Verification of Real Time Systems - CS5270 lecture 10

Hugh Anderson

National University of Singapore
School of Computing

March, 2007

Delorean...

Outline

1. Administration
   - Assignment 3
   - The road map...
2. Model checking
   - CTL example (smv/nusmv)
   - LTL model checker - spin
3. Timed CTL model checking
   - Regional transition systems...
   - Timed CTL-
   - The modelling relation $\models_\tau$ for timed CTL-

Assignment 3

A reminder... Assignment number 3:

- On the web site
- Due 9th April! ...

The immediate road map

The topics:

- **Preliminaries for Model Checking**
  - Temporal logic..., Kripke semantics
- **CTL Model Checking**
  - The CTL model checking relation
  - The CTL model checking algorithm, with optimizations

  - Example smv/spin - CTL/LTL model checkers
- **TCTL Model Checking**
  - The TCTL model checking relation
  - The TCTL model checking algorithm, with optimizations
  - Example Uupaal - TCTL model checker

The smv model checker

CTL model checker:

- McMillan (1992) wrote smv a CTL model checker, making sources public
- More recently, NuSMV is being developed further, and now includes LTL, and other extensions including SAT solvers, BMC...
- Now has an extensive history

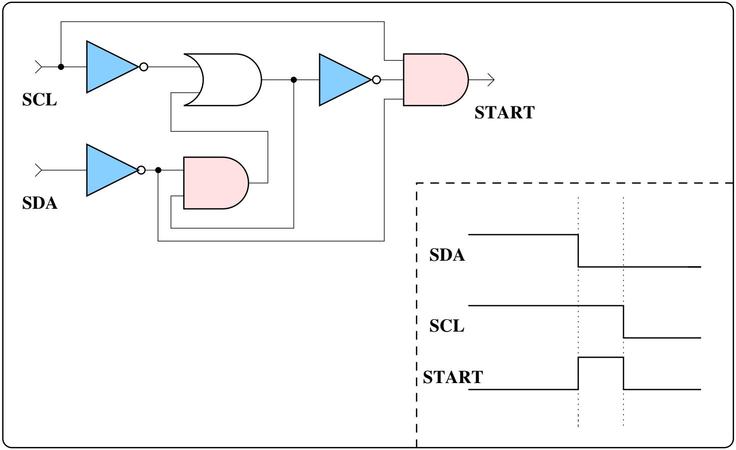
The smv model checker

10^{2}00 states and beyond!

- The tool smv has been particularly useful for hardware design checking, although it is similar to all the other style systems we have seen
- specifying a model in an automata style
- useful for any responsive software system (protocols)
- For a change give a hardware verification example
  - electronic circuit to detect the START condition for I2C signalling.

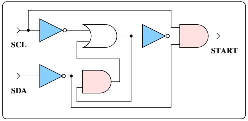
Example: smv model checker

If SDA goes low while SCL is high...



Example: smv model checker

SMV code for the circuit:




```
MODULE mainVAR
    inv1      : boolean;
    inv2      : boolean;
    inv3      : boolean;
    or2gate1  : boolean;
    and2gate1 : boolean;
    and3gate1 : boolean;
    SCLIN     : boolean;
    SDAIN     : boolean;
ASSIGN
    inv1              := !SCLIN;
    inv2              := !SDAIN;
    inv3              := !or2gate1;
    next( or2gate1 ) := inv1 | and2gate1;
    and2gate1         := inv2 & or2gate1;
    and3gate1         := inv3 & SCLIN & inv2;
SPEC
    (AG((SDAIN=1)&(SCLIN=1)&(AX SDAIN=0)) -> AF and3gate1)
```

Example: smv model checker

When we try it out:

