Verification of Real Time Systems - CS5270
8th lecture

Hugh Anderson

National University of Singapore
School of Computing

March, 2007
Duckburg...
Outline

1. Administration
   - Assignment 2
   - The road map...

2. Efficiency in TTS
   - From regions to zones
   - Matrix notation and zone operations
   - Closed zones and graph representation

3. Preliminaries to Model Checking
   - Behaviour, safety, liveness, automata, reachability...
   - Extensional and intensional logic
   - Linear and branching time
Outline

1. Administration
   - Assignment 2
   - The road map...

2. Efficiency in TTS
   - From regions to zones
   - Matrix notation and zone operations
   - Closed zones and graph representation

3. Preliminaries to Model Checking
   - Behaviour, safety, liveness, automata, reachability...
   - Extensional and intensional logic
   - Linear and branching time
Outline

1. Administration
   - Assignment 2
   - The road map...

2. Efficiency in TTS
   - From regions to zones
   - Matrix notation and zone operations
   - Closed zones and graph representation

3. Preliminaries to Model Checking
   - Behaviour, safety, liveness, automata, reachability...
   - Extensional and intensional logic
   - Linear and branching time
Outline

1. Administration
   - Assignment 2
   - The road map...

2. Efficiency in TTS
   - From regions to zones
   - Matrix notation and zone operations
   - Closed zones and graph representation

3. Preliminaries to Model Checking
   - Behaviour, safety, liveness, automata, reachability...
   - Extensional and intensional logic
   - Linear and branching time
A reminder... Assignment number 2:

- On the web site
- Due on 22nd March...
The reduction...

What we did...

- **TTS**
  - State: $(s)$
  - Transition: $a; x \xrightarrow{g}$

- **TS**
  - State: $(s, V)$
  - Transition: $\delta \xrightarrow{a}$

- **TA**
  - State: $(s, V)$
  - Transition: $a \xrightarrow{a}$

- **QTS**
  - State: $(s, [])$
  - Transition: $a \xrightarrow{a}$
The immediate road map

The topics:

- **TTS: Timed transition systems**
  - Reduction: $\text{TTS} \rightarrow \text{TS}_{\text{TTS}} \rightarrow \text{TA}_{\text{TTS}} \rightarrow \text{RTS}$ (by quotienting)

- **Efficiency in TTS**
  - Regions

- **Zones**
  - Notation
  - Operations
  - Optimizations

- **Preliminaries for Model Checking**
  - Behaviour, safety, liveness, automata, reachability
  - Temporal logic
Outline

1 Administration
   - Assignment 2
   - The road map...

2 Efficiency in TTS
   - From regions to zones
   - Matrix notation and zone operations
   - Closed zones and graph representation

3 Preliminaries to Model Checking
   - Behaviour, safety, liveness, automata, reachability...
   - Extensional and intensional logic
   - Linear and branching time
What is wrong with regions?

Unwieldy:

- The number of regions can be very large:
  - It is exponential in the number of clocks, and in the size of the maximal constraints appearing in the clock constraints.
  - As a result, practical verification of transition systems based on regional transition systems becomes infeasible.
What is a zone?

A more compact representation:

- ...of equivalence classes of valuations....
  - Can be efficiently represented as Difference Bounded Matrices (edge weighted directed graphs).
  - DBMs admit a canonical representation.
  - DBMs can be manipulated efficiently.
Regions versus zones

47 regions versus 1 zone!

47 regions in zone \((2 \leq x \leq 5) \land (2 \leq y \leq 4)\)
Formally:

**Definition of zone:**

A zone $\mathcal{Z}$ is a clock constraint of the “two-variable difference” form

$$\mathcal{Z} ::= x \ op \ c \ | \ x - y \ op \ c \ | \ z_1 \land z_2$$

where $\ op \in \{<, \leq, >, \geq\}$, and $c \in \mathbb{N}$. 

Hugh Anderson

Verification of Real Time Systems - CS5270 8th lecture
What is this?

- A zone $Z$ is a convex union (or **hull**) of all the regions $R$:
  $Z = \bigcup_i R_i$.

