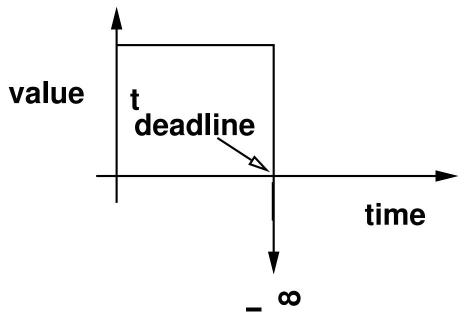
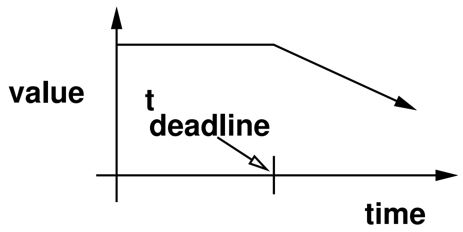
Real-Time Systems

Definition: *Real-time systems are those in which the correctness of the system depends not only on the logical results of the computation but also on the time at which the results are produced.*

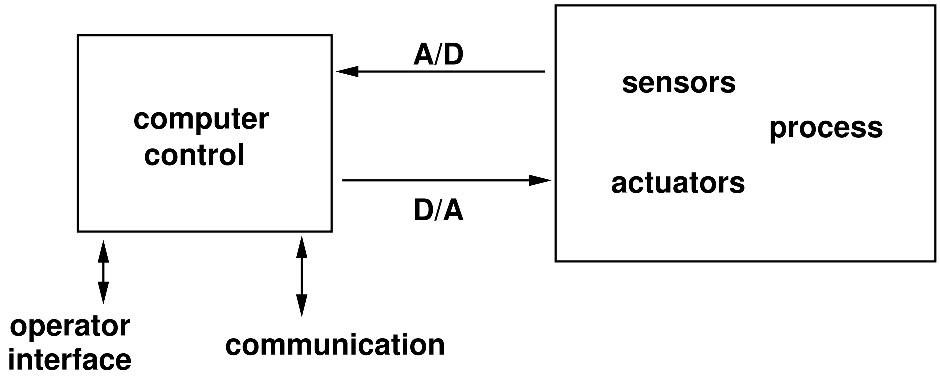
Hard real-time systems: *time critical* - it is imperative that the system respond within a specified time period.



Soft real-time systems: the system will function correctly, although at a lower overall utility if deadline are occasionally missed.



Real-Time Systems



HARD REAL TIME
- distributed process control
- flight-control / avionics
- air traffic control

SOFT REAL TIME
- ticket reservation
- multiuser operating systems
- stock trading

