Discrete Event Simulation

Process Oriented Simulation

Richard M. Fujimoto
Professor

Computational Science and Engineering Division
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0765, USA

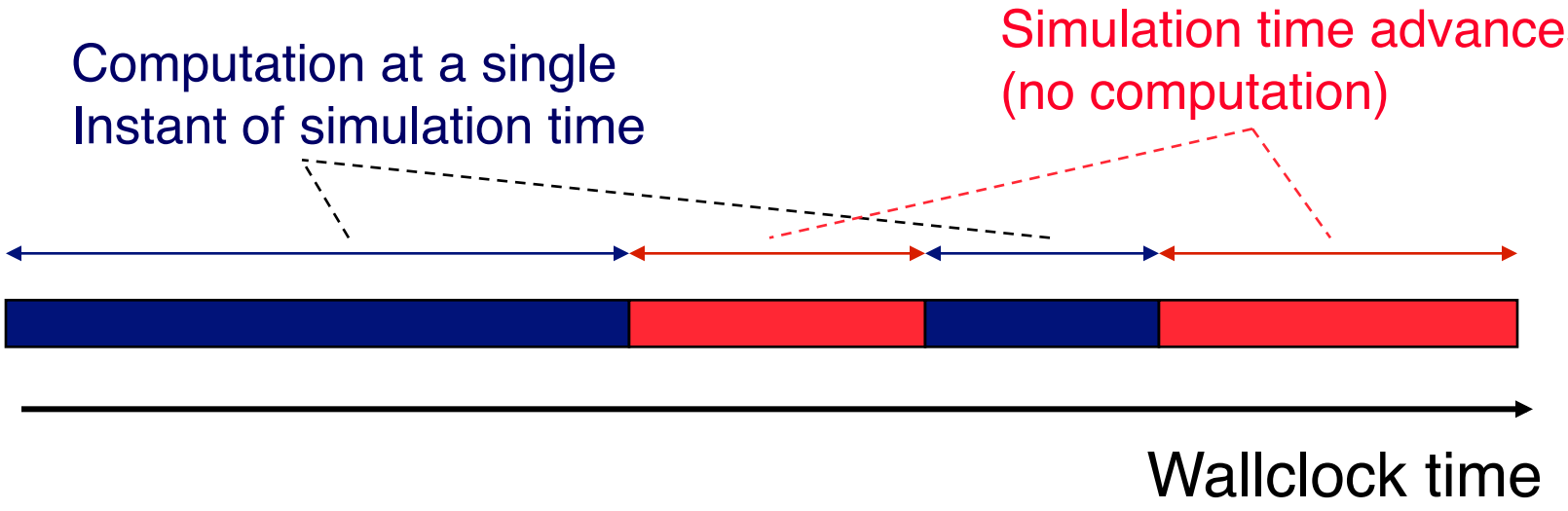
http://www.cc.gatech.edu/~fujimoto/

Outline

- Fundamental concepts: Processes, resources
- Simulation primitives
- Example
- Implementation

Process Oriented Simulation

- Focus simulation program around behavior of entities
  - Aircraft: arrives, waits for runway, lands, departs
- Process-oriented simulation
  - Process: thread of execution describing entity behavior over time
  - Resources: shared resource used by entities (e.g., runway)
- Execution: alternate between
  - simulation computations at a single instant of simulation time, and
  - advances in simulation time (no computation)



Event vs. Process Oriented Views

Event oriented view

State variables

```
Integer: InTheAir;
Integer: OnTheGround;
Boolean: RunwayFree;
```

```
Arrival Event
{
   …
}
```

```
Landed Event
{
   …
}
```

```
Departure Event
{
   …
}
```

Entities modeled by event handlers

Process oriented view

State variables

```
Integer: InTheAir;
Integer: OnTheGround;
Boolean: RunwayFree;
```

```
Aircraft 1
{
   Arrive
   Land
   Depart
}
```

```
Aircraft 2
{
   Arrive
   Land
   Depart
}
```

```
Aircraft n
{
   Arrive
   Land
   Depart
}
```

