Exercises PVL LTL

Wishnu Prasetya

October 31, 2018

1. Below you see a Kripke structure; let’s call it $M$. Give its explicit definition in terms of a tuple etc (see the formal definition in the slides).

![Kripke structure diagram]

(a) Why don’t we have final states there?
(b) How is the notion of ‘execution’ defined for a Kripke structure? And what is an ’abstract execution’?
(c) Give an execution of $M$ that satisfies the property $X(busy \ U (x=0))$. Does $M$ satisfies the property?
(d) So, given a property Kripke structure $M$, an (abstract) execution $\Pi$, and a property $\phi$, and an natural number $i$, what is the difference between:

- $M |\models= \phi$
- $\Pi |\models= \phi$
- $\Pi, i |\models= \phi$

2. Express the following requirements in LTL. Make the necessary assumptions if you have to; but be reasonable.

(a) $P$ and $Q$ cannot not use a resource $r$ simultaneously.

**Answer:**

$\square \neg (use(P,r) \land use(Q,r))$

where $use(P,r)$ is a predicate which is true while and as long as $P$ is using $r$. Importantly note that it does not represent a program call.

(b) Whenever $P$ requests access to $r$, eventually it will get the access.

**Answer:**

$\square (req(P,r) \rightarrow \diamond use(P,r))$

where $req(P,r)$ is a predicate which is true while and as long as $P$ is requesting for $r$. 
(c) Whenever $P$ requests access to $r$, eventually it will get the access; but only if $P$ persists on maintaining the request.

**Answer:**

$$\Box (\text{req}(P, r) \rightarrow (\text{req}(P, r) \lor \neg \text{req}(P, r) \lor \text{use}(P, r)))$$

(d) $P$ cannot access $r$ without first requesting it; and it cannot do so (make a request) without first releasing $r$ (if it was busy using $r$).

**Answer:**

$$\Box (! \neg \text{req}(P, r) \land \neg \text{use}(P, r)) \lor (\text{req}(P, r) \land \neg \text{use}(P, r))$$

$$\Box (\text{use}(P, r) \land \neg \text{req}(P, r)) \lor \neg \text{use}(P, r) \land \neg \text{req}(P, r))$$

3. Construct Buchi automata representing the following LTL formulas:

(a) $p \, W \, q$, where $p, q$ are atomic propositions.

**Answer:**

$$\begin{array}{c}
\text{0} \\
p \in \\
\text{1} \\
q \in \\
\ast
\end{array}$$

Where the above is a standard Buchi with both states accepting.

(b) $\neg (x>0 \, U \, x=y)$

**Answer:** Note that $\neg (p \, U \, q) = (p \land \neg q) \, W \, (\neg p \land \neg q)$. So the above property is equivalent to:

$$(x>0 \land \neg x=y) \, W \, (\neg x>0 \land \neg x=y)$$

This results in the following standard Buchi automaton. Both states are accepting.

$$\begin{array}{c}
\text{0} \\
x>0, x=y \notin \\
\text{1} \\
\ast
\end{array}$$

(c) $p \, U \, (q \, U \, r)$, where $p, q$ are atomic propositions.

**Answer:** The following standard Buchi with $\{0, 1\}$ as initial states, and state 3 as the only accepting state.

(d) $(X \, x>0) \, U \, x=y$

**Answer:** A standard Buchi, with 2 as the accepting state.

(e) $\Diamond \Box (x>0 \rightarrow x=y)$

**Answer:** Notice first that $x>0 \rightarrow x=y$ can also be written as $\neg(x>0) \lor x=y$. Below is a standard Buchi with 1 as the accepting state.
(f) \( (p \cup q) \textbf{W} r \)

**Answer:** Using this standard Buchi, with \{0, 2\} as the initial states:

With \( F = \{1, 3, 5\} \) as the accepting states. Accepting via 1 describes executions whose prefix repeatedly satisfy \( p \cup q \), zero or more times, and ends up in \( q \) (in the case of at least one time \( p \cup q \)); and then it is followed up with \( r \).

Accepting via 3 describes the scenario of executions that remain in \( p \cup q \) forever, without ever to go over to \( r \).

Finally, accepting via 5 describes executions whose prefix repeatedly satisfy \( p \cup q \), zero or more times, and then they go over to \( r \); thus still owing one future \( q \), but this future \( q \) is met (after \( r \)).