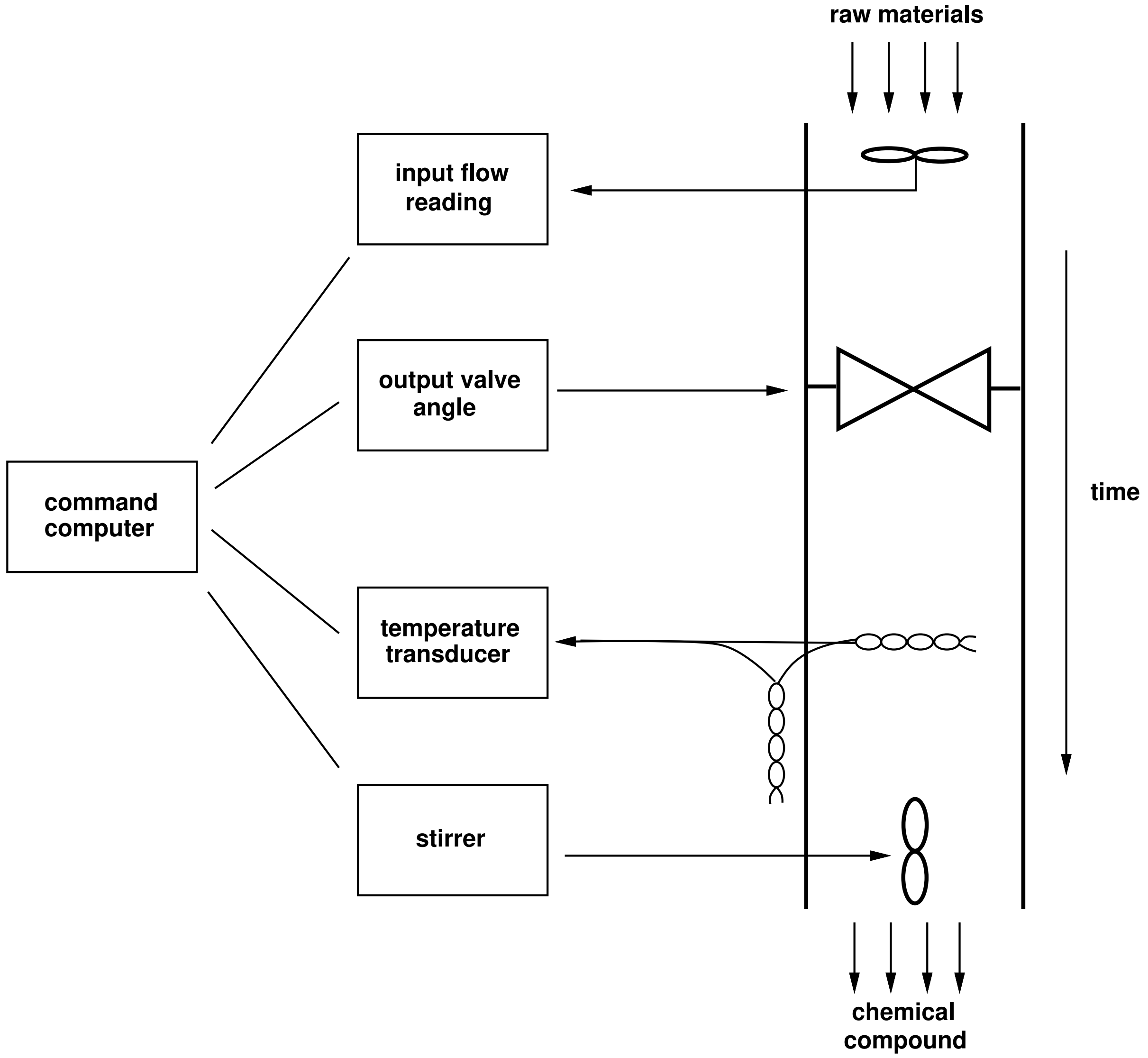
determinism and predictability

$\Downarrow$

techniques for proving that tasks will meet their timing constraints at design time.

A common misconception is that real-time computing equals fast computing. This is not true. A real-time system is fast enough for a particular application and supports temporal analysis.

EXAMPLE: Chemical Process Control



timing constraints:

periodic — once per period T, precisely T units apart

aperiodic — start time, finish time, or both

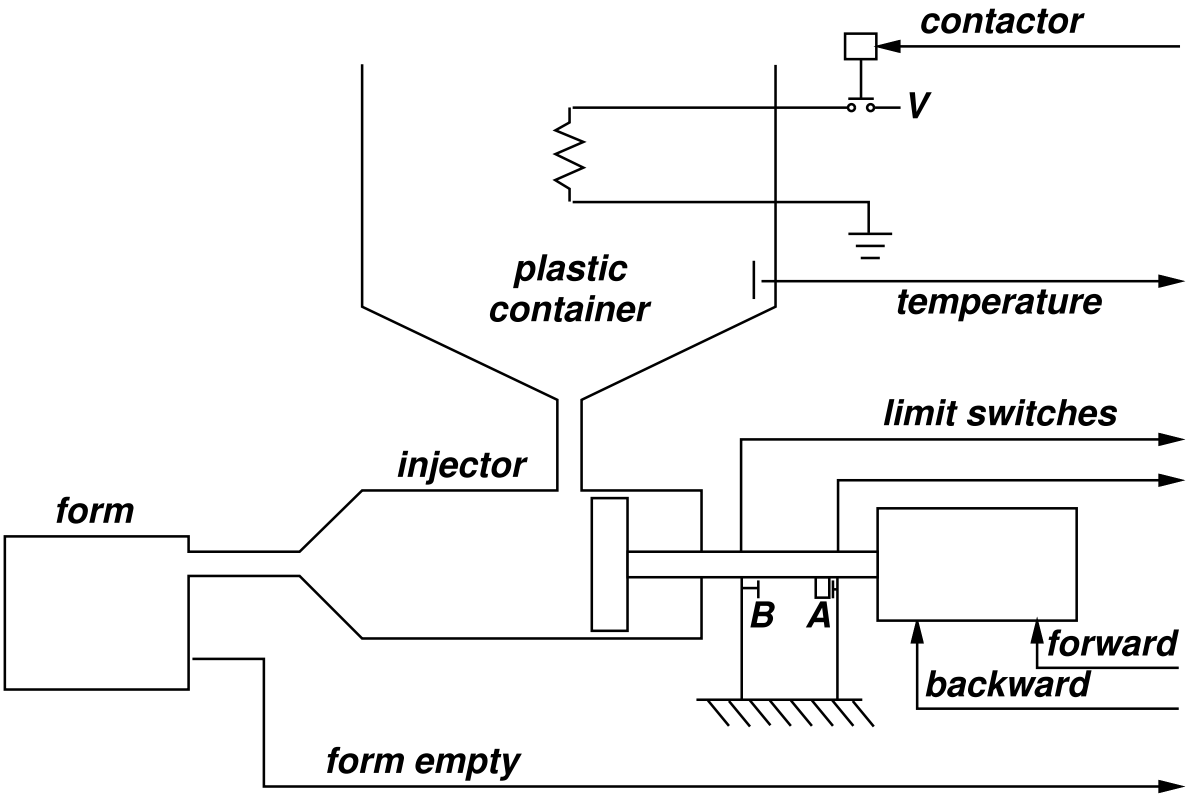
Challenges for RT Systems

1. **variety**
   - multiple operational contexts (critical and sub-critical
   - upgradeable control code
2. **continuous domains**
   - sample and hold - discretized state variables
   - feedback control
   - numerical solutions
3. **reliability and safety**
   - avionics
   - automobiles
   - prosthetics/rehabilitative devices
   - life support / medicine
   - astronautics
   - robotics
4. **real-time facilities**
   - tools for specifying RT constraints
     – times to perform actions
     – times for process completion
     – policies for graceful degradation
     – response to changing timing constraints
   - excess capacity and worst case perfromance prediction

Challenges for RT Systems - cont.

5. **interface with physical devices**
   - direct (not OS-level device drivers)
   - interrupts
   - device registers
6. **efficiency**
   - throughput
   - determinism - minimal computational variance
7. **concurrency**
   - uni-, multi-, and distributed processors
   - sequential programs cumbersome
     - difficult to predict performance
     - difficult to realize design specifications
     - difficult to upgrade / extend
     - difficult to implement fault detection and handling
   - fault tolerance - processing and communication

Example: injection molding



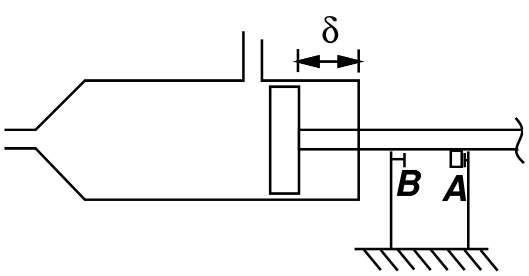
CONSTRAINTS:

- keep plastic at proper temperature (liquid, but not boiling)
- control injector solenoid (make sure that the motion of the solenoid terminates before the piston reaches the end of its travel)

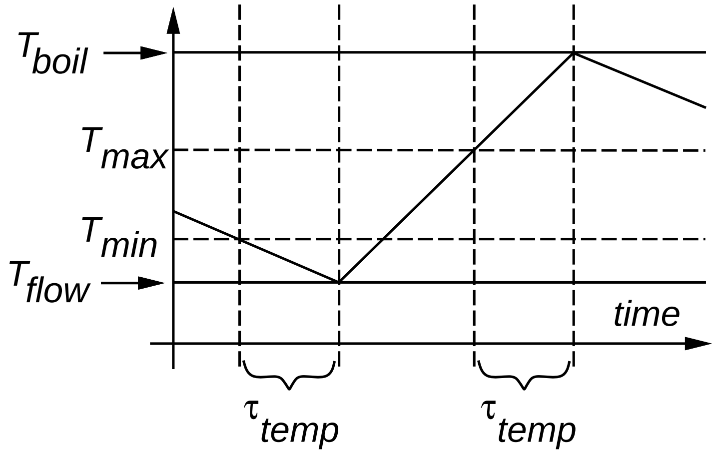
Example: injection molding (cont.)

TIMING CONSTRAINTS:

- the injector must be off within τ_{inj} seconds after A or B limit signals, so that $v_{inj}\tau_{inj} < \delta$



- the temperature control contactor must be activated within τ_{temp} seconds of a temperature event, so that $T_{boil} < T < T_{flow}$



Example: injection molding (cont.)

Concurrent Control Tasks

injector control:

```
in position A;
while(1) {
   wait_until(form_empty);
   on(forward);
   wait_until(B);
   off(forward);
   on(backward);
   wait_until(A);
   off(backward);
}
```

temperature control:

```
while(1) {
   analog_in(Temp);
   if (Temp > Tmax) {
      off(contactor);
   }
   else if (Temp < Tmin) {
      on(contactor);
   }
}
```

Example: injection molding (cont.)

Sequential Program

```
(* initialization *)
while(1) {
   while( !form_empty ) {
      analog_in(Temp);
      if (Temp > Tmax) off(contactor);
      else if (Temp < Tmin) on(contactor);
      delay(τ);
   }
   on(forward);
   while( !B ) {
      analog_in(Temp);
      if (Temp > Tmax) off(contactor);
      else if (Temp < Tmin) on(contactor);
      delay(τ_inj);
   }
   off(forward);
   on(backward);
   while( !A ) {
      analog_in(Temp);
      if (Temp > Tmax) off(contactor);
      else if (Temp < Tmin) on(contactor);
      delay(τ_inj);
   }
   off(backward);
}
```