We found some causes for concern in the paper by Eugene et al 2001.

Union

These all have to do with having clocks on epsilon transitions.

The following equation is presented on page 7.

$$(a \cdot 5 \cdot b) \circ (7 \cdot c) = a \cdot 5 \cdot b \cdot 2 \cdot c$$

We are using the following basis for replacement in the above TRegex.

$$x \cup x = x$$

$$7 \cdot (c \cup c) = 7 \cdot c$$

$$(a \cdot 5 \cdot b) \circ (7 \cdot (c \cup c)) = a \cdot 5 \cdot b \cdot 2 \cdot c$$

Now we can rewrite this using the rules from page 9

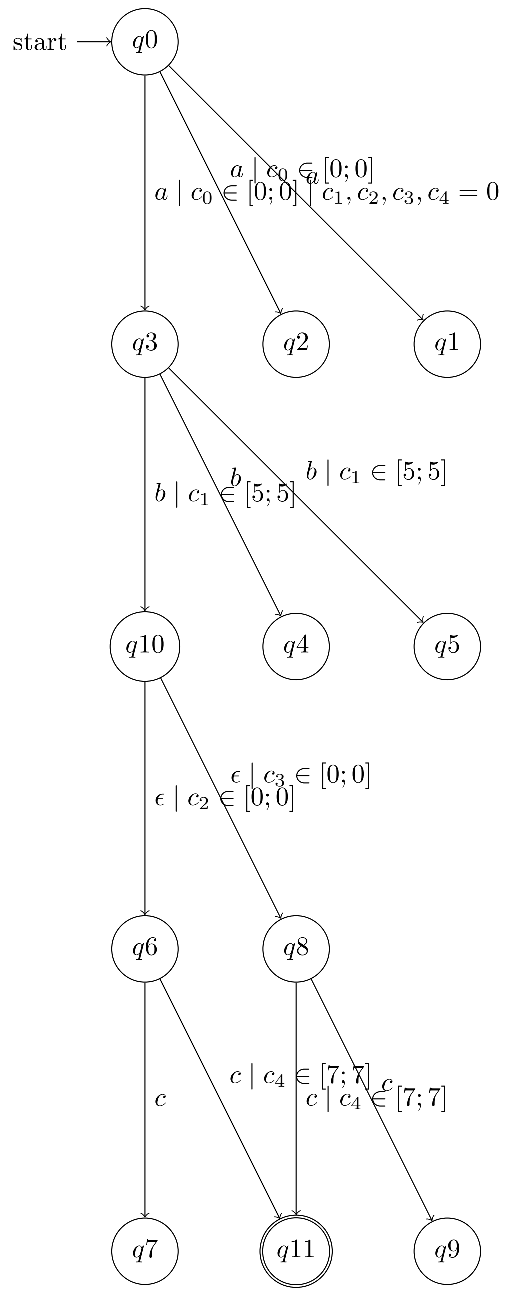
$$a \cdot 5 \cdot b \cdot 2 \cdot c \in [\![< a >_{[0;0]} \cdot < b >_{[5;5]} \circ (< c > \vee < c >)_{[7;7]}]\!]$$

Finally on page 11 we can see this can be re-written using concatenation

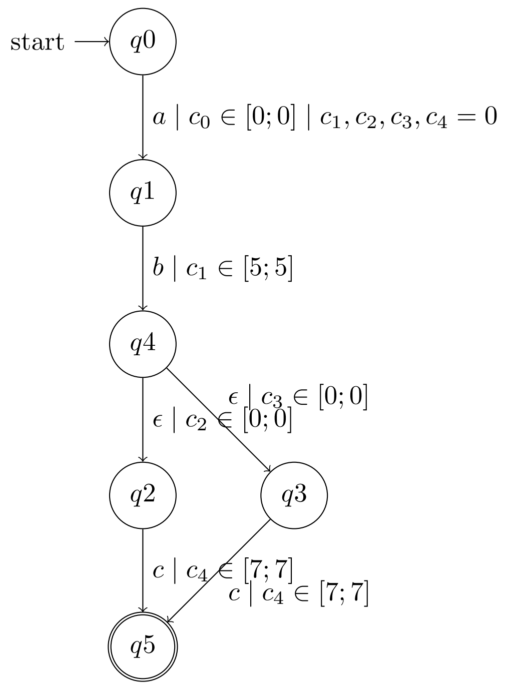
$$[\![(< a >_{[0;0]} \cdot < b >_{[5;5]} \cdot < c > \vee < c >)_{[7;7]}]\!]$$

Hence the word $a \cdot 5 \cdot b \cdot 2 \cdot c$ must be in both of these regular expressions.

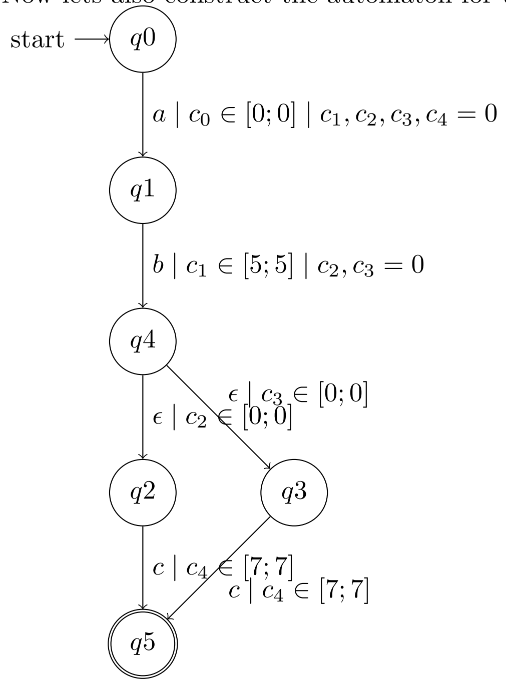
Now if we construct the Timed Automata using the rules on page 18 and 19 (definition 14) in the paper, we obtain the following Automata. First we construct for the absorbed concatenation.



