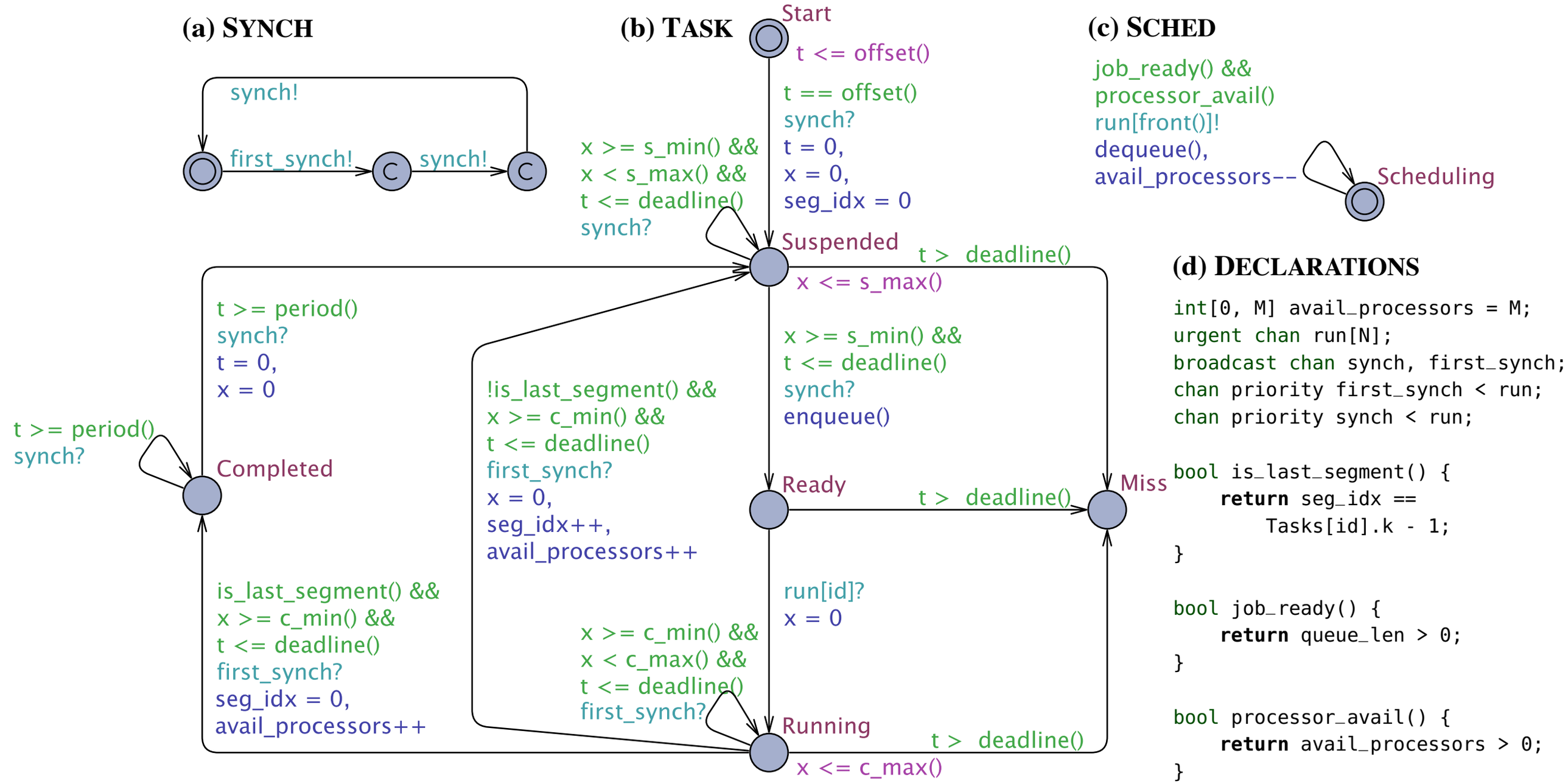
(a) SYNCH
synch!
first_synch!
synch!
(b) TASK
Start
t <= offset()
t == offset()
synch?
t = 0,
x = 0,
seg_idx = 0
x >= s_min() &&
x < s_max() &&
t <= deadline()
synch?
Suspended
t > deadline()
x <= s_max()
x >= s_min() &&
t <= deadline()
synch?
enqueue()
Ready
t > deadline()
Miss
run[id]?
x = 0
x >= c_min() &&
x < c_max() &&
t <= deadline()
first_synch?
Running
t > deadline()
x <= c_max()
t >= period()
synch?
t = 0,
x = 0
Completed
t >= period()
synch?
!is_last_segment() &&
x >= c_min() &&
t <= deadline()
first_synch?
x = 0,
seg_idx++,
avail_processors++
is_last_segment() &&
x >= c_min() &&
t <= deadline()
first_synch?
seg_idx = 0,
avail_processors++
(c) SCHED
job_ready() &&
processor_avail()
run[front()]!
dequeue(),
avail_processors--
Scheduling
(d) DECLARATIONS
int[0, M] avail_processors = M;
urgent chan run[N];
broadcast chan synch, first_synch;
chan priority first_synch < run;
chan priority synch < run;
bool is_last_segment() {
return seg_idx ==
Tasks[id].k - 1;
}
bool job_ready() {
return queue_len > 0;
}
bool processor_avail() {
return avail_processors > 0;
}