To ensure that all process clocks remain bounded, we can associate an idling transition with all terminating locations, and periodically reset the clocks also.

We must ensure that $T$ remains bounded. Either omit altogether, or let it increase modulo some constant. Either $T$ remains bounded. We need ensure that $T$ remains bounded.

are necessary:

For the case that the time domain is the natural numbers and all the discrete CTS’s.

Model Checking Discrete CTS’s

A. Pnueli
The idling and tick transitions are implemented by separate modules. Note that the idling module 

\[ t \in [0, 20) \], \quad t \in [0, 6) \], \quad t \in [0, 6) \], \quad y \in [3, 5) \], \quad x \in \text{boolean} \]

The global progress condition \( \text{Prog} \) is obtained as the conjunction of the local progress conditions. The idling and tick transitions are implemented by separate modules. Note that the idling module defines:

\[
\text{Prog} = P_1.\text{Prog} \land P_2.\text{Prog}.
\]

The idling and tick transitions are implemented by separate modules. Note that the idling module defines: 

\[
\text{Prog} = P_1.\text{Prog} \land P_2.\text{Prog}.
\]

The idling and tick transitions are implemented by separate modules. Note that the idling module defines:

\[
\text{Prog} = P_1.\text{Prog} \land P_2.\text{Prog}.
\]
This module represents process $P_1$ of program ANY-Y.

Module $T$

```
ASSIGN
VAR
loc: 0..2;
prog: t > high;
 MODULE $T$ (x, y, t, low, high)

MODULE $L$ (x, y, t, low, high)

DEFINE prog := t < high;
VAR loc: 0..2;
ASSIGN
init(loc) := 0;
init(y) := 0;
init(t) := 0;
next(loc) := case
  t < low: loc;
  loc = 0 & !x: 1;
  loc = 0 & x: 2;
  loc = 1: 0;
  1: loc;
end;
next(y) := case
  loc = 1 & next(loc) = 0: (y + 1) mod 7;
  1: y;
end;
next(t) := case
  t >= low: 0;
  1: t;
end;
```
This module represents process $P_2$ of program ANY-Y.

A. Pnueli

Module $R$
This module represents the \texttt{tick} transition. Parameter \texttt{Prog} is true if all processes allow time to progress. \texttt{tick} transition always increments all clocks by 1, and the progress conditions are computed accordingly.

The master clock $T$ is advanced modulo 21. The \texttt{justice} requirements guarantee that the clocks will advance (by 1) infinitely many times.

\begin{verbatim}
MODULE Tick(Prog,t_1,t_2,T)
ASSIGN init(T):=0;
next(t_1):=case Prog: t_1+1; 1: t_1; esac;
next(t_2):=case Prog: t_2+1; 1: t_2; esac;
next(T):=case Prog: (T+1) mod 21; 1: T; esac;

JUSTICE T = 0, I = 0

\end{verbatim}
The two properties we model check for time bounds $3, 5$ are

$$
\forall x (\exists y > 21 \land (x.T < 15 \land x.P[1] > x.P[2] \land x.T < 0))
$$

The next example shows an interesting case where mutual exclusion is guaranteed due to the correct timing constraints imposed on the system.

The timing constraints of a property which does not mention time explicitly but is valid only due to the beginning of the execution. Note that the formula faithfully represents this property even though $T$ increments module 21.

Property 1 states that variable $y$ never rises above the value of $T$. This is an example of a property which mentions time explicitly but is valid only due to the timing constraints imposed on the system.

The Specication
Fischer's Mutual Exclusion Program

\[ 0 = \text{integer where } x \]

\[ \text{loop forever do} \]

A. Pnueli

Fischer's Mutual Exclusion Program

\[ 0 = x : 6m \]

\[ 0 = x : 6j \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]

\[ 0 = x \]
The algorithm is correct for any time bounds \([L, U]\) such that \(2L > U\). In our case, we run it with \(L = 3\) and \(U = 5\).
A.Pnueli

Module proc

MODULE proc(id,x,t,Low,High)

DEFINE prog := t >= Low

next (t) := case

I

next (t) := 0

! x :=

I

next (t) := 0

! toc = 9 & next (toc) = toc

t := 6

! toc :=

I

next (toc) := case

!

next (toc) := 0

! toc in {8} & toc in {7,9}

t := 0

! toc in {4,5} & toc in {3,9}

t := 0

! toc in {3,4} & toc in {6,9}

t := 1

! toc in {1,2} & toc in {5,6}

t := 1

! toc :=

I

next (toc) := case

!

next (toc) := 0

! init(t) := 0

! init(x) := 0

! init(loc) := 1

ASSIGN

VAR

DEFINE prog := t > High

MODULE proc(id,x,t,Low,High)
MODULE Tick (Prog, t_1, t_2, T)

ASSIGN

\[ i = 0 \]

\[ \text{INIT} T = 0, I = 0 \]

\[ \text{next} (I) = \text{case} \]

\[ \text{next} (I) = \text{case} \]

\[ \text{next} (I) = \text{case} \]

\[ \text{next} (I) = \text{case} \]

\[ \text{next} (I) = \text{case} \]

\[ \text{next} (I) = \text{case} \]

\[ \text{next} (I) = \text{case} \]

\[ \text{next} (I) = \text{case} \]

\[ \text{next} (I) = \text{case} \]

\[ \text{next} (I) = \text{case} \]

\[ T = 0, I \neq 0 \]

Timed and Hybrid Systems, NYU, Spring, 2007

138
The properties we wish to check for program Fischer are:

- **Mutual Exclusion**
  \[
  \neg (p[1].loc = 2 \land p[1].loc = 8) \land \neg (p[2].loc = 2 \land p[2].loc = 8)
  \]

- **Accessibility**
  \[
  \diamond \leftarrow 2 \land \Box (p[1].loc = 8) \land \Box (p[2].loc = 8)
  \]

Examples follow:

Running the program with \( T = 2 \) and \( \Omega = 5 \), both properties are invalid.

Running this program with \( T = 3 \) and \( \Omega = 5 \), both properties are valid.

The specification is:

\[
\begin{align*}
\& (p[1].loc = 1 \to p[2].loc = 1 \\
& \lor p[1].loc = 2 \to p[2].loc = 2 \\
& \lor p[1].loc = 8 \to p[2].loc = 8)
\end{align*}
\]
Counter Example to Mutual Exclusion

A. Pnueli

Model Checking Discrete CTS

Timed and Hybrid Systems, NYU, Spring 2007
Counter Example to Accessibility:

From $\mathcal{C}_3, m_3$ to $\mathcal{C}_5, m_5$
Lecture 8: Model Checking Discrete CTS

A. Pnueli

Counter Example to Accessibility:

From $\langle f_3, m_3, f_3 \rangle$ to $\langle f_3, m_3, f_3 \rangle$,
St. 42 = x: 0, t_1: 0, t_2: 2, T: 0, loc_1: 4, loc_2: 2
St. 41 = x: 0, t_1: 5, t_2: 2, T: 0, loc_1: 4, loc_2: 2, tick: 2
St. 40 = x: 0, t_1: 3, t_2: 0, T: 6, loc_1: 4, loc_2: 2, 4
St. 39 = x: 0, t_1: 3, t_2: 2, T: 6, loc_1: 4, loc_2: 2, 4
St. 38 = x: 0, t_1: 1, t_2: 0, T: 4, loc_1: 4, loc_2: 2, 4
St. 37 = x: 0, t_1: 1, t_2: 2, T: 4, loc_1: 4, loc_2: 2, 4

A Pnueli

Lecture 8: Model Checking Discrete CTS