Since it can be a bit hard to see what is going on, i will remove all states that have no way of reaching any final state (this will also be done from now on.)


start
$q0$
$a \mid c_0 \in [0;0] \mid c_1, c_2, c_3, c_4 = 0$
$q1$
$b \mid c_1 \in [5;5]$
$q4$
$\epsilon \mid c_3 \in [0;0]$
$\epsilon \mid c_2 \in [0;0]$
$q2$
$q3$
$c \mid c_4 \in [7;7]$
$c \mid c_4 \in [7;7]$
$q5$

Now lets also construct the automaton for the normal concatenation.


start
$q0$
$a \mid c_0 \in [0;0] \mid c_1, c_2, c_3, c_4 = 0$
$q1$
$b \mid c_1 \in [5;5] \mid c_2, c_3 = 0$
$q4$
$\epsilon \mid c_3 \in [0;0]$
$\epsilon \mid c_2 \in [0;0]$
$q2$
$q3$
$c \mid c_4 \in [7;7]$
$c \mid c_4 \in [7;7]$
$q5$

For the normal concatenation there is no problem, however for the absorbed concatenation we can quickly prove that it accepts no words at all.

The only way to take the edge $< q1, q4 >$ is if the clock c_1 is bigger equal to 5, this would also mean c_2 and c_3 are equal to 5 since they are reset at the same time as c_1.

However in order to move on from $q4$ either c_2 or c_3 would have to be 0, but there is no way to reach $q4$ with those clocks being zero, therefore the automaton can never reach an accepting state.

For normal concatenation this is however not a problem, because the clocks c_2 and c_3 are both reset before entering q_4. This means the normal concatenation automaton will accept some words.

Therefore the two automata are not equivalent, despite their timed regular expressions being equivalent.

Iterator

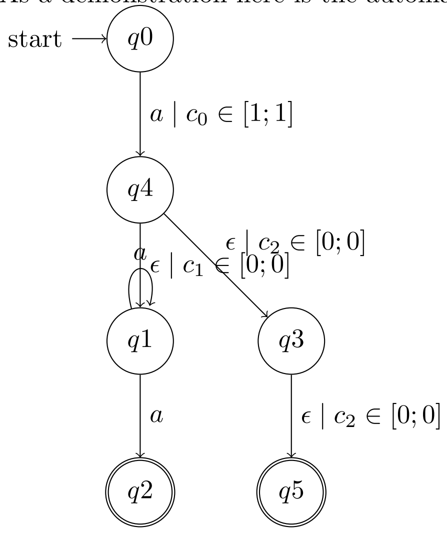
There is another issue too along the same lines for the iterator.

On page 20 for definition 14 it is described that

$$[\![\varphi_1 *]\!] = [\![\varphi_1 + \vee < \epsilon >]\!]$$

This runs into the same problem as described above, where the abosrbed concatenation will not reset the clock used for the union transitions.

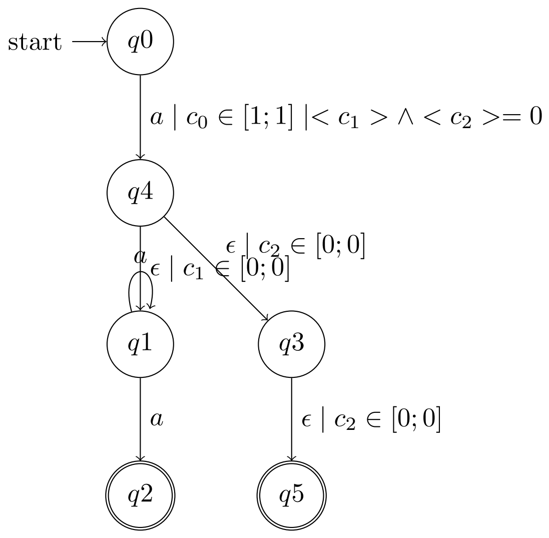
As a demonstration here is the automaton for $1 \cdot a \circ a*$


start
$q0$
$a \mid c_0 \in [1;1]$
$q4$
$\epsilon \mid c_2 \in [0;0]$
a
$\epsilon \mid c_1 \in [0;0]$
$q1$
$q3$
a
$\epsilon \mid c_2 \in [0;0]$
$q2$
$q5$

Again any transitions from $q4$ will fail, because there are no ways to reach $q4$ and still have clock c_1 or c_2 be equal to 0.

However there is another problem here not caused by the union operator. This is instead caused by the epsilon automaton, this again has a time constraint of 0. Therefore even if the union wasnt a problem this automaton could never exit at $q5$ meaning it wouldn't accept $1 \cdot a$ like the regular expression can.

Just for the record, this is how it would look with normal concatenation.


start
$q0$
$a \mid c_0 \in [1;1] \mid < c_1 > \wedge < c_2 >= 0$
$q4$
$\epsilon \mid c_2 \in [0;0]$
a
$\epsilon \mid c_1 \in [0;0]$
$q1$
$q3$
a
$\epsilon \mid c_2 \in [0;0]$
$q2$
$q5$

How to fix

The fix we suggest for this is to remove any $time = 0$ constraints, as it causes problems as soon as clocks are not being reset. That would fix all the problems we have found and documented in this document.