AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

Test of Distributed Systems
Lecture 12

Never claims:

Safety properties

Liveness properties

Examples

Today's lecture

- Automata and Acceptance
- Safety Properties
- Liveness Properties

Semantics

- The meaning of a process is described by an automaton, also called finite-state transition system:
  - set of states
  - set of state-to-state transitions
- Two kinds of automaton:
  - those accepting ***finite*** words (Standard)
  - those accepting ***infinite*** words (Büchi)

Büchi Automaton

- A Büchi automaton has an acceptance condition for infinite computation sequences
- A Büchi automaton accepts an infinite computation sequence there if exists a state that is visited infinitely often
- All LTL formulas can be expressed as a Büchi automaton

Verifying an LTL formula

- To verify that model M satisfies LTL formula f we need:
  - A: the automaton for the model M
  - B: the Büchi automaton for the negation of f
  - P: the product automaton of A and B
- In the product automaton each transition denotes a joint transition of automata A and B
- If P accepts an infinite word, then M does not satisfy f

Verifying a Safety Property

- Verify that []mutex holds for model M
- You claim []mutex
- The model checker SPIN attempts to show that ![]mutex holds
- Play a game with SPIN:
  - You "win" if it is never true that !mutex
  - SPIN "wins" if it can find a computation in which !mutex holds

Never Claim for ![]mutex

```
never { /* !([]mutex) */
T0_init:
  if
  :: (! ((mutex))) -> goto accept_all
  :: (1) -> goto T0_init
  fi;
accept_all:
  skip
}
```

Model Checking for Safety

- If ***the final state*** of the never claim is reached, the property ![]mutex holds
- That is, the property []mutex does not hold
- The model checker yields a ***counterexample*** in form of a sequence leading to that state
- The acceptance condition is a ***finite word***

Verifying a Liveness Property

- Verify that <>csp holds for model M
- You claim <>csp
- The model checker SPIN attempts to show that !<>csp holds
- Play a game with SPIN:
  - You "win" if there is a computation in which csp holds
  - SPIN "wins" if it can find a computation in which csp never holds (→ acceptance cycle)

Never Claim for !<>csp

```
never { /* !(<>csp) */
accept_init:
T0_init:
  if
  :: (! ((csp))) -> goto T0_init
  fi;
}
```

Model Checking for Liveness

- If an acceptance cycle of the never claim is reached, the property !<>csp holds
- That is, the property <>csp does not hold
- The model checker yields a ***counterexample*** in form of a loop of states
- The acceptance condition is an ***infinite word***

Never Claim for ![]<>csp

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

Never Claim for !<>[]csp

```
never { /* !(<>[]csp) */
T0_init:
  if
  :: (! ((csp))) -> goto accept_S9
  :: (1) -> goto T0_init
  fi;
accept_S9:
  if
  :: (1) -> goto T0_init
  fi;
}
```