Verification of Real-Time Systems

Examples

Example: Petersson's algorithm

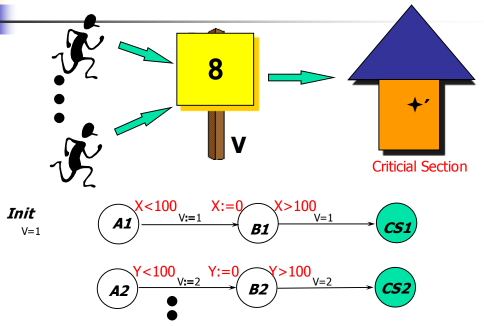
turn: shared variable

- Process 1
- Loop
- flag1:=1; turn:=2
- While (flag2 and turn=2) wait
- CS1
- flag1:=0
- End loop

- Process 2
- Loop
- flag2:=1; turn:=1
- While (flag1 and turn=1) wait
- CS2
- flag2:=0
- End loop

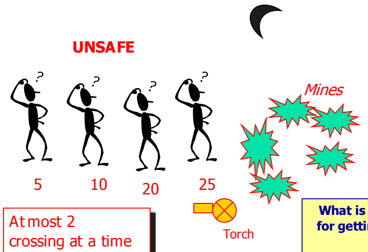
Question: no more than one process run in CS?

Example: Fischer's Protocol



Example: the Vikings Problem

Real time scheduling



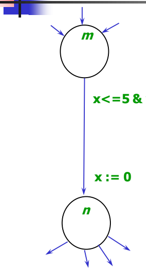
At most 2
crossing at a time
Need torch

What is the fastest time for getting all vikings on the safe side ?

Problem: reachability analysis

- Give an automaton and a location n, or a local property F
- Question: does it exist an execution of the automaton, that leads to n (or a state where F holds)?
- This is the so called reachability problem.

Timed Automata: Semantics



State

(***location*** , ***clock-assignment***)

Transitions

$(m, x=2.4, y=3.1415) \xrightarrow{1.1} (m, x=3.5, y=4.2415)$

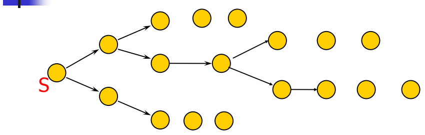
$(m, x=1.14, y=3.1415) \longrightarrow (n, x=0, y=3.1415)$

Reachability Problems

n is reachable from m if there is a sequence of transitions:

$(m, x=r, y=s) \longrightarrow^{*} (n, x=r', y=s')$

Computation Tree (of a system)

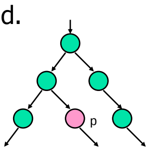


all possible executions of a systems

Specifying properties of real-time systems
--- UPPAAL querry language

E<>p "p Reachable"

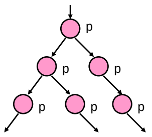
- it is possible to reach a state in which p is satisfied.



- p is true in (at least) one reachable state.

A[]p "Invariantly p"

- **`A[] p`** – p holds invariantly.

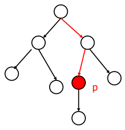


- p is true in all reachable states.

Specifying properties in UPPAAL

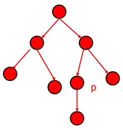
P is Possible

Written as E<> p in UPPAAL



P is always true

Written as A[] p in UPPAAL



```
p::= A.n | gd | gc | p and p |
     p or p | not p | p imply p
```

where

- A.n denotes the node n of automaton A
- gd is a guard on data variables
- gc is a guard on clocks

Example querries in UPPAAL

- Reachability properties: E<> Q
  - E<> P.stop
  - E<> (y>200)
- Invariant properties: A[] Q
  - A[] not (P1.CS and P2.cs)
  - A[] (i < 100)
  - A[] (x>10 imply i>100)
    - After 10, i should be larger than 100
- Deadlock-freedom
  - A[] !deadlock