Analysis of Software Artefacts

Model Checking Real Time Systems (Lecture 7)
Agenda

- Introduce quantitative analysis
- Introduce real time CTL or RTC
- Overview of model checking real time systems
Errors can have catastrophic effects

- Robots
- Industrial machinery
- Controllers for aircraft

Some applications are

to function correctly

Several applications require predictable response times

Why very real-time systems?
- Synchronization
  - Periodicity

Limitations on types of processes

Simple to use and provides useful information

Powerful tool for analyzing real-time systems

RMS (Rate monotonic scheduling)
Computing-Scheduling and Resource Management, real-time systems, in Foundations of Real-Time
Tokuda, H. Fixed priority scheduling theory for hard
• J.P. Te不失cky, I. Sha, J.K. Strosnider, and H.
IEEE Transactions on Software Engineering,
Timing analysis for fixed-priority hard real-time
• M.C. Harbour, M.H. Klein, and J.P. Te不失cky.

Some papers on RMS
for certain types of systems, does not scale

much harder than RMS

use real time version of CTL

no restrictions on system being specified

Real time model checking
We will only cover discrete time.

Analysis for dense time is very hard.

dense time uses integers to represent time.
dense time uses real numbers to represent time.

Dense versus discrete time.
only way to talk about time is using the $X$ operator •

will become true, but does not say when $f$ says that sometime in the future $\text{EF}(f)$ •

regular CTL has no notion of time •

---

Real time CTL (RTCTL)
Finishing a transaction is denoted by \( T \). finished

It always finishes in the next 3 cycles

\( T \). started, then

If in a state a transaction \( T \) starts (denoted by

---

Example
\[
\begin{align*}
\text{is true until } & b \text{ becomes true} \\
[q, q'] & \text{ becomes true somewhere in the time interval } b \\
& \quad \text{and for all } i, \forall i > \ell \\
& \quad \text{for some } q, \exists q \geq \forall a \ \text{if and only if} \\
& \quad b^{[q, q']} \cap f \\
& \quad \ldots \quad 0, s_1, s_2, \ldots \\
& \quad \text{consider a path } \not\exists \\
\end{align*}
\]
Points to notice
\[ q, a \] is true in the time interval \( f \) •

\( b \models \exists ! \ s \text{ such that } a \geq b \geq \exists ! \ s \text{ if and only if} \]

\[ q, a \text{ is true on a path } \not
\]

\( \cdots \not \}

\( \text{consider a path } \not
\]

---

\( \mathcal{C} \) path operator \([q, a]\)
\[
\begin{align*}
\text{Example revisited} & \\
\text{time cycles} & \\
\text{a transaction started always finishes with next three} & \\
\text{represent } \mathcal{F} \text{ in terms } \mathcal{H}^{[q',q]} & \\
\mathcal{A} \mathcal{C} \mathcal{G} \left[ \begin{array}{c}
\mathcal{T}. \text{started} \\
\mathcal{L}^{[0,3]} \text{finished}
\end{array} \right] & \rightarrow \mathcal{A \mathcal{F}}
\end{align*}
\]
\( \text{ACG} \text{[EF]} \text{[Restart]} \)

\( \text{ACG} \text{[AP]} \text{[DeviceEnable]} \)

\( \text{ACG} \text{[AP]} \text{[DeviceEnable]} \leftarrow \text{ACF} \text{[AP]} \text{[Ready]} \)

\( \text{EF} \text{[AP]} \text{[Started]} \lor \text{Ready} \)

---

Specification patterns revisited
operations

• Identify bottlenecks in your system, i.e., slow

• Extremely useful in fine-tuning the system

• Provide information on how much a system deviates

Quantitative timing analysis
(MIN(T. started, T. finished)

- return \( \infty \) if no such path exists

- to a state in final to a state between a state in start

- returns

Inputs: two sets of states, start and final

Minimum Delay Analysis
\text{MAX(} \text{start,} \text{ finished}) \bullet

- reaches \text{finish} - path from a state in \text{start} that never returns \infty if there is an infinite
  - to a state in \text{final} - longest path between a state in \text{start} and \text{final}

\text{Inputs: two sets of states, start and final}

\underline{Maximum Delay Analysis}
earlier measures strictly based on path length

condition is true on a path

counting measures how many times a given...
(MIN(T, started, T, finished, T, idle))

a path from start to finish

Output: minimum times condition cond is true alone

Input:

Minimum condition counting
\textbf{MAX(T\text{.}\text{started}, T\text{.}\text{finished}, T\text{.}\text{idle})} \bullet

\textit{a path from start to finish}

\textbf{output: maximum times condition \texttt{cond} is true alone}

\textbf{inputs} \bullet

\textit{a condition: \texttt{cond}}

\textit{set of final states: \texttt{final}}

\textit{set of starting states: \texttt{start}}

Maximum condition counting
we will now describe the round-robin policy

scheduler picks a process in state ready to run

\( \text{running} \quad \Rightarrow \quad \text{process is running} \)

\( \text{ready} \quad \Rightarrow \quad \text{process is ready to run} \)

\( \text{idle} \quad \Rightarrow \quad \text{process is not doing anything} \)

each process has the following four states

\[ P_0, P_1, \ldots, P_{u-1} \]

assume that there are \( n \) processes

---

Round robin scheduling
\( n \mod (1 - n + \text{last}) \text{mod } n \), \( \text{last} \text{mod } 1 \) 

- Scheduler scans the processes in the following order:
- Some processes are ready to run
- Initial value of \( \text{last} \) is 0
- Keep a variable \( \text{last} \)

---

Round robin scheduling
set variable last to $i$

let the process that is picked be $P$ and is in the state ready and schedule it

pick the first process that is scanned

Round robin scheduling
is it possible that a process is in the
state ready and never gets to run?

Why is this claim not possible?

Is Round Robin Scheduling Fair?