Logic in Computer Science

Model checking

A *transition system* (or Kripke frame) is a triple (S, R, L) where S is a finite set of states, $R(s, t)$ a binary relation on S such that

$$\forall s \; \exists t \quad R(s, t)$$

and L is a labelling function, so that $L\ s$ gives a value 0 or 1 to each atom.

A *path* or *behavior* or *possible run of a program* for this transition system is an infinite sequences of state $\pi = \pi_0, \pi_1, \pi_2, \ldots$ such that $R(\pi_n, \pi_{n+1})$ for all n.

To such a path, we can associate a model α of LTL by taking $\alpha\ p\ n = L\ \pi_n\ p$ and we define $\pi \models \varphi$ to mean $\alpha \models \varphi$. (This is equivalent to the definition presented in the book.)

We define $(S, R, L) \models \psi$ to mean $\pi \models \psi$ for all path π of (S, R, L).

A *model-checker* for LTL is an algorithm deciding $(S, R, L) \models \psi$.

Example of a LTL model-checking problem

It is possible to encode the *Hamiltonian Path Problem* as a LTL model-checking problem. The Hamiltonian Path Problem is the following problem: given a graph (V, G) to decide if there is a way to enumerate V as a sequence of vertices $v_1, \ldots, v_n$ (where each vertex appears exactly once) and such that $G(v_1, v_2), \ldots, G(v_{n-1}, v_n)$. This is a well-known NP-complete problem.

For this reduction, we introduce the atoms p_v for each v in V and define the following transition system. We take S to be $V \cup \{b\}$ where b is not in V and add new edges $R(v, b)$ for all v in V and $R(b, b)$, and $R(v, v')$ if $G(v, v')$. We then have

$$\forall s \; \exists t \quad R(s, t)$$

The labelling function is defined by taking $L\ b\ p_v = 0$ and $L\ v'\ p_v = 1$ if $v = v'$ and $L\ v'\ p_v = 0$ if $v \neq v'$.

The following formula ψ is then such that $(S, R, L) \models \psi$ iff the Hamiltonian Path Problem has *not* a solution

$$\psi = \bigvee_{v \in V} (F(p_v) \to F(p_v \wedge XF(p_v)))$$

Indeed this implies that for any path π, there exists v such that either π does not visit v or π visits v twice.