- To encode zones in a DBM, we
  - construct a new clock variable $x_0$ which will always have the value 0, and then encode all constraints as $x_i - x_j < m$ or $x_i - x_j \leq m$ where $m \in \mathbb{Z}$.
  - For example the following terms on the left are translated to those on the right:

$$
\begin{align*}
  x_2 & < 3 & \implies & & x_2 - x_0 & < 3 \\
  x_5 & \geq 7 & \implies & & x_0 - x_5 & \leq -7 \\
  x_2 - x_5 & > 8 & \implies & & x_5 - x_2 & < -8
\end{align*}
$$
Finiteness and hence termination

Ignore constraints bigger than $C_x$:

- To ensure **termination**:
  - Remove constraints of the form $x < m$, $x \leq m$, $x - y < m$ and $x - y \leq m$ if $m > C_x$.
  - Replace $x > m$, $x \geq m$ with $x > C_x$ if $m > C_x$.
  - Replace $y - x > m$, $y - x \geq m$ with $y - x > C_x$ and $y - x \geq C_x$ if $m > C_x$. 
Outline

1. Administration
   - Assignment 2
   - The road map...

2. Efficiency in TTS
   - From regions to zones
   - Matrix notation and zone operations
   - Closed zones and graph representation

3. Preliminaries to Model Checking
   - Behaviour, safety, liveness, automata, reachability...
   - Extensional and intensional logic
   - Linear and branching time
Compact notation:

For \( n - 1 \) clock variables, we then write out an \( n \times n \) matrix \( M \), with elements drawn from \((\mathbb{Z} \times \{<, \leq\}) \cup \infty\) according to the following rules:

- For constraints like \( x_i - x_j < c \), set \( M_{i,j} = (c, <) \)
- For constraints like \( x_i - x_j \leq c \), set \( M_{i,j} = (c, \leq) \)
- Otherwise set \( M_{i,j} = \infty \)
Consider this clock zone:

\((0 \leq x_1 < 1) \land (0 < x_2 < 3) \land (x_2 - x_1 \geq 1)\)

then the DBM is

<table>
<thead>
<tr>
<th></th>
<th>(x_0)</th>
<th>(x_1)</th>
<th>(x_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_0)</td>
<td>((0, \leq))</td>
<td>((0, \leq))</td>
<td>((0, &lt;))</td>
</tr>
<tr>
<td>(x_1)</td>
<td>((1, &lt;))</td>
<td>((0, \leq))</td>
<td>((-1, \leq))</td>
</tr>
<tr>
<td>(x_2)</td>
<td>((3, &lt;))</td>
<td>(\infty)</td>
<td>((0, \leq))</td>
</tr>
</tbody>
</table>
Tightening constraints

The canonical DBM:

Obtained by strengthening/tightening all the constraints:

<table>
<thead>
<tr>
<th></th>
<th>$x_0$</th>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0$</td>
<td>$(0, \leq)$</td>
<td>$(0, \leq)$</td>
<td>$(-1, \leq)$</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$(1, &lt;)$</td>
<td>$(0, \leq)$</td>
<td>$(-1, \leq)$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$(3, &lt;)$</td>
<td>$(3, &lt;)$</td>
<td>$(0, \leq)$</td>
</tr>
</tbody>
</table>
Tightening constraints

Halfspace view:

\[(0 \leq x_1 < 1) \land (0 < x_2 < 3) \land (x_2 - x_1 \geq 1)\]
Tightening constraints

Halfspace view:

\((0 \leq x_1 < 1) \land (0 < x_2 < 3) \land (x_2 - x_1 \geq 1)\)

(Normalised)
Tightening constraints

Halfspace view:

\[(0 \leq x_1 < 1) \land (0 < x_2 < 3) \land (x_2 - x_1 \geq 1)\]

(Strengthened)

\[-x_2 \leq -1\]
\[x_1 - x_2 \leq -1\]
\[x_2 < 3\]
\[x_1 < 1\]
\[x_2 - x_1 < 3\]
\[-x_1 \leq 0\]

ZONE
Operations on zones

The intersection:

$D_2$ and $D_1$ illustrate the intersection of sets in a 2D space.
Operations on zones

Time elapses:

\[x\]

\[y\]

\[D\]

\[D\hat{\leftarrow}\]
Operations on zones

A clock is reset:

\[ R_y D \uparrow \]
Operations on zones

The PAST operation?

\[ D \]

\[ x \]

\[ y \]
Practice versus mathematics:

- In the **mathematics**, en-route to the finite **RTS** or **ZTS**, we construct (**infinite**) transition systems.
- This is fine, but **not** actually **possible** (obviously).
- Instead we generate the transition systems in **one step** from the TTS.
- The following slides attempt to show the flavour of the algorithm in pictures...
Drawing the operations (regions)

Show the regions in a diagram: Original state

$s_1$, $x \leq 1$, $y > 0$

$s_2$

$s_1 \xrightarrow{A; \{x\}} s_2$

Hugh Anderson
Verification of Real Time Systems - CS5270 8th lecture
Drawing the operations (regions)

