick}$ contains the following transitions

- $\neg \text{done} \land \text{event} = \text{tick} \land \neg \text{done}'$
- $\text{done} \land g_0 \land e_0 \land t_0 \land \text{done}'$ where $g_0 \land e_0 \land t_0$ is a transition in $\text{Tick}_0$
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 \( \text{alexps} \rightarrow \mathcal{M}_0 \) is a new TFSM \( \mathcal{M} = (GV, LV, S, \text{init}, Act, Ch, T) \) such that \( LV = LV_0 \cup LV_1; S = S_0 \cup \{ \text{init} \}; Act = Act_0 \cup \{ \text{alexps} \}; Ch = Ch_0; T = \{ (\text{init}, \text{alexps}, \text{init}_0), (\text{init}, \text{tick}, \text{init}) \} \cup T_0; \) and \( Ch = Ch_0 \cup \{ a \} \).

Let \( \mathcal{B} = (\overline{\nu}, \overline{\psi}, \text{Init}, \text{Trans}, \text{Out}, \text{In}, \text{Tick}) \) be the BDD machine encoding of \( [b] \text{alexps} \rightarrow P_1 \). We have \( \overline{\psi} = \overline{\nu} \cup \{ \text{done} \}; \text{Init} = \neg \text{done} \). Trans contains the following transitions

- \( \text{done} \land g_0 \land e_0 \land t_0 \land \text{done}' \) where \( g_0 \land e_0 \land t_0 \) is a transition in \( \text{Trans}_0 \)

In contains the following transitions

- \( \neg \text{done} \land b \land (\text{count}_a > 0) \land (\text{size}_a[(\text{top}_a - \text{count}_a)\%L] = \text{alexps}.\text{count}) \land \bigwedge_{i=1}^{\text{alexps}.\text{count}}(\text{alexps}[i]' = a[(\text{top}_a - \text{count}_a)\%L[i]]) \land (\text{count}_a' = \text{count}_a - 1) \land \text{done}' \land \text{Init}_0 \). The guard of the channel in transition includes \( \text{done} \) 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.

- \( \text{done} \land g_0 \land e_0 \land t_0 \land \text{done}' \) where \( g_0 \land e_0 \land t_0 \) is a transition in \( \text{In}_0 \)

Out contains the following transitions

- \( \text{done} \land g_0 \land e_0 \land t_0 \land \text{done}' \) where \( g_0 \land e_0 \land t_0 \) is a transition in \( \text{Out}_0 \)

Tick contains the following transitions

- \( \neg \text{done} \land \text{event} = \text{tick} \land \neg \text{done}' \)
- \( \text{done} \land g_0 \land e_0 \land t_0 \land \text{done}' \) where \( g_0 \land e_0 \land t_0 \) is a transition in \( \text{Tick}_0 \)

**Time Function**

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

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

Because the \( \text{Wait}[t] \) is a simple TFSM which is not composed by other TFSM, the BDD machine encoding of \( \text{Wait}[t] \) 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 $1 \leq i \leq t - 1$
- $(state_1, \tau, init_0)$. 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 $\mathcal{B} = (\bar{V}, \bar{\tau}, Init, Trans, Out, In, Tick)$ be the BDD machine encoding of $P_0.timeout[t]P_1$. We have $\bar{\tau} = \bar{\tau}_0 \cup \bar{\tau}_1 \cup \{clk\}$ 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 $Init = Init_0 \land clk = 0$; and $Trans$ contains following transitions:

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

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

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

$Out$ contains following channel transitions:

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

Tick includes below transitions.

- $g_0 \land e_0 \land t_0 \land (clk \geq 0 \land clk < t \land clk' = clk + 1) \lor (clk = -1 \land clk' = -1)$ where $g_0 \land e_0 \land t_0$ is a transition in $Tick_0$
- $clk = t + 1 \land g_1 \land e_1 \land t_1 \land clk' = t + 1$ where $g_1 \land e_1 \land t_1$ is a transition in $Tick_1$
**Time Interrupt** The TFSM $M_0\text{interrupt}\[t\]M_0$ behaves as $P_0$ until $t$ time units elapse and then switches to $M_1$. There is no need for $M_1$ to execute concurrently with $M_0$ 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 occur at any time. Therefore for the time interrupt, a new variable is presented. The time interrupt of $M_0\text{interrupt}\[t\]M_1$ is a TFSM $M = (GV, LV, S, init, Act, Ch, T)$ such that $LV = LV_0 ∪ LV_1 ∪ \{clk\}$ where $clk ∈ \{-1..t + 1\}$ is a new variable to count the time passage which is initialized with 0; $clk = -1$ to discard $M_1$ after $M_0$ terminates and $clk = t + 1$ to discard $M_0$ after $M_1$ interrupts, $S = S_0 ∪ S_1$; $init = init_0$; $Act = Act_0 ∪ Act_1$; $Ch = Ch_0 ∪ Ch_1$; and $T$ is the minimum transition relation defined as follows. For any $(s_0, [g_0])e_0\{progo\}, s'_0) ∈ T_0$; any $(s_1, [g_1])e_1\{prog1\}, s'_1) ∈ T_1$, $T$ contains below transitions:

- $(s_0, [g_0])e_0\{progo\}, s'_0)$ if $e_0 ≠ tick ∧ e_0 ≠ ✓$
- $(s_0, [g_0])e_0\{progo\}; clk = -1, s'_0)$ if $e_0 ≠ tick ∧ e_0 = ✓$
- $(s_0, [g_0])e_0\{progo\}; \{0 ≤ clk < t\}clk + + else\{clk = -1\}clk = -1, s'_0)$ if $e_0 = tick$
- $(t, [clk = t]τ\{clk = t + 1\}, init)$ for all $s ∈ S_0$
- $(s_1, [g_1] ∧ clk = t + 1)e_1\{prog1\}, s'_1)$

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

- $clk = t ∧ g_0 ∧ e_0 ∧ t_0 ∧ [(event = ✓ ∧ clk' = -1) ∨ (event ≠ ✓ ∧ clk' = clk)]$ where $g_0 ∧ e_0 ∧ t_0$ is a transition in $Trans_0$
- $clk = t ∧ event = τ ∧ clk' = t + 1 ∧ Init'_1$
- $clk = t + 1 ∧ g_1 ∧ e_1 ∧ t_1 ∧ clk' = t + 1$ where $g_1 ∧ e_1 ∧ t_1$ is a transition in $Trans_1$

$In$ contains following transitions:

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

$Out$ contains following transitions:

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

$Tick$ includes below transitions:

- $g_0 ∧ e_0 ∧ t_0 ∧ [(0 ≤ clk < t ∧ clk' = clk + 1) ∨ (clk = -1 ∧ clk' = -1)]$ where $g_0 ∧ e_0 ∧ t_0$ is a transition in $Tick_0$
- $clk = t + 1 ∧ g_1 ∧ e_1 ∧ t_1 ∧ clk' = t + 1$ where $g_1 ∧ e_1 ∧ t_1$ 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 $LV = LV_0; S = S_1 \times \{0, 1, \ldots, d\}$ 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 $d_1 \in \{0, 1, \ldots, d\}$.
- for any $(s, [g]tick\{prog\}, s') \in T_0, ((s, d_1), [g]tick\{prog\}, (s', d_1 + 1)) \in T$ for all $d_1 \in \{0, 1, \ldots, d - 1\}$.

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

- $clk \leq t \land g_0 \land e_0 \land t_0 \land \{(event \neq \checkmark \land clk' = clk) \lor (event = \checkmark \land clk' = -1)\}$ where $g_0 \land e_0 \land t_0$ is a transition in $Trans_0$

$In$ includes below transitions:

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

$Out$ includes below transitions:

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

$Tick$ includes below transitions:

- $\lfloor 0 \leq clk < t \land clk' = clk + 1 \lor (clk = -1 \land clk' = -1)\rfloor \land g_0 \land e_0 \land t_0$

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 $s \in init_0 \cup \{state_1, \ldots, state_t\}$.

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 $B = (\overline{V}, \overline{v}, Init, Trans, Out, In, Tick)$ be the BDD machine encoding of $\mathcal{M}_0\, within[t]$ where $\overline{v} = \overline{v}_0 \cup \{clk\}, -1 \leq clk \leq t$ records the number elapsed time units so far and $clk = -1$ indicates an visible action just happens and $Init = Init_0 \land clk = 0$; and $Trans$ includes below transitions
- \( \text{clk} \leq t \land g_0 \land e_0 \land t_0 \land [(\text{event} \neq \tau \land \text{clk}' = -1) \lor (\text{event} = \tau \land \text{clk}' = \text{clk})] \) where \( g_0 \land e_0 \land t_0 \) is a transition in \( \text{Trans}_0 \)

**In** includes below transitions

- \( \text{clk} < t \land g_0 \land e_0 \land t_0 \land \text{clk}' = -1 \) where \( g_0 \land e_0 \land t_0 \) is a transition in \( \text{In}_0 \)

**Out** includes below transitions

- \( \text{clk} < t \land g_0 \land e_0 \land t_0 \land \text{clk}' = -1 \) where \( g_0 \land e_0 \land t_0 \) is a transition in \( \text{Out}_0 \)

**Tick** includes below transitions:

- \( g_0 \land e_0 \land t_0 \land [((\text{clk} \geq 0 \land \text{clk} < t \land \text{clk}' = \text{clk}+1) \lor (\text{clk} = -1 \land \text{clk}' = -1)] \)

where \( g_0 \land e_0 \land t_0 \) is a transition in \( \text{Tick}_0 \)

### 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 \( e \rightarrow M_0 \) 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 \( M_0 \) 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.