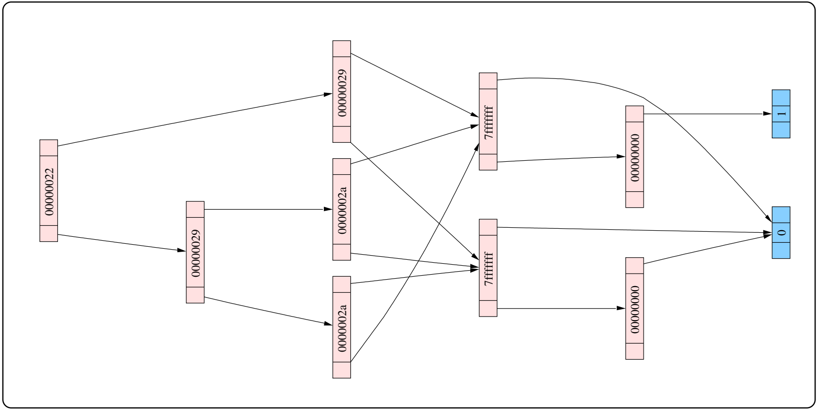
```
[hugh@pnp176-44 FormalVerification]$ NuSMV I2C
*** This is NuSMV 2.3.1 (compiled on Mon Apr 3 10:11:22 UTC 2006)
*** For more information on NuSMV see <http://nusmv.irst.itc.it>
*** or email to <nusmv-users@irst.itc.it>.
*** Please report bugs to <nusmv@irst.itc.it>.
-- specification
    (AG ((SDAIN = 1 & SCLIN = 1) & AX SDAIN = 0) -> AF and3gate1)
    is true
[hugh@pnp176-44 FormalVerification]$
```

- Verifies that it is always true that if SDA and SCL were both HIGH, and that if the next state had SDA LOW, then eventually and3gate=START will be HIGH?

Example: smv model checker

ROBDD representation of model:

- Generated automatically from within smv...



The spin model checker

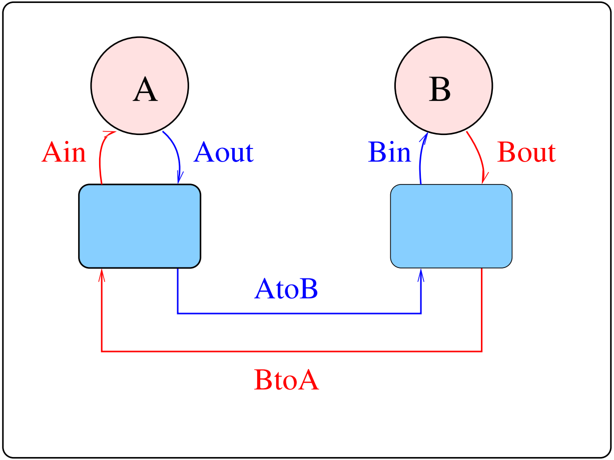
LTL style:

- spin is the name of an LTL model checker
- xspin is a graphical interface for it
- PROMELA is the model checking language it uses.
- Developed by Gerard Holzmann at Bell Labs at

http://www.spinroot.com/

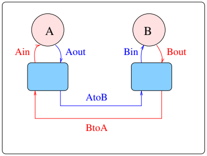
Modelling protocols

Modelling the whole system:



Modelling protocols

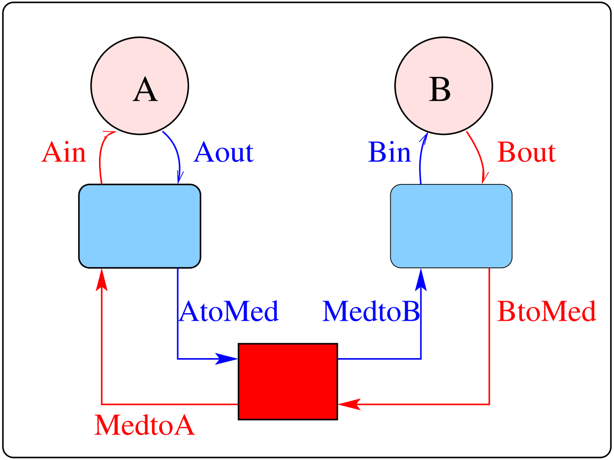
PROMELA code to connect whole system:




```
chan  Ain    = [2] of { byte };
chan MedtoA = [2] of { byte, int };
...
atomic  {
    run protocol( Ain, Aout, MedtoA, AtoMed );
    run protocol( Bin, Bout, MedtoB, BtoMed );
    run medium( AtoMed, MedtoB );
    run medium( BtoMed, MedtoA );
    run application( Aout, Ain );
    run application( Bout, Bin )
};
```

