

**ECE 741/841**

7 November 2002

## Modeling Time

- Most system models have some notion of time.
- Time can be abstracted, discrete(time steps), or continuous.
- Time can be qualitative or quantitative.
  - event1 occurs after event2.
  - event1 occurs 52.3 seconds after event2.

## Abstracted Time

- We can abstract time out of a model when time is not being considered.
- Time might be abstracted because:
  - it is not relevant,
  - it will be considered in another model,
  - to simplify the model,

## Abstracted Time, Example

- In the model of the ripple carry adder, time was not considered.
- Only the functionality of the adder was specified, modeled, and verified.
- One problem with a ripple carry adder is that it **is slow**; the speed of the operation is determined by the speed of the one bit adder times the length of the word.

## Discrete Time

- Time can be discrete in a model by assigning increments of time to transitions.
- Transitions can have arbitrary units.
  - One gate delay.
  - One clock cycle.
  - One transition.
- These units can be related to a concrete time at a lower level of abstraction.
  - Example: One gate delay equals 1.3 nanoseconds.

## Discrete Time, Example

%-- Model of trajectories

```
next_state(s,phi): State =  
    s WITH [  
        x          := x(s) + v(s)*tstep*cos(heading(s)),  
        y          := y(s) + v(s)*tstep*sin(heading(s)),  
        heading   := heading(s) + tstep*trkrate(phi(s)),  
        phi        := phi  
    ]  
  
trajectory(s,df,n): RECURSIVE State =  
    IF n = 0 THEN s  
    ELSE  
        next_state(trajectory(s, df, n-1),df(n))  
    ENDIF  
    MEASURE n
```

## **Continuous Time**

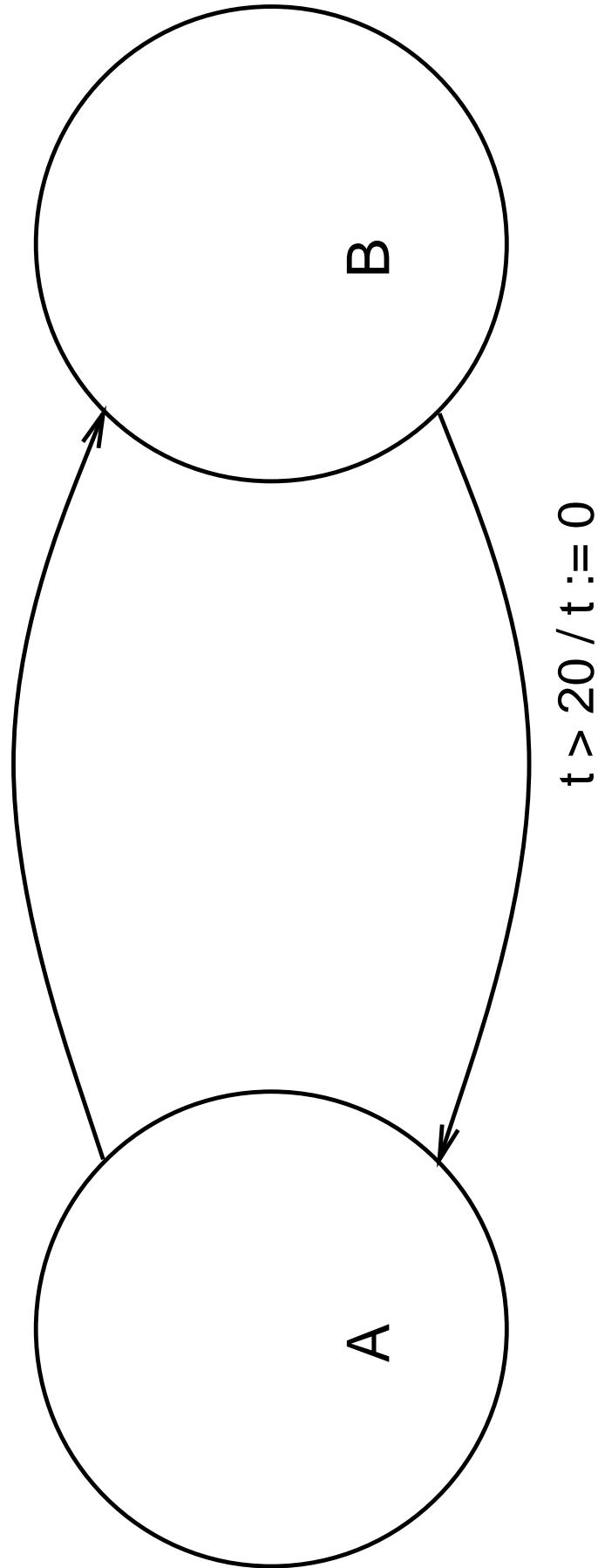
- Logic (propositional, first, and higher-order) is mostly suitable to represent discrete behavior.
- It is possible to represent continuous behavior within the framework of the higher-order logic language.
- The trajectory model has been defined in the PVS system using the notion of differentiability
- This is an area of research at this time.

