Fixed-Priority Preemptive Scheduling Semantics of AADL in UPPAAL Timed Automata

Andreas Johnsen

School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
andreas.johnsen@mdh.se

The scheduling automaton providing the required thread execution semantics is shown in Figure 1. The labels of the scheduling automaton are defined as follows:

- (int)ready_queue[x]: is a sorted queue of currently dispatched threads. The queue is sorted according to a given scheduling policy where the first element in the queue (x=0) is the (identifier of the) thread being processed and where the second element is the next thread to be processed, and so forth.
- (clock)sch_clocks[x][2]: is a list of clocks in sets of two, each set referenced by an identifier x of a currently dispatched thread. Each dispatched thread has two clocks, the first (sch_clocks[x][0] of thread with identifier x) is used to keep track of a thread's execution time, and the second (sch_clocks[x][1] of thread with identifier x) is used to keep track of a thread's deadline.
- (int)sch_info[x][3]: is a list of threads' scheduling properties (integers) in sets of three, each set referenced by an identifier x of a currently dispatched thread. Each dispatched thread has three scheduling properties, the first (sch_info[x][0] of thread with identifier x) is the execution time, the second (sch_info[x][1] of thread with identifier x) is the deadline, and the third (sch_info[x][2] of thread with identifier x) is the priority. Note that the required properties are related to a given scheduling policy. For example, we consider priorities of threads since we assume a fixed priority scheduler in this particular example.
- (int)preempt_stack[x][2]: is a stack of sets of currently preempted threads (integer identifiers) and the amount time each thread has been preempted. Given a stack of preempted threads, the first set of elements in the stack (preempt_stack[0][0] is the thread identifier and preempt_stack[0][1] is the amount of time) corresponds to the thread that first was preempted.
- (int)nr_preempted: number of currently preempted threads.
- (int)threads: number of currently dispatched threads.
- (int)check_preempt: holds the identity of a thread that is dispatched at the same time as another thread is running. It is used to check if the dispatched thread preempts the running thread.
- (chan)dispatched[(int)x],(chan)run[(int)x], (chan)complete[(int)x],(chan)preempt[(int)x]: are channels used to synchronize every thread transition of every thread in the system. Synchronization

with a particular thread is done through its identity. For example, run[2] is a synchronization channel with thread having identity equal to 2.

- (void)schprotocol((int)x): is a function sorting threads in the ready_queue according to a given scheduling policy. The function is called each time a thread dispatches where the thread's identity is given as argument to the function. In this example, we assume fixed priority scheduling.
- (void)completion((int)x): is a function removing threads from the ready_queue. The function is called each time a thread completes its execution, where the thread's identity is given as argument to the function.
- (void)addTime(): is a function adding preempted time to the threads in the preempt_stack. The function is called when a preemption occurs, whereupon the execution time of the thread causing the preemption is added to the preemption time of every preempted thread.
- (void)checkTime((int)x): is a function adding preempted time to the threads in the (int)preempt_stack[x][2] stack. The function is called when a thread-dispatch not causing any preemption occurs, to check if the dispatched thread is prior to any preempted threads in the ready_queue whereupon preemption time is added.

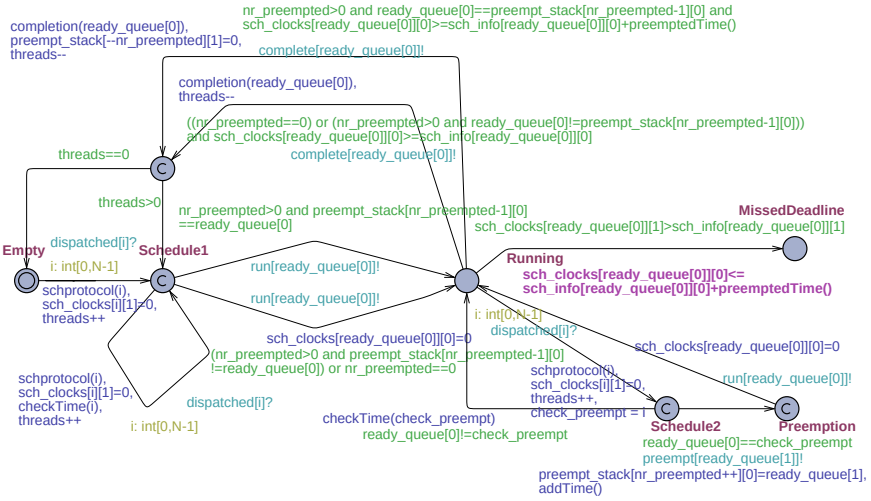

Fig. 1. The scheduler automaton.

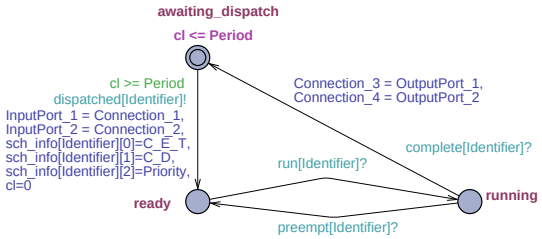

Fig. 2. Example of a thread automaton controlled by the scheduler