Show the regions in a diagram: Time passing

\[ A; \{x\} \]

\[ x \leq 1 \quad y > 0 \]

Hugh Anderson

Verification of Real Time Systems - CS5270 8th lecture
Show the regions in a diagram: Action move

\[ s_1 \xrightarrow{A; \{x\}} s_2 \]

\[ x \leq 1 \quad y > 0 \]
Show the regions in a diagram: Carry on time...
Show the regions in a diagram: Time passing move

Diagram showing the transition from region $s_1$ to region $s_2$ with conditions $x \leq 1$ and $y > 0$. The diagram also illustrates the transitions $s_1 \rightarrow s_1'$, $s_1' \rightarrow s_2'$, $s_2' \rightarrow s_2''$ with the time passing move denoted by $\delta$. The moves are labeled with the transition relation $A; \{x\}$. The diagram includes graphical representations of the regions and transitions, with arrows indicating the direction of movement through the regions.
Drawing the operations (regions)

Show the regions in a diagram: Another action...

Diagram:

- States: $s_1$, $s_2$, $s_1'$, $s_2'$, $s_1''$, $s_2''$
- Transitions:
  - $A; \{x\}$ from $s_1$ to $s_2$
  - $\delta$ from $s_1'$ to $s_1''$
  - $\delta$ from $s_2'$ to $s_2''$
  - $A$ from $s_1'$ to $s_2'$
  - $A$ from $s_1''$ to $s_2''$
- Variables:
  - $x \leq 1$
  - $y > 0$
Show the regions in a diagram: and so on...

\[ s_1 \xrightarrow{\delta} s'_1 \xrightarrow{\delta} s''_1 \]
\[ s_2 \xrightarrow{\delta} s'_2 \xrightarrow{\delta} s''_2 \]

\[ s_1 \xleftarrow{\delta} s'_2 \xleftarrow{\delta} s''_2 \]

\[ A; \{x\} \]

\[ x \leq 1 \quad y > 0 \]
Show the regions in a diagram: TIME ABSTRACTED

\[ s_1 \xrightarrow{x \leq 1, y > 0} s_2 \]

Hugh Anderson
Verification of Real Time Systems - CS5270 8th lecture
Show the regions in a diagram: from before...

\[
\begin{align*}
S_1 & \xleftarrow{\delta} S_1' \\
S_1' & \xrightarrow{A} S_2' \\
S_2' & \xleftarrow{\delta} S_2'' \\
S_2'' & \xrightarrow{A} S_2'' \\
\end{align*}
\]
Drawing the operations (zones)

Show the zones in a diagram: smaller

\[ s_1 \xrightarrow{x \leq 1, y > 0} s_2 \]

\[ s_1 \xrightarrow{\delta} s_1' \]

\[ s_2 \xrightarrow{\delta} s_2' \]
Drawing the operations (zones)

Show the zones in a diagram: TIME ABSTRACTED

\[ s_1 \quad A; \{x\} \quad s_2 \]

\[ x \leq 1 \quad y > 0 \]

Hugh Anderson

Verification of Real Time Systems - CS5270 8th lecture
Operations on zones

Zones are relatively easily manipulated:

- Following three operations are needed for use in evaluating zone transitions:
  - If $\mathcal{D}_1$ and $\mathcal{D}_2$ are two clock zones, then the intersection of the zones is a new clock zone $\mathcal{D}_1 \land \mathcal{D}_2$.
  - $\mathcal{D} \uparrow$ is the **time-elapsed zone** defined by $\mathcal{D} \uparrow = \{ V + \delta \mid V \in \mathcal{D} \}$ with $\delta \in \mathbb{R}_{\geq 0}$.
  - The **clock-reset zone** $R_X \mathcal{D}$ is defined by $R_X \mathcal{D} = \{ R_X(V) \mid V \in \mathcal{D} \}$ where $R_X(V)(\lambda) = 0$ if $\lambda \in X$ or $R_X(V)(\lambda) = V(\lambda)$ otherwise.
Outline

1. Administration
   - Assignment 2
   - The road map...

2. Efficiency in TTS
   - From regions to zones
   - Matrix notation and zone operations
   - Closed zones and graph representation

3. Preliminaries to Model Checking
   - Behaviour, safety, liveness, automata, reachability...
   - Extensional and intensional logic
   - Linear and branching time
Want a canonical representation

Equivalence of zones:
- We do not want two different zones to represent the same set of valuations (i.e. \((y - x \leq 3, x = 2, y = 4)\) the same as \((y - x = 2, x = 2, y = 4)\).

