Problem Sheet 10: Verification of Real-Time Systems
Sample Solutions

Chris Poskitt and Carlo A. Furia
ETH Zürich

Starred exercises (∗) are more challenging than the others.

1 MTL Property Checking

i. Yes: it simply means that a holds at every position in the timed word (if any).

ii. No: this requires that there is always a future position, 1 time unit in the future, where a holds; but this is not the case in the last position of any (non-empty) timed word. (The only position that can be reasoned about relative to the end position *is* the end position, which is exactly 0 time units in the future.) A counterexample is the timed word $(a, 1.0)$ $(a, 2.0)$.

iii. Yes: the formula requires that *if* there is a future position 1 time unit in the future, *then* a holds there.

iv. Yes: the clock x is reset after reading a. After that, it is checked upon reading c, the clock constraint requiring that x is in the range $(0, 1)$.

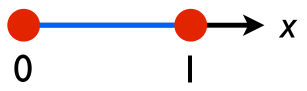
v. Yes: the clock x is reset after reading a. After that, it is always checked upon reading c, which is always preceded by reading a b. (If b occurred 1 or more time units after the reading of a, then the same would occur for the reading of c. But this would violate the clock constraint.)

vi. Yes: the clock x is reset after reading a. Then there is a reading of b followed by c, satisfying the sequence of events required by the until formula. Timing argument analogous to previous questions.

vii. No. A counterexample is the timed word $(a, 1.0)$ $(b, 1.2)$ $(c, 1.3)$.

2 Region Automaton Construction

i. For illustration, we first draw the clock regions associated with the timed automaton. Since there is only one clock x, and because the maximum constant in the clock constraints is 1, our diagram is very simple:

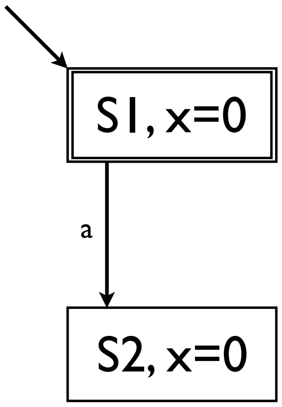


The initial and accepting state of the region automaton will be $(S1, x = 0)$. To determine the outgoing edges from this state, we first determine the *time successors* of the region $x = 0$. This is the set of clock regions that can be reached from $x = 0$ by letting time pass, i.e.

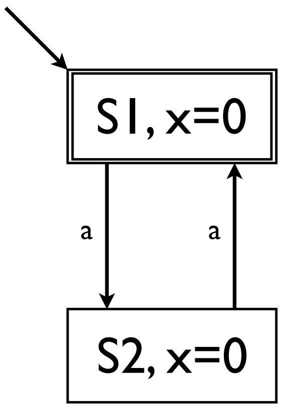
$$0 < x < 1$$
$$x = 1$$
$$x > 1$$

(We don't break $x > 1$ into smaller clock regions because the largest constant in the clock constraints is 1.)

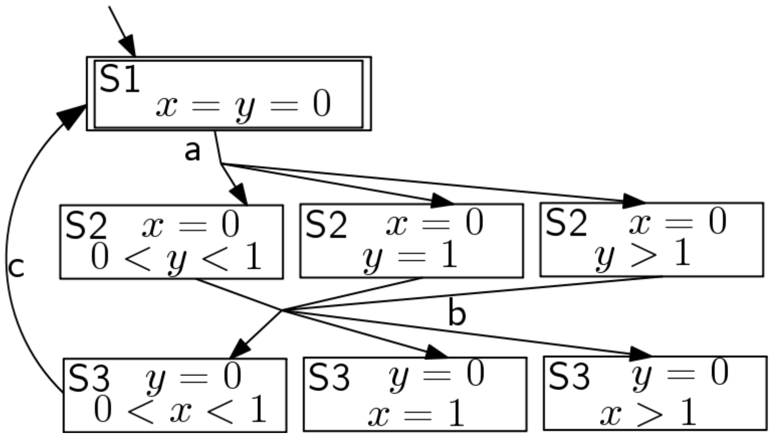
In the original timed automaton we can reach state $S2$ from $S1$ exactly when $x = 1$ and the next position of the timed word is a. One might think that we should hence add to the region automaton an edge from $(S1, x = 0)$ to $(S2, x = 1)$ on a. However, the original automaton resets x to 0, so instead of adding an edge to $(S2, x = 1)$, we add an edge to $(S2, x = 0)$, i.e.



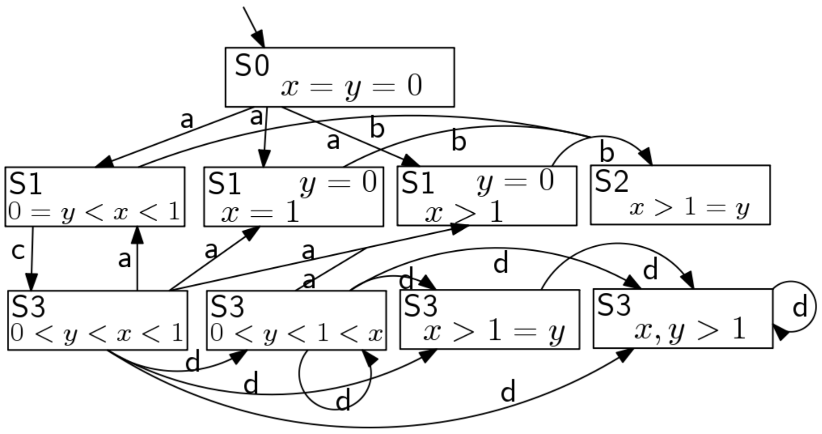
Through similar reasoning, for the other edge in the original timed automaton, we complete the region automaton (omitting the non-reachable states):



ii. Through a similar process to the previous question we get the (somewhat larger!) region automaton that follows. You can construct it more efficiently by noting that at least one clock is reset on each transition (e.g. for the first transition, the corresponding regions will all have $x = 0$ but varying y).



iii. (∗)



3 Semantics of MTL Formulae

i. The empty word satisfies $\Box\ \Diamond > 0$ true, but the same is not true for non-empty words ($\Diamond\!>\!0$ true does not hold relative to the final position, since there are no positions greater than 0 time units in the future).

ii. The formula $\Box\ \Diamond \geq 0$ true is satisfied by any word. For such a word w, the relation $w, i \models \Diamond \geq 0$ true must hold for all positions i in w, i.e. there is some $j \geq i$ such that $w, j \models$ true. Clearly this is the case because we can always take $j = i$.

iii. The formulae are not in general equivalent. Let $w = (p, 4)\ (q, 8)$ and $a = 1, b = 2, c = 3, d = 6$. Then $w \models \Diamond[a + c, b + d]\ q = \Diamond[4, 8]\ q$, but $w \not\models \Diamond[a, b]\ \Diamond[c, d]\ q = \Diamond[1, 2]\ \Diamond[3, 6]\ q$.