

# The Formal Verification of a Reintegration Protocol

Lee Pike

Formal Methods Group  
NASA Langley Research Center  
lee.s.pike@nasa.gov

February 8, 2005

(The contents herein are not necessarily endorsed by the United States  
Government.)

# Acknowledgments

- ▶ The reintegration protocol was developed by Wilfredo Torres-Pomales, Mahyar Malekpour, and Paul Miner.
- ▶ Wilfredo significantly helped me understand the protocol's behavior and requirements.
- ▶ Bruno Dutertre and Leonardo de Moura provided many helpful suggestions about Timeout Automata and SAL.

# Timeout Automata<sup>1</sup> (Semantics)

- ▶ A set of state variables  $V$ .
- ▶ A *global clock*,  $c \in \mathbb{R}^{0\leq}$ .
- ▶ A set of *timeout* variables  $T$  such that for  $t \in T$ ,  $t \in \mathbb{R}^{0\leq}$ .

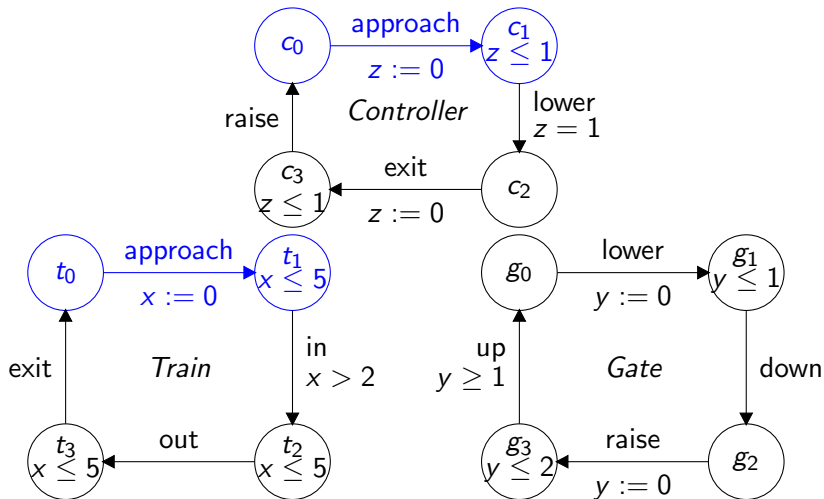
Construct a transition system  $\langle S, S^0, \rightarrow \rangle$ :

- ▶ States are mappings of all variables to values.
- ▶ Transitions are either *time transitions* or *discrete transitions*.
  - ▶ Time transitions are enabled if the clock is less than all timeouts. Updates clock to least timeout.
  - ▶ Discrete transitions are enabled if the clock equals some timeout. Updates state variables and timeouts.

---

<sup>1</sup>B. Dutertre and M. Sorea. "Timed Systems in SAL," 2004. 

# The Train-Gate-Controller



# TGC SAL (Pseudo)-Specs: From Asynchrony to Synchrony

## Asynchronous Composition

```
train: MODULE =  
    t_state = t0  
    AND t_to = time  
-->  
    t_state' = t1;  
    flag1' = TRUE;  
    msg1' = approach;
```

```
controller: MODULE =  
    c_state = c0  
    AND flag1 = TRUE  
    AND msg1 = approach  
-->  
    c_state' = c1;  
    flag1' = FALSE;
```

# TGC SAL (Pseudo)-Specs: From Asynchrony to Synchrony

## Asynchronous Composition

```
train: MODULE =
    t_state = t0
    AND t_to = time
-->
    t_state' = t1;
    flag1' = TRUE;
    msg1' = approach;

controller: MODULE =
    c_state = c0
    AND flag1 = TRUE
    AND msg1 = approach
-->
    c_state' = c1;
    flag1' = FALSE;
```

## Synchronous Composition

```
train: MODULE =
    t_state = t0
    AND t_to = time
    AND c_state = c0
-->
    t_state' = t1;
    msg1' = approach;

controller: MODULE =
    c_state = c0
    AND t_to = time
    AND msg1' = approach
-->
    c_state' = c1;
```

# TGC SAL (Pseudo)-Specs: From Synchrony to Clockless

## Clocked Semantics

```
train: MODULE =  
    t_state = t0  
    AND t_to = time  
    AND c_state = c0  
-->  
    t_state' = t1;  
    msg1' = approach;
```

```
controller: MODULE =  
    c_state = c0  
    AND t_to = time  
    AND msg1' = approach  
-->  
    c_state' = c1;
```