**Definition:** A zone is *closed* if no constraint can be strengthened without reducing the set of associated valuations.
Closed zones are equivalent iff identical

Graph representations (simplified)...

\[\begin{align*}
\text{x}_0 & \rightarrow \text{x}_1 & \rightarrow \text{x}_2 & \rightarrow \text{x}_3 & \rightarrow \text{x}_0 \\
& & & & \text{x}_0 \\
\text{x}_1 & & & & \text{x}_3 \\
\& & & & \text{x}_3
\end{align*}\]
A shortest path reduction is performed on the graph (computed in $O(n^3)$ time), where redundant edges are removed when they can be. For example $x_1 \xrightarrow{10} x_2$ is replaced by $x_1 \xrightarrow{2} x_3 \xrightarrow{2} x_2$. 
If $D$ is closed then $D$ is a subset of $D'$ iff for every constraint $x - y \leq m'$ in $D'$ there is $x - y \leq m$ in $D$ with $m \leq m'$. If $D$ is closed then $D$ is non-empty iff there are no negative weight cycles in the graph.
Graphs for DBMs: graph reduction...

\[ (0 \leq x_1 < 1) \land (0 < x_2 < 3) \land (x_2 - x_1 \geq 1) \]
Graphs for DBMs: graph reduction...

\[(0 \leq x_1 < 1) \land (0 < x_2 < 3) \land (x_2 - x_1 \geq 1)\]
Outline

1. Administration
   - Assignment 2
   - The road map...

2. Efficiency in TTS
   - From regions to zones
   - Matrix notation and zone operations
   - Closed zones and graph representation

3. Preliminaries to Model Checking
   - Behaviour, safety, liveness, automata, reachability...
   - Extensional and intensional logic
   - Linear and branching time
Behaviours

The behaviour of a TS...

- ...is its set of runs, or its set of computations.

To verify behaviours against a property, we can consider questions like:

- Does every computation (run) of the transition system have a desired property X? or
- Is it true that in no computation, C is immediately followed by on-ac?.
Two types of behaviours:

- In handouts, the ideas of **safety** and **liveness** were introduced, identifying two types of behaviours that require different analysis methods.
  - A **safety property** is like “something bad doesn’t happen”, whereas
  - **liveness** is like “something bad (or good) must eventually happen”.
- We can often formulate safety properties in terms of the reachability of a state.
What is an automata?

- An automata is a state transition system with some set of accepting states, which may be used to distinguish between good and bad computations.

- We can use automata matching a particular transition system to specify desired behaviour of the system, in a form like “Is there a run of the automaton that leads to the (desired) accepting state?”, or “Is there a run of the automaton that leads to an accepting state in which property $P$ holds?”.

- These are examples of a reachability problem.
Finite automaton

Definition:
A finite automaton is a 5-tuple \((Q, \Sigma, \Delta, q_0, F)\), where

- \(Q\) is a finite set called the **states**
- \(\Sigma\) is a finite set called the **alphabet**
- \(\Delta : Q \times \Sigma \rightarrow Q\) is the **transition function**
- \(q_0\) is the **start state**
- \(F \subseteq Q\) is the set of **accepting states**
Automata theory...

These sort of problems have clear links to automata theory, and

we could easily cast a lot of this discussion in terms of the languages accepted by (finite) automata.

To reason about liveness properties, we need to consider infinite sequences.

A Büchi automaton is an extension of a finite state automaton to one which accepts an infinite input sequence if, and only if, there is a run of the automaton which has infinitely many states in the set of final states.
Büchi automata

Definition:
A Büchi automaton is a 5-tuple \((Q, \Sigma, \Delta, q_0, F)\), as for a regular automaton, but with \(F\) interpreted differently. In particular \(s_0 a_0 s_1 a_1 s_2 \ldots\) is an accepting infinite trace if

- \(s_0 \in Q\)
- \((s_i, a_i, s_{i+1}) \in \Delta\) for all \(i\)
- For infinitely many \(j\), the state \(s_j\) is in \(F\)
Büchi automata

What are they good for?

They are useful for specifying behavior of nonterminating systems, such as
- hardware (electronic circuits) or
- operating systems.

For example, you may want to specify a property like
- “for every measurement, a recording eventually follows”, or
- the reverse “there is a measurement which is not followed by a recording”.

For the second example, an argument limited to finite sequences cannot satisfy this property.
Properties: Reachability and deadlock

Are related...

- For example, is there a run leading to **deadlock**?
- A deadlocked system can do no more computation, more formally:

**Definition**: The run $s_0 \xrightarrow{*} s_k$ with $s_0 \in S_{in}$ is in **deadlock** if no action is enabled at $s_k$. 
Properties: qualitative

Questions such as...

