5. Iterating With MP (Repetitive Event Patterns)

Iteration in MP

Suppose that you have a class which meets more than one time. This is represented in MP by using repetitive event patterns. Repeated events are the analogy to "loops" in other programming languages. Both set ('{' ...'}') and sequence ((...)) iteration are supported. The examples show sequence iteration but are equally applicable to set iteration.

Scope

This is a good time to explain the concept of scope which is basically how many times (maximum) each iteration (repetitions indicated by the (*…*), (+…+) syntax) can run. Scope only makes sense if there is some iteration in your model which is not always necessary. Be careful with scope as you can quickly generate a LOT of traces from even a moderately sized model.

Always use scope 1 (default setting) to get started in your modeling. Unlike other programming languages, note that scope settings are controlled when you run your model. If you need some restrictions on scope they are indicated separately in the model via ENSURE statements (which filter precluded conditions from your traces).

The various types of iteration are discussed below.

"For" Style Looping

"For" style looping iterates some specified number of times. This is in contrast to "while" style looping which repeats as long as or until some specified condition is true. While looping is covered a little bit later in this tutorial.

MP syntax allows two choices with respect to specifying iteration:

- (* My_event *) which means my_event can happen zero or more times
- (+ My_event +) which means my_event happens at least one time
- Refer to section 5.4 in the MP manual for more information.

At least once (+ ... +) iteration

Example 8 Attend_class with one or more iteration

ROOT Student: (+ Attend_class +)

[(Go_to_library|Visit_with_friends)];

/* this is a composite event definition */

Attend_class: Find_a_good_seat

Plug_in_laptop

/* these activities may happen in parallel */

{ Listen_to_instructor,

Make_notes,

Check_your_email_during_a_pause }

[Ask_prof_a_question]

Leave_the_classroom;

Clicking on run (scope still at default 1) produces 6 traces: (two flavors of Attend_class) x (three flavors of post class activities). Pulling to the side allows you to clearly see the concurrent events in the model.