# TGC SAL (Pseudo)-Specs: From Synchrony to Clockless

## Clocked Semantics

```
train: MODULE =
    t_state = t0
    AND t_to = time
    AND c_state = c0
-->
    t_state' = t1;
    msg1' = approach;

controller: MODULE =
    c_state = c0
    AND t_to = time
    AND msg1' = approach
-->
    c_state' = c1;
```

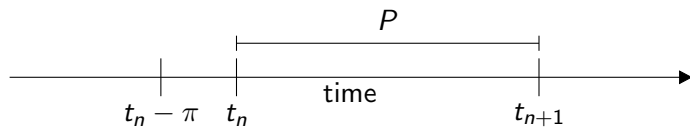
## Clockless Semantics

```
train: MODULE =
    t_state = t0
    AND t_to = min(t_to, c_to, g_to)
    AND c_state = c0
-->
    t_state' = t1;
    msg1' = approach;

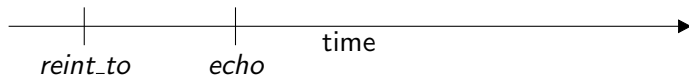
controller: MODULE =
    c_state = c0
    AND t_to = min(t_to, c_to, g_to)
    AND msg1' = approach
-->
    c_state' = c1;
```



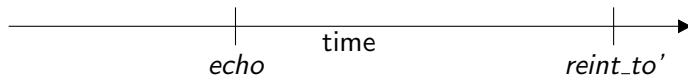
# The Frame Property



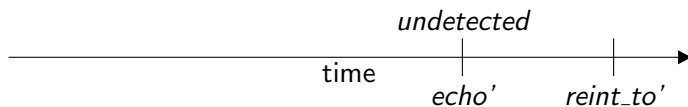
# The Peril of Time-Triggered Simulation



# The Peril of Time-Triggered Simulation



# The Peril of Time-Triggered Simulation



# Safety Properties

## Theorem (No Operational Accusations)

*For all operational nodes  $i$ ,  $accs[i]$  does not hold during the reintegration protocol.*

## Theorem (Synchronization Acquisition)

*For all operational nodes  $i$ ,  $|clock - echo(i)| < \pi$  upon termination of the reintegration protocol.*

## State Variables & Initialization

- ▶ *accs*: ARRAY of booleans, one for each monitored node.
- ▶ *seen*: ARRAY of naturals, one for each monitored node.
- ▶ *mode*:  $\{prelim\_diag, frame\_synch, synch\_capture\}$
- ▶ *clock*:  $\mathbb{R}^{0\leq}$ .
- ▶ *fs\_finish*:  $\mathbb{R}^{0\leq}$ .
- ▶ *pd\_finish*:  $\mathbb{R}^{0\leq}$ .

```

for each i, accs[i] := false;
mode := prelim_diag;
for each i, seen[i] := 0;
  
```

# Preliminary Diagnosis Mode

```
pd_finish := clock + P +  $\pi$ ;  
while clock < pd_finish do {  
  for each i, when echo(i) do {  
    if (seen[i] < 2 and not accs[i])  
      then seen[i] := seen[i] + 1  
      else accs[i] := true;  
  };  
};  
for each i, if seen[i] = 0 then accs[i];  
mode := frame_synch;
```

# Frame Synchronization Mode

```

for each  $i$ ,  $seen[i] := 0$ ;
 $fs\_finish := clock$ ;
while  $clock - fs\_finish < \pi$  do {
  for each  $i$ , when  $echo(i)$  do {
    if ( $seen[i] = 0$  and not  $accs[i]$ )
    then {
       $fs\_finish := clock$ ;
       $seen[i] := seen[i] + 1$ ;
    };
    else  $accs[i] := true$ ;
  };
};
 $mode := synch\_capture$ ;

```



# Synchronization Capture Mode

```
for each  $i$ ,  $seen[i] := 0$ ;  
while  $seen\_cnt \leq trusted/2$  do {  
  for each  $i$ , when  $echo(i)$  do {  
    if ( $seen[i] = 0$  and not  $accs[i]$ )  
      then  $seen[i] := seen[i] + 1$ ;  
  };  
};  
 $clock := 0$ ;
```