CAP 4800/5805 Computer Simulation Concepts

Lecture 20 and 22

Production-Based Models (continues)

Beetle population growth

- The following first-order logic expressions represent a logistic growth rate of beetles (let p = population size, g = population growth, and t = current time)

$$t(t_1) \wedge p(small, t_1) \wedge \mathrm{g}(slow, t_1) \supset t(t_2) \wedge (t_2 > t_1) \wedge p(medium, t_2) \wedge g(fast, t_2)$$
$$\supset t(t_3) \wedge (t_3 > t_2) \wedge p(steady, t_3) \wedge g(slow, t_3)$$

- Even though first-order logic provides one method of incorporating time into modeling, it is preferable to have a language that has a better integrated concept of time advance
- The class of languages that specify time as an integral component are known as **temporal logic programming languages**.

Tempura

- Tempura is one of temporal logic programming languages
- Based on the concept of truth within a time interval σ. σ is an interval with a discrete number of states $< \sigma_0, \sigma_1, \ldots, \sigma_{|\sigma|-1} >$
- The output is a list of variable values with two distinctions when compared against the Prolog output: (1) the variables are termed “states”, and (2) an interval of states is output rather than a single state value list

Operators

- $\square e$ (always). Expression e is true throughout the interval. This is interpreted as “for all time values”

- $\Diamond e$ (sometimes). Expression *e* is true at some time in the interval. This is interpreted as "for at least one time value"
- $\bigcirc e$ (next). Expression *e* is true at the next time step.
- $e_1; e_2$ (chop). There is an interval which can be subdivided into two sub-intervals so that e_1 holds in the first subinterval and e_2 holds in the second subinterval.

Example

Consider the following interval σ:

	X	*Y*
σ_0	4	5
σ_1	2	5
σ_2	3	1
σ_3	5	2
σ_4	8	2

- The interval is defined as $<\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4>$ and is of length 5
- The state space is two-dimensional, where (*X*, *Y*) is a point in the state space
- $\Diamond(X = 2 \wedge Y = 5)$ is true since state σ_1 satisfies this expression
- While $\Box(Y = 5)$ isn't true on the above interval, it is true on the subinterval of length two: $\sigma' = <\sigma_0, \sigma_1>$
- $\Diamond\bigcirc(X > 5)$ is true – it states that at some point in the interval the next *X* component of the state will be greater than 5
- Here are some additional expressions that are true for interval σ:
    1. $\Diamond(X = 2) \wedge \Diamond(Y = 1)$. True for *X* in σ_1 and Y in σ_2
    2. $\bigcirc\bigcirc\bigcirc(Y = 2)$. True for *Y* in σ_3
    3. $\Box(X > 1 \wedge X < 9)$. True for all states in σ

4. $\square(X < 5); \square(Y < 4)$. Subinterval $<\sigma_0, \sigma_1>$ concatenated with $<\sigma_2, \sigma_3, \sigma_4>$ satisfies this compound expression. Another possibility is $<\sigma_0, \sigma_1, \sigma_2>$ concatenated with $<\sigma_3, \sigma_4>$

Event-Based Approach

Since events are the counterpart components of states – since states are nothing more than what occurs between events – we should be able to construct modeling methods that focus on events.

Finite Event Automata (FEA)

- FEA models represent the event analog to state automata
- The formalism for an event graph is identical to that of an FSA except that (1) nodes in the graph represent events instead of states, and (2) transitions (no the states) are labeled with time durations used for scheduling future events
- Events change the state of a system, but the events in Fig. 4.4 (Lecture 13) are not shown explicitly
- An FSA is a model type that emphasizes states instead of events
- Figure 4.25 shows the events that are associated with the states in Fig. 4.5 (Lecture 13)
- An event graph may be generated automatically from the state graph in Fig. 4.4 by:
    - Taking the times of the states, and assigning them to the respective event graph transitions
    - Taking the events in Fig. 4.25 and using these instead of the state names

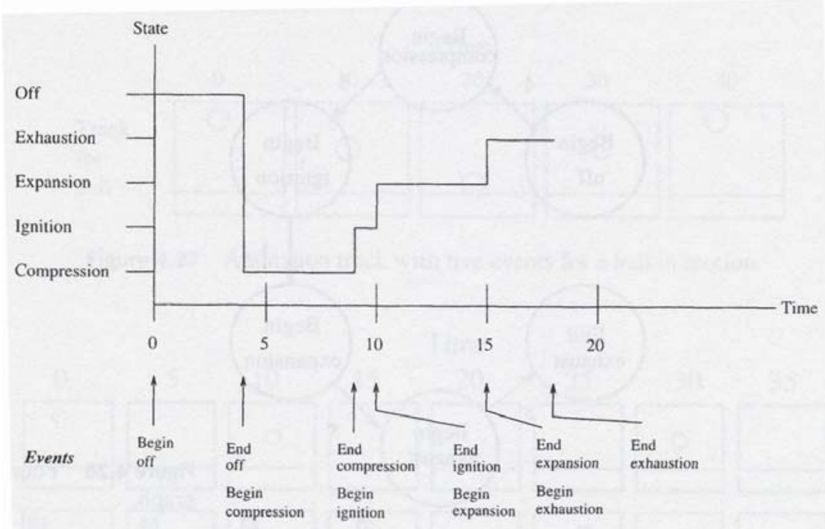

Figure 4.25 State trajectory with events for the engine model.

- **Four stroke engine event graph**