- Every request is eventually served.
- The sensor signal \( x_{11} \) is sensed infinitely often.
- From any stage of the computation the all clear state can be reached within 3 steps
Properties: quantitative

Questions such as...

- Every request is served within 3 microseconds.
- The sensor signal $x_{11}$ is sensed every 10 milliseconds for ever.
- From any stage of the computation the all clear state can be reached within 1 second.
Outline

1 Administration
   • Assignment 2
   • The road map...

2 Efficiency in TTS
   • From regions to zones
   • Matrix notation and zone operations
   • Closed zones and graph representation

3 Preliminaries to Model Checking
   • Behaviour, safety, liveness, automata, reachability...
   • Extensional and intensional logic
   • Linear and branching time
The universe is not black and white

Consider...

Please answer YES or NO: Will the next answer you give me be NO?

You are either going to die in a bomb raid or you are not...

- **Extensional** logic means that you can determine the truth of a formula from the truth values of its parts.
- **Intensional/modal** logic refers to **QUALIFIED truth** (words like *could*, *eventually*, *possibly* and so on).
QUALIFIED truth

- The basic modal operators are
  - \( \Box \) which represents necessity and
  - its dual \( \Diamond \) which represents possibility (\( \Diamond A = \neg \Box \neg A \)).

- The language of modal logic consists of
  - propositional variables,
  - a set of Boolean connectives such as \( \{ \land, \lor, \neg \} \), and
  - the modal operators.
Consider...

“The engine is too hot.”

- The *meaning* is clear, it does not vary with time, but ...
- the *truth* value of the assertion *can vary in time*.
  - Sometimes it is true, and sometimes it is false, and
  - it is never true and false simultaneously.

Temporal logics are a good mechanism for expressing *qualitative* temporal properties of reactive systems.

Operators related to TIME, so that (for example) \( \Box \phi \) means that propositional variable \( \phi \) must hold in all the following (later) states.
Temporal operators

Common to use mnemonic letters $X, G, F, U, R...$

- **Operators:**
  - $X \phi$ indicating that $\phi$ must hold in the **next** state.
  - $G \phi$ (or $\Box \phi$) indicating that $\phi$ must hold in all the **following** states.
  - $F \phi$ (or $\Diamond \phi$) indicating that **eventually** (finally) $\phi$ must hold somewhere.
  - $\phi U \psi$ indicating that $\phi$ has to hold **until** $\psi$ holds at the current or a future position.
  - $\phi R \psi$ The dual of $U$, $\psi$ holds until the first state where $\phi$ holds.

- **Quantification**
  - $A$ for all paths
  - $E$ there **exists** a path...
Outline

1. Administration
   - Assignment 2
   - The road map...

2. Efficiency in TTS
   - From regions to zones
   - Matrix notation and zone operations
   - Closed zones and graph representation

3. Preliminaries to Model Checking
   - Behaviour, safety, liveness, automata, reachability...
   - Extensional and intensional logic
   - Linear and branching time
LTL: Linear time view

The set of runs...

\[ \{01010101 \ldots, 01023232 \ldots, \} \]
CTL: Branching time view

Branches into the future...
Linear Temporal Logic, Computation Tree Logic

- In LTL, one can encode formulæ about events along a single computation path.
- By contrast, CTL is a modal branching-time temporal logic. The operators quantify over all possible future paths from a given state.
- CTL and LTL are both subsets of a more general temporal logic CTL*.
- There are expressions in CTL that cannot be expressed in LTL and vice versa.
- In CTL formulæ each of the temporal operators must be preceded by a path quantifier: A, or E.
Consider this TS:

- $\text{A} \left( \text{FG} \ p \right)$, is an LTL formula representing: for all paths, eventually $p$ holds globally (i.e. from then on).
- $\text{AF} \left( \text{AG} \ p \right)$ is CTL for: for all paths, eventually you get to a state where for all paths $p$ holds globally (i.e. from then on).
- LTL formula is not the same thing as CTL formula.
LTL: All runs that start in $s$ have $p$ holding eventually:

The possible (infinite) runs from $s$ are

$$ssssssssss... \} \text{ i.e. } s^\infty$$

or...

$$stuuuuuu...$$

$$ssstuuuuu...$$

$$sssstuuuu...$$

$$...$$

so in the linear time view, for state $s$, $A(FG p)$.
CTL: Eventually you get to where $p$ holds from then on:

The CTL counterexample for $\text{AF}(\text{AG} p)$ is

so in the CTL view, for state $s$, $\text{AF}(\text{AG} p)$ is not true.