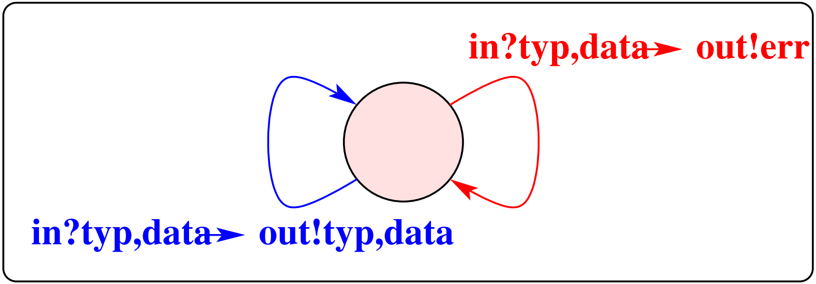
Modelling protocols

System with a noisy channel:



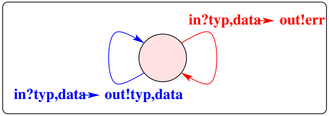
Modelling protocols

A noisy channel transition system:



Modelling protocols

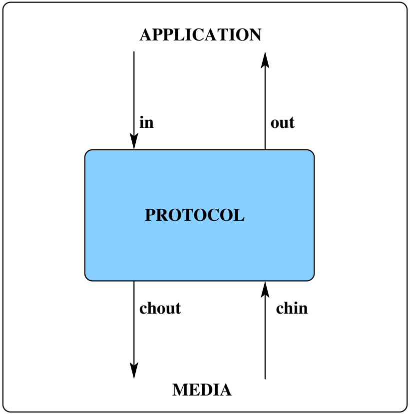
PROMELA code for noisy channel transition system:




```
proctype medium( chan in, out ) {
    byte typ;
    int data;
    do
        :: in?typ,data ->
              if
                    :: out!typ,data
                    :: out!err,0
              fi
    od
};
```

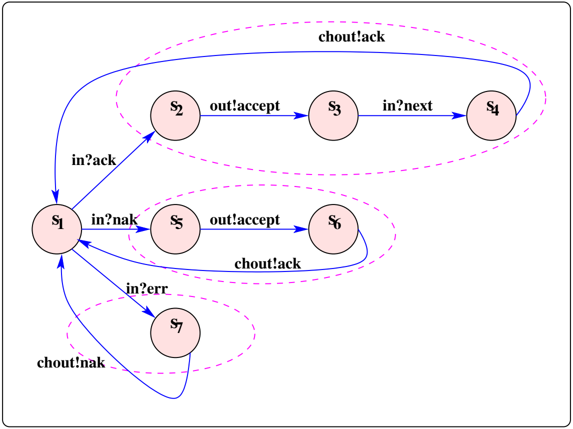
Modelling protocols

Protocol lies between the application and the media:



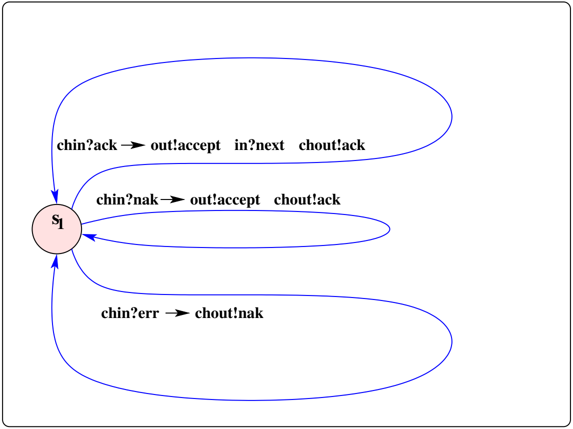
Modelling protocols

Protocol transition system:



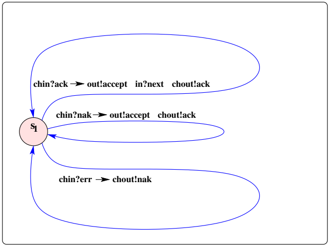
Modelling protocols

Simplified protocol transition system:



Modelling protocols

PROMELA code for protocol:




```
proctype  protocol( chan in, out, chin, chout ) {
   byte o,i;
   in?next(o);
   do
      :: chin?ack(i) -> out!accept(i); in?next(o); chout!ack(o)
      :: chin?nak(i) -> out!accept(i); chout!ack(o)
      :: chin?err(i) -> chout!nak(o)
   od
  }
```

Specifying properties?