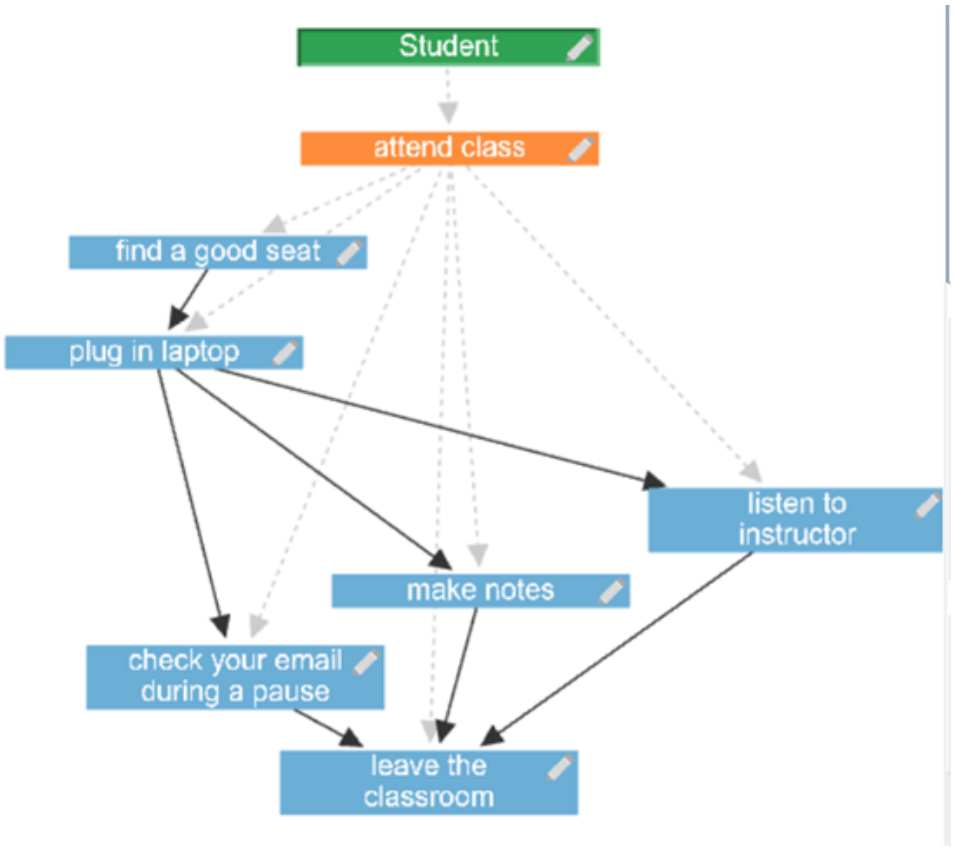

Figure 12 Attend_class with iteration - trace #1

In the previous Attend_class example set the scope to 2 using the slider tool at the top of the model (shown below as circled). The syntax change for 1 or more iterations is shown highlighted.

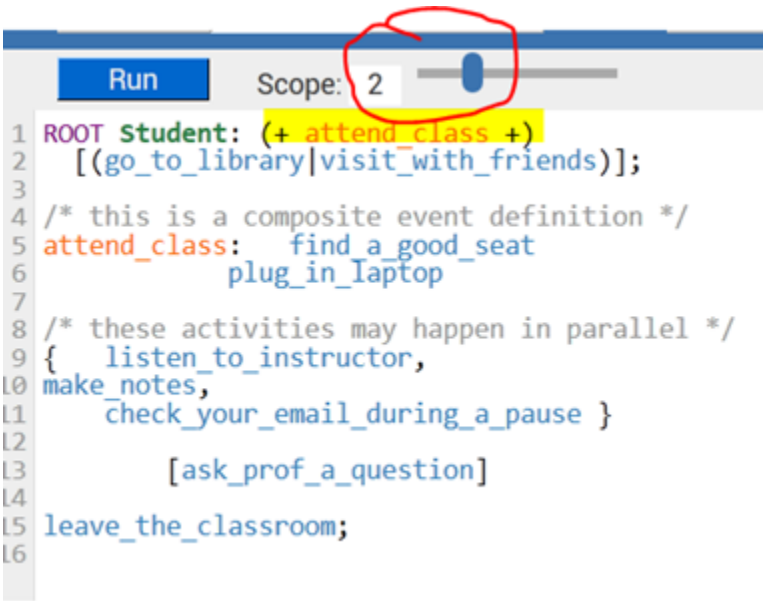

Figure 13 Choosing scope from the GUI

MP produces 18 traces. Notice that the first 6 traces are scope 1. The next 12 traces have Attend_class happening exactly twice. The difference in the various traces is caused by the optional events happening or not.

Zero or more (* … *) iteration

If the repetition of Attend_class is changed to zero or more (as indicated below) and scope 2 is run, you now get 21 traces. This is the prior 18 cases plus the three cases corresponding to not going to any classes.

Example 9 zero or more iterations

ROOT Student: (* Attend_class *)

[(Go_to_library|Visit_with_friends)];

/* this is a composite event definition */

Attend_class: Find_a_good_seat

Plug_in_laptop

/* these activities may happen in parallel */

{ Listen_to_instructor,

Make_notes,

Check_your_email_during_a_pause }

[Ask_prof_a_question]

Leave_the_classroom;

Note: The difference between the (*…*) syntax and the [ ] syntax is nothing *if you run at scope one* (single iteration maximum) since the [ ] really means 0 or 1 instances of this event. So, you run at a higher scope, Go_to_library still will run only 0 or 1 time, but Attend_class can appear any number of times up to the scope.

"While" and "Repeat Until" Style Looping

From the MP Guide Section 2.3:

Behavior of the loop

while(Condition) do Loop_body

can be modeled (in pseudo-code) as

Check_Condition

(* Condition_is_true

Perform_Loop_body

Check_Condition

***)**

Condition_is_false

In MP syntax this modeled as a composite event as shown in the example:

Example 10 while looping

ROOT Loop_Example: loop_while;

loop_while:

(* Condition_is_true

Perform_Loop_body

Check_Condition *)

Condition_is_false ;

Running this on scope 3 produces four traces: one for (never executes loop body), and then one for each number of possible iterations (as specified by scope setting) through the loop body.

The (*…*) syntax means that the loop could possibly be executed zero times. In contrast a "repeat…until" loop is always executed at least once, which is implemented the same way as the above example, but uses the (+…+) repetition syntax.

next page (event interactions)