Properties of Concurrent Programs

- Concurrent programs are (usually) *reactive*
- Properties must deal with *behaviour*, not results
- Two kinds of *functional properties*:
  - **Safety properties** Program does nothing wrong
  - **Liveness properties** Program does something (good)
- Real-time and performance requirements may be added

Safety Properties

- A safety property ensures that the program does nothing wrong

Examples

- *At most one process may use the printer at a time*
- *The variable x never decreases*
- *The motor turns off only if the key has been removed*

Formal treatment

- Property ϕ is satisfied for execution α: $\phi[\alpha]$
- ϕ is a *safety property* iff

$$\forall \alpha : \neg\phi[\alpha] \Rightarrow \exists \beta \leq \alpha : \forall \gamma : \neg\phi[\beta\gamma]$$

- Can be stated by *invariants* (and history variables)
- Can be shown by *model-checking* or *inductive proofs*

Liveness Properties

- A liveness property ensures that the program makes progress

Examples

- *The program will return to input mode again and again*
- *The variable x will never become constant*
- *The green light is lit when the Go button is pressed*

Formal treatment

- Can be stated using *Temporal Logic*
- Can be shown by *temporal reasoning*, e.g. proof lattices

Lack-of-progress Properties

Deadlock

- *Deadlock* = cycle of processes waiting for each other (for ever)
- Typical cause: incremental reservation of shared resources

Starvation

- A process suffers from *starvation* if it could make progress, but never does so
- Typical causes: Unfair scheduling, priorities, bad luck

Livelock

- *Livelock* = mutual starvation (*after-you-after-you*)
- Like deadlock, but can be *escaped*
- Typical cause: Symmetrical strategies
- **Note:** Andrews sets *livelock* = *deadlock* — we don't

Atomic Actions

Idea

- *Atomic* = virtually indivisible

Definition

- Two actions, a and b, are *mutually atomic* iff

  $a \parallel b$ has the same effect as $a; b$ or $b; a$

- A program has *atomic actions* if they are mutually atomic
- Assuming a to be atomic is denoted by $\langle\, a \,\rangle$

Interleaving Model

- Assume that all actions a program are atomic:

  Any (parallel) execution of the program corresponds to some sequential *interleaving* of the atomic actions

Critical References

- A *simple variable* is held in a machine word
- Access to simple variables is assumed atomic on standard HW
- A *critical reference* is either:
  - Reading a simple variable written by another process
  - Writing a simple variable accessed by another process
- Access to *non-simple variables* counts for more critical references

Rule of Critical References

- S contains at most one critical reference $\Rightarrow$ S is atomic

Transition Systems

- General mathematical model of discrete behaviour

Definitions

- A *(labelled) transition system* TS is a tuple $(\Sigma, \mathcal{A}, \mathcal{T}, s_0)$, where:
  Σ is a set of *states*
  $\mathcal{A}$ is a set of *actions* (or labels)
  $\mathcal{T}$ $\subseteq \Sigma \times \mathcal{A} \times \Sigma$ is the *transition relation*
  s_0 $\in \Sigma$ is the *initial state*
- $(s, a, s') \in \mathcal{T}$:
  Action a can be executed in state s resulting in a new state s'
- For a given TS, this fact is usually written $s \xrightarrow{a} s'$
- An *execution* of TS is a finite or infinite sequence

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots$$

where s_0 is the initial state and $(s_i, a_{i+1}, s_{i+1}) \in \mathcal{T}$ for every i.

Inductive Invariance Technique

- Let there be given a concurrent program with atomic actions.
- A state predicate I is said to be *inductive* if
  - I holds for the initial state.
  - Any atomic action a of the program *preserves* I, i.e. for any state s for which I is satisfied, it is either the case that:
    a) a cannot be executed in s, or
    b) the execution of a in state s results in a state s' that again satisfies I.
- If I is inductive, it is an invariant of the program.
- To show a), *known invariants* and may be used.
- If I fails to be inductive, it may be *strenghtened*.