For reachability and temporal claims:

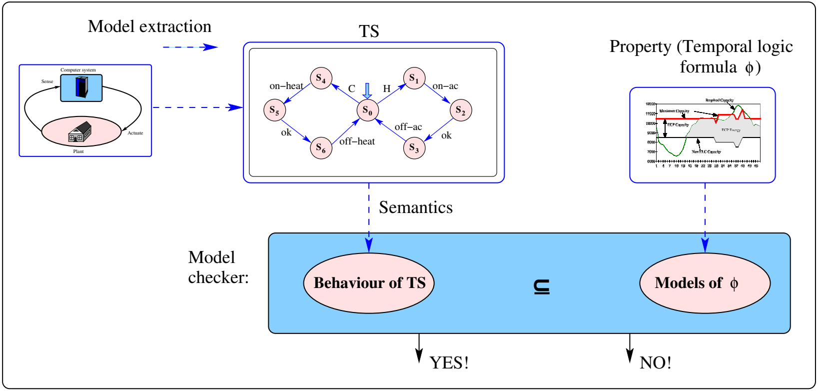
- Add assertions in the PROMELA code for reachability claims:

```
assert(maxbuffered=3);
```

- Convert LTL formula into Buchi automata for (negative) temporal claims?

The big picture again......

Models, properties, LANGUAGE of a model, property:



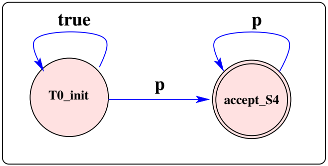
The big picture

Model theoretic view:

- To assert an LTL temporal formula/claim on a model, we have to show that the language of the model (all executions) is included in the language of the claim.
- It is easier to claim the negative ...
  - It is easier to prove that the intersection of the language of the model and the claim is empty.
  - Hence NEVER claims in PROMELA.

Modelling claims: A(FG p) ... spin -f “<>[]p”

PROMELA code for NEVER claim:




```
never {    /* <>[]p */
   T0_init:    if
                  :: ((p)) -> goto accept_S4
                  :: (1) -> goto T0_init
               fi;
   accept_S4:  if
                  :: ((p)) -> goto accept_S4
               fi;
   }
```

If the product of the model and the negation of this automaton is empty (i.e. no acceptance), then A(FG p).

Regional transition system

Timed CTL versus traditional CTL:

- Major difference is that we have clocks:
  - in the automaton (X, a finite set of clock variables), and
  - in the TCTL formula (Z, a different finite set of clock variables).

We have to take these clocks into account both in the definition of TCTL, and also the model checking relation for TCTL: $\models_\tau$. The Kripke structure is different, as it corresponds to an RTS (regional transition system) instead of a standard transition system.

Regional transition system

The Kripke structure for RTS:

- The (composite) states of the RTS are a pair $\bar{r} = (s, [v]_\approx)$, termed a *region*, where s corresponds to the original state of the transition system, and $[v]_\approx$ is the regional equivalence class for v as defined in Chapter 4.

We begin by formally defining an RTS, and its corresponding model, in terms of the time abstract transition system $\mathrm{TA_{TTS}}$. Recall that the (possibly infinite) states in $\mathrm{TA_{TTS}}$ are of the *composite* form (s, v), where s corresponds to the states of the original timed transition system, and v is a valuation of the clocks of that syste.

Regional transition system

Formal definition of RTS:

Given $\mathrm{TA}_{\mathrm{TTS}} = (S, S_0, \mathrm{Act}, \rightsquigarrow)$, then the RTS is a quotiented transition system $\mathrm{RTS} = (\overline{R}, \overline{R}_0, \mathrm{Act}, \rightarrow)$ where

$$\begin{array}{rcl} \overline{R} & = & \{(s, [v]_\approx) \mid (s, v) \in S \wedge v \in [v]_\approx\}, \text{ and} \\ \overline{R}_0 & = & \{(s, [v]_\approx) \mid (s, v) \in S_0 \wedge v \in [v]_\approx\}, \end{array}$$

and $(s, [v]_\approx) \xrightarrow{a} (s', [v']_\approx)$ if and only if there is a transition $(s, v) \overset{a}{\rightsquigarrow} (s', v')$ in $\mathrm{TA}_{\mathrm{TTS}}$.