Entities modeled by processes

Simulation Primitives

Primitives needed to advance simulation time

- AdvanceTime(*T*): advance *T* units of simulation time
  - Also called "hold"
  - E.g.: AdvanceTime(R) to model using runway R units of simulation time
- WaitUntil(*p*): simulation time advances until predicate *p* becomes true
  - Predicate based on simulation variables that can be modified by other simulation processes
  - E.g.: WaitUntil(RunwayFree) to wait until runway becomes available for landing
- Other combinations
  - WaitUntil(p,T): Wait up to T units of simulation time for predicate p to become true
  - Not used in the air traffic example

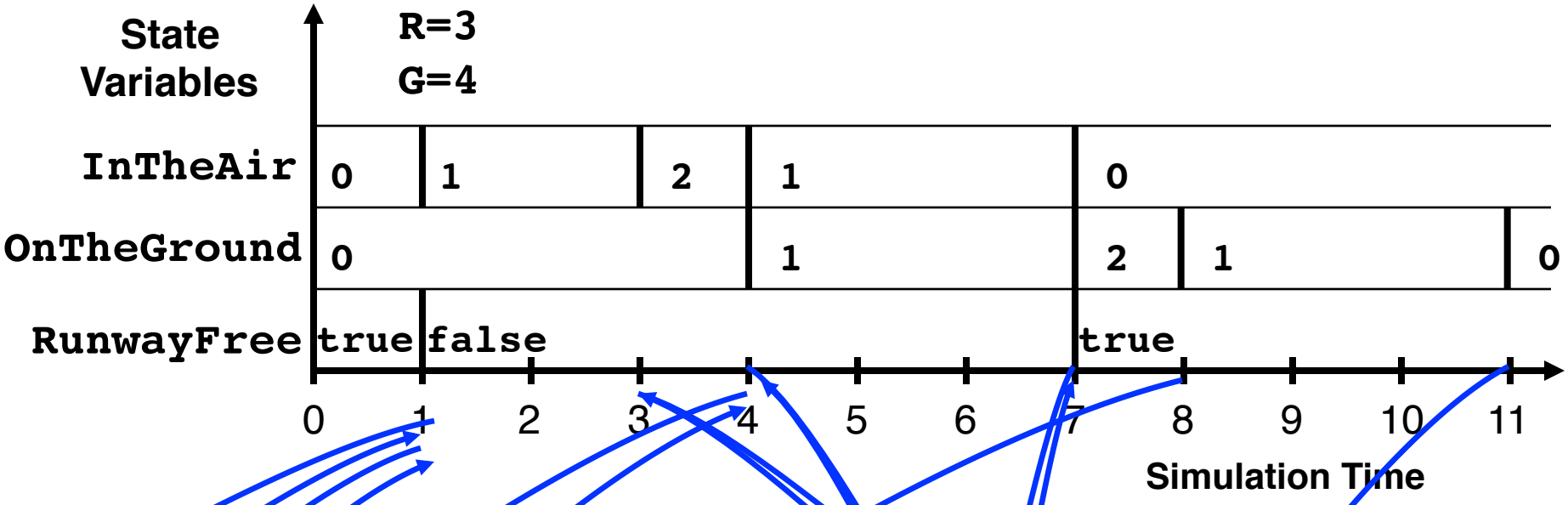
Process Model Example: Aircraft

A new aircraft process is created with each Arrival event

```
   /* simulate aircraft arrival, circling, and landing */
Integer: InTheAir;
Integer: OnTheGround;
Boolean: RunwayFree;

1  InTheAir := InTheAir + 1;
2  WaitUntil (RunwayFree); /* circle */
3  RunwayFree := FALSE;    /* land */
4  AdvanceTime(R);
5  RunwayFree := TRUE;
   /* simulate aircraft on the ground */
6  InTheAir := InTheAir - 1;
7  OnTheGround := OnTheGround + 1;
8  AdvanceTime(G);
   /* simulate aircraft departure */
9  OnTheGround := OnTheGround - 1;
```