## Continuous Time, Example

```
Trajectory : type = {tr : [# x,y,theta,phi : (Differentiable) #] |  
FORALL(t:real) : (  
    D(x(tr))(t) = v * cos(theta(tr)(t)) AND  
    D(y(tr))(t) = v * sin(theta(tr)(t)) AND  
    -PI/4 <= phi(tr)(t) AND  
    phi(tr)(t) <= PI/4 AND  
    D(theta(tr))(t) = g*tan(phi(tr)(t))/v)  
)
```

## Finite State Machine with Time

$t > 30 \text{ AND } \text{sensor\_hit} / t := 0$



## Description in PVS, I

```
two_state : theory
begin

state : type = {A,B}
events : type = {null,sensor_hit}
t : var real
e : var events
s : var state

next_state(s,t,e) : [state,real] =
if s=A then
  if t > 30 AND e=sensor_hit then (B,0)
else (A,t) endif
else % s=B
  if t > 20 then (A,0)
else (B,t) endif
endif
```

```
good_state(s,t) : bool =  
s = B implies t <= 20  
  
good_state_next : theorem  
forall s,t,e : good_state(next_state(s,t,e))  
  
end two_state
```

## Description in PVS, II

```
two_state_plus : theory
begin

c_state : type = {A,B}

state : type = [# cs:c_state,
              t:real
#]

events : type = {null,sensor_hit}

e : var events
s : var state
```

```

next_state(s,e) : state =
  if cs(s)=A then
    if t(s) > 30 AND e=sensor_hit then (#cs:=B,t:=0#)
    else (#cs:=A,t:=t(s)#) endif
  else % s=B
    if t(s) > 20 then (#cs:=A,t:=0#)
    else (#cs:=B,t:=t(s)#) endif
  endif

good_state(s) : bool =
  cs(s)=B implies t(s) <= 20

good_state_next : theorem
forall s,e : good_state(next_state(s,e))

end two_state_plus

```

## Description in PVS, III

```
two_state_plus2 : theory
begin

c_state : type = {A,B}

state : type = [# cs:c_state,
               t:real
               #]

events : type = {null,sensor_hit}

e : var events
s,s0 : var state
ef : var [posnat -> events]
n : var nat
m : var posnat
```

```

next_state(s,e) : state =
if cs(s)=A then
  if t(s) > 30 AND e=sensor_hit then (# cs:=B,t:=0 #)
    else (# cs:=A,t:=t(s) + 1 #) endif
  else % s=B
    if t(s) > 20 then (# cs:=A,t:=0 #)
      else (# cs:=B,t:=t(s) + 1 #) endif
    endif
endif

step(s0,ef,n) : recursive state =
if n = 0 then s0
else
  next_state(step(s0,ef,n-1),ef(n))
endif

measure n

```

```
start_B_22_A : theorem
  cs(s0) = B AND t(s0) = 0 implies
    cs(step(s0,(lambda m:null),22)) = A
end two_state_plus2
```