Regional transition system

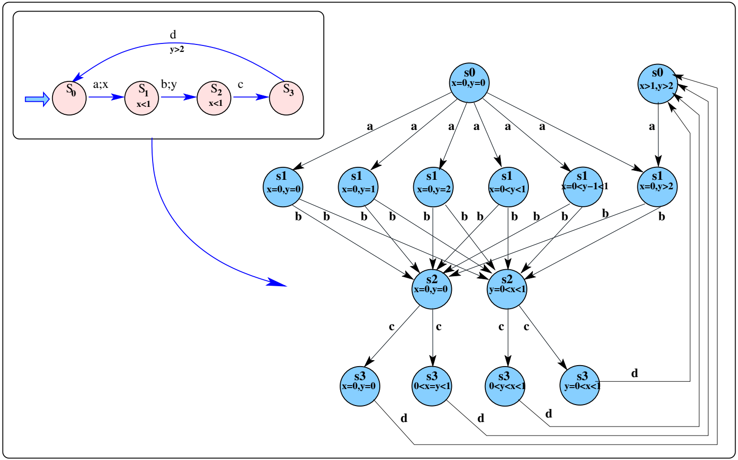
Notations when discussing RTS:

There are three levels:

1. The elements of the set $\overline{R}$ are called the *regions* of the RTS.
2. The notation for identifying a particular region will be $\overline{r} \in \overline{R}$, and
3. a (transitory) state with a particular clock valuation within that region will be denoted by $r = (s, v)$.

Regional transition system

From TTS to RTS:



Regional transition system

RTS is finite (reminder):

- Note that since $\equiv_{\mathrm{REG}}$ is a stable equivalence relation of finite index, the RTS is a finite structure.
- We do not need to differentiate between the RTS and the *zone* based transition system here, instead considering that the zone based transition system is just a more efficient version of the RTS.

The semantics for TCTL is again defined in terms of a Kripke structure or TCTL-model. This model is derived from the RTS.

Regional transition system

Definition for the model:

A TCTL model $\overline{M}$ over a set AP of atomic propositions is a 4-tuple $(\overline{R}, \Delta, AP, \mathcal{L})$, where

- $\overline{R}$ is the finite set of **regions** derived from the RTS.
- $\Delta \subseteq \overline{R} \times \overline{R}$ is a **transition relation** derived from $\rightarrow$ in RTS. It must be *total.*
- AP is a finite set of **atomic propositions**.
- $\mathcal{L} : \overline{R} \rightarrow 2^{AP}$ is a function which **labels** each region with the set of atomic propositions true in that region.

Timed CTL-

Definition:

Given a proposition $p \in AP$ (atomic propositions), $x \in X$ (clock variables), $z \in Z$ (clock variables in the property formula) and $\phi \in \Phi(X \cup Z)$ (clock constraints), then p and ϕ are both TCTL- formulæ, and if ψ_1 and ψ_2 are TCTL- formulæ, then

- $\neg\psi_1$ is a TCTL- formula
- $\psi_1 \wedge \psi_2$ is a TCTL- formula
- $\psi_1 \vee \psi_2$ is a TCTL- formula
- $z \text{ in } \psi_1$ is a TCTL- formula
- $\mathrm{A}(\psi_1 \,\mathrm{U}\, \psi_2)$ is a TCTL- formula
- $\mathrm{E}(\psi_1 \,\mathrm{U}\, \psi_2)$ is a TCTL- formula

Timed CTL-

How time is handled in timed CTL-:

- The FREEZE definition in line 4. The meaning of "$z \text{ in } \psi$" is that ψ holds when the property formula clock z is reset to 0. Corresponds to the clock reset.
- Temporal operators are subscripted with time constraints:

$$A(\psi_1 \, U_{\leq 5} \, \psi_2)$$

expresses the idea that ψ_1 holds until within 5 time units, ψ_2 becomes true. This may be defined in TCTL- using the FREEZE operator:

$$z \text{ in } A((\psi_1 \wedge z \leq 5) \, U \, \psi_2)$$

Timed CTL-

How time is handled in timed CTL-:

- Example 1:

$$A(\text{alarm}\,U_{<7}\,\text{boileroff})$$

expresses the idea that the alarm is on until (within 7 time units) the boileroff is signaled.