Execution Example




```
Flight 1
1 InTheAir := InTheAir+1;
2 WaitUntil (RunwayFree);
3 RunwayFree := FALSE;
4 AdvanceTime(R);
5 RunwayFree := TRUE;
6 InTheAir := InTheAir-1;
7 OnTheGround:=OnTheGround+1;
8 AdvanceTime(G);
9 OnTheGround:=OnTheGround-1;
```

```
Flight 2
1 InTheAir := InTheAir+1;
2 WaitUntil (RunwayFree);
3 RunwayFree := FALSE;
4 AdvanceTime(R);
5 RunwayFree := TRUE;
6 InTheAir := InTheAir-1;
7 OnTheGround:=OnTheGround+1;
8 AdvanceTime(G);
9 OnTheGround:=OnTheGround-1;
```

Implementation

Process-oriented simulations are built over event oriented simulation mechanisms (event list, event processing loop)

- Event computation: computation occurring at an instant in simulation time
  - Execution of code section ending with calling a primitive to advance simulation time
- Computation threads
  - Typically implemented with co-routine (threading) mechanism
- Simulation primitives to advance time
  - Schedule events
  - Event handlers resume execution of processes

Aircraft Process

Identify computation associated with each simulation event

```
   /* simulate aircraft arrival, circling, and landing */
1  InTheAir := InTheAir + 1;
2  WaitUntil (RunwayFree); /* circle */
3  RunwayFree := FALSE;    /* land */
4  AdvanceTime(R);
5  RunwayFree := TRUE;

   /* simulate aircraft on the ground */
6  InTheAir := InTheAir - 1;
7  OnTheGround := OnTheGround + 1;
8  AdvanceTime(G);

   /* simulate aircraft departure */
9  OnTheGround := OnTheGround - 1;
```

Aircraft Arrival

Aircraft Landing

Aircraft On The Ground

Aircraft Departs

Implementation: AdvanceTime(T)

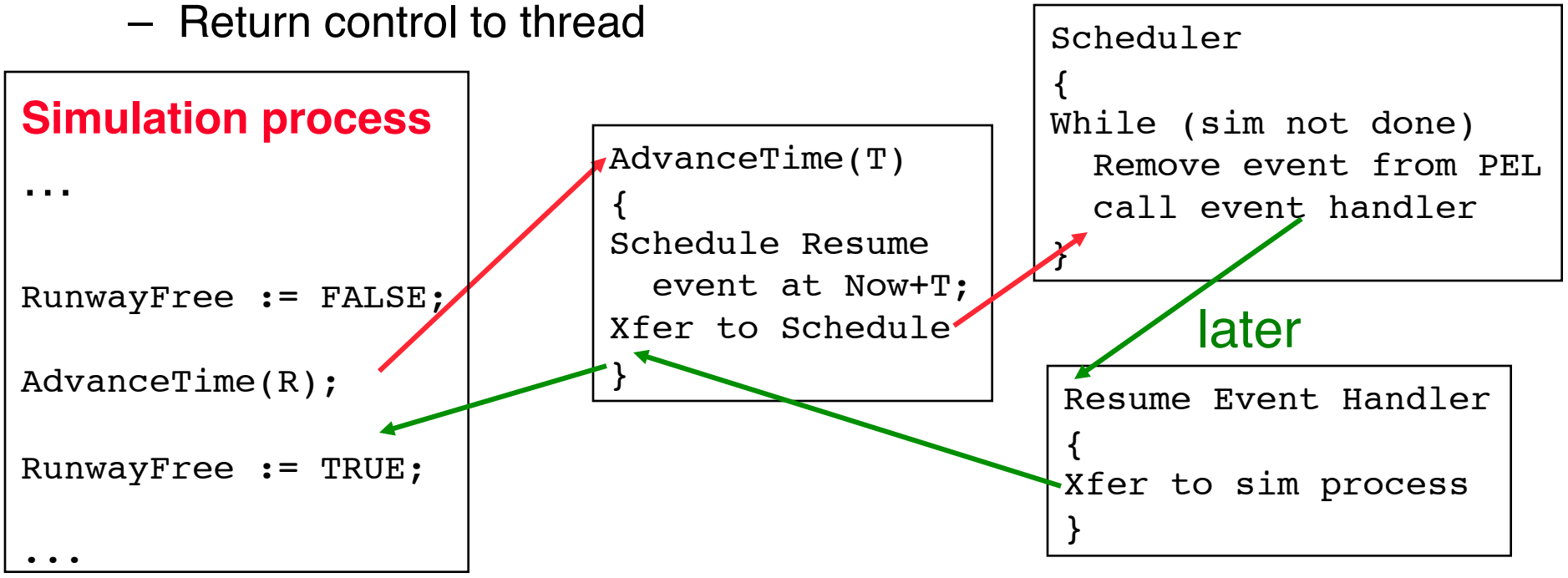
Causes simulation time in the process to advance by T units

Execute AdvanceTime(T):

- Schedule Resume event at time Now+T
- Suspend execution of thread
- Return execution to event scheduler program

Process Resume event:

- Return control to thread



Implementation: WaitUntil (p)

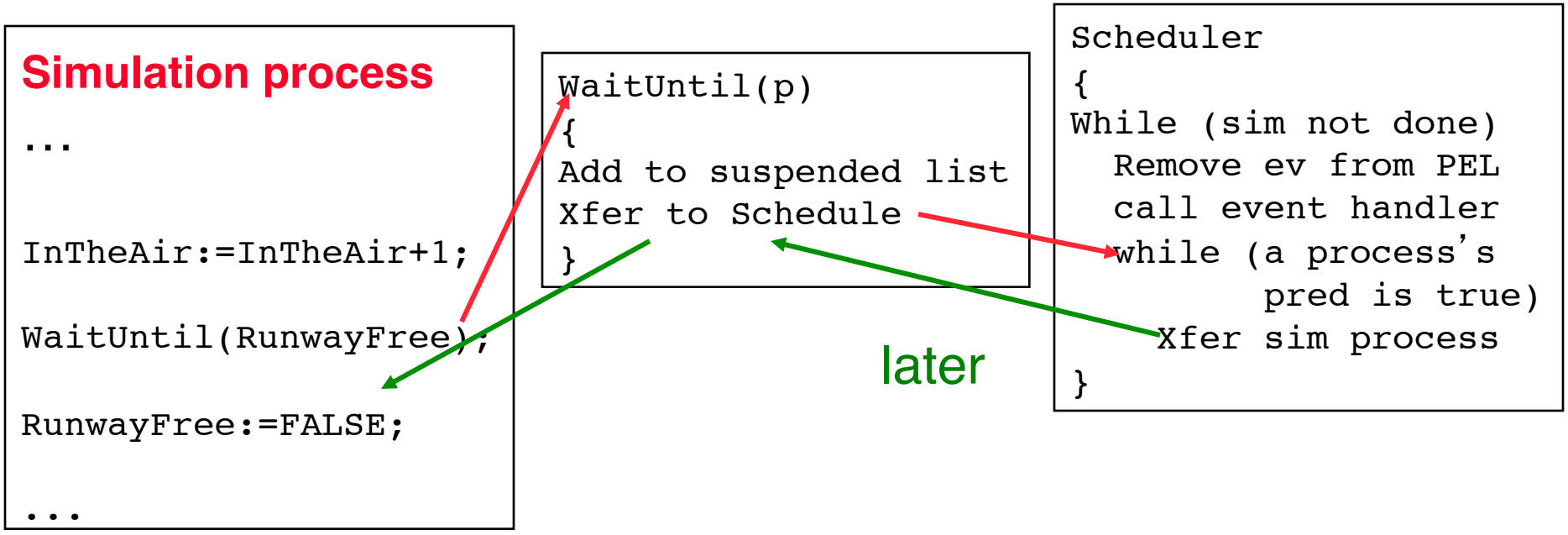
Suspend until predicate p evaluates to true

Execute WaitUntil (p):

- Suspend execution of thread, record waiting for p to become true
- Return execution to event scheduler program

Main scheduler loop

- For each suspended process, check if execution can resume
- Prioritization rule if more than one can resume



Additional Notes

- Theoretically, both views are equivalent:
  - Process-oriented simulations can be transformed to event-oriented simulations and vice versa
  - How?
- Practically, runtime performance differs:
  - Event-oriented views typically execute faster than process-oriented views
  - Why?

Summary

- Process-oriented simulation typically simplifies model development and modification
- Requires threading (e.g., co-routine) mechanism
- Additional complexity and computation overhead to suspend and resume simulation processes