TLZ (Abstract)

Leslie Lamport

Fri 13 May 1994 [18:54]

A Z schema can specify a *functional* system—one that produces an output in response to an input. For example, a simple text editor is a functional system; it can be specified by a Z schema that describes the effect of each keystroke on the screen. A *reactive* system interacts with its environment in a more complex fashion. Adding an interrupt key, which allows the user to stop a long operation before it completes, turns a simple text editor into a reactive system. Concurrent and distributed systems are usually best viewed as reactive systems. Z by itself is inadequate for specifying reactive systems.

In principle, temporal logic is ideal for specifying reactive systems. In practice, conventional temporal logic does not work very well. Real systems are complex, and complex formulas involving temporal operators are difficult to understand and easy to get wrong.

TLA, the Temporal Logic of Actions, is a form of temporal logic that does work well for specifying reactive systems. It combines nontemporal action specifications with a soupcon of temporal logic. Complexity is relegated to the action specifications, not the temporal operators.

Formally, an *action* is a Boolean-valued expression containing primed and unprimed variables. An action is interpreted as a predicate on pairs of states, where a state is an assignment of values to variables. The action $x' = x + y$ is true on the pair $\langle s, t \rangle$ of states iff the value of x in state t equals the sum of the values of x and y in state s. TLA uses two temporal operators: the familiar $\Box$ of linear-time temporal logic, which means *always*, and $\boldsymbol{\exists}$, existential quantification over flexible variables, which is a hiding operator. A temporal formula is interpreted as a predicate on behaviors, which are infinite sequences of states. The canonical form of a TLA specification is

$$\boldsymbol{\exists}\, y : I \wedge \Box[\mathcal{N}]_{\langle x, y \rangle} \wedge L$$

where x and y are tuples of variables, I is a state predicate (an action with no primed variables), $\mathcal{N}$ is an action, and L is the conjunction of fairness conditions on actions. This formula essentially asserts of a behavior that there exist values for the variables y (possibly different values for each state in the behavior) such that I holds in the initial state, every successive pair of states satisfies $\mathcal{N}$ or leaves x and y unchanged, and L is satisfied. There are two fairness conditions that can be asserted about an action $\mathcal{A}$: weak fairness, which asserts that an $\mathcal{A}$ step must occur if $\mathcal{A}$ remains continuously enabled, and strong fairness, which asserts that an $\mathcal{A}$ step must occur if $\mathcal{A}$ is repeatedly enabled.

A Z schema can be viewed as the specification of an action. One