- Example 2:

$$EF_{<7}(\text{alarm})$$

expresses the idea that the alarm will be on within 7 time units.

Timed CTL- $\models_\tau$ relation

Definition:

$$
\begin{array}{lcl}
\overline{M},(r,f) \models_\tau p & \Leftrightarrow & p \in L(\overline{r}) \\
\overline{M},(r,f) \models_\tau \phi & \Leftrightarrow & v \cup f \models \phi \\
\overline{M},(r,f) \models_\tau \neg\psi_1 & \Leftrightarrow & \text{iff it is not the case that } \overline{M},(r,f) \models_\tau \psi_1 \\
\overline{M},(r,f) \models_\tau \psi_1 \wedge \psi_2 & \Leftrightarrow & \text{iff } \overline{M},(r,f) \models_\tau \psi_1 \text{ and } \overline{M},(r,f) \models_\tau \psi_2 \\
\overline{M},(r,f) \models_\tau \psi_1 \vee \psi_2 & \Leftrightarrow & \text{iff } \overline{M},(r,f) \models_\tau \psi_1 \text{ or } \overline{M},(r,f) \models_\tau \psi_2 \\
\overline{M},(r,f) \models_\tau z \text{ in } \psi_1 & \Leftrightarrow & \text{iff } \overline{M},(r,z \text{ in } f) \models_\tau \psi_1 \\
\overline{M},(r,f) \models_\tau \mathrm{A}(\psi_1 \,\mathrm{U}\, \psi_2) & \Leftrightarrow & \text{iff for every path } \overline{\pi} = s_0\ s_1\ \ldots \text{ from } r, \\
 & & \text{where for some } j, \overline{M},\overline{\pi}(j) \models_\tau \psi_2, \\
 & & \text{and } \forall i < j, \overline{M},\overline{\pi}(i) \models_\tau \psi_1 \vee \psi_2 \\
\overline{M},(r,f) \models_\tau \mathrm{E}(\psi_1 \,\mathrm{U}\, \psi_2) & \Leftrightarrow & \text{iff there is a path } \overline{\pi} = s_0\ s_1\ \ldots \text{ from } r, \\
 & & \text{where for some } j, \overline{M},\overline{\pi}(j) \models_\tau \psi_2, \\
 & & \text{and } \forall i < j, \overline{M},\overline{\pi}(i) \models_\tau \psi_1 \vee \psi_2
\end{array}
$$

The modelling relation $\models_\tau$

Comments on the relation:

- In this definition, the progression of time is defined in reference to the states of the original $\mathrm{TS}_{\mathrm{TTS}}$. In particular, a path from one state r is an infinite sequence of states $\overline{\pi} = s_0\ s_1\ \ldots$ such that $s_0 = r$ and $s_i \rightarrow s_{i+1}$. A particular i-th element of $\overline{\pi}$ is $\overline{\pi}(i)$.
- The notation found in the definition for the **FREEZE** operator $(\overline{M}, (r, z \text{ in} f) \models_\tau \psi_1)$ indicates that $\overline{M}, (r, f) \models_\tau \psi_1$ if all occurences of z in f are reset to 0.

The modelling relation $\models_\tau$

Comments on the relation:

- An interesting element of the definition is found in the definitions for $\mathrm{E}(\psi_1 \,\mathrm{U}\, \psi_2)$ and $\mathrm{A}(\psi_1 \,\mathrm{U}\, \psi_2)$, where at some j, $\overline{M}, \overline{\pi}(j) \models_\tau \psi_2$, but for all $i < j$, $\overline{M}, \overline{\pi}(i) \models_\tau \psi_1 \vee \psi_2$.
- If you compare this with the similar definition from CTL, you find in that case the condition "for all $i < j$, $M, \overline{\pi}(i) \models \psi_1$" (i.e. ψ_1 instead of $\psi_1 \vee \psi_2$).

The modelling relation $\models_\tau$

Explanation:

We can see the need for the expression $\psi_1 \vee \psi_2$ instead of just ψ_1 by considering the *big* step from a particular valuation in $\bar{r}_1$ to another in $\bar{r}_2$ seen below. For all points in the two regions we want $\mathrm{A}(\psi_1 \,\mathrm{U}\, \psi_2)$, but for the two points connected by the line, ψ_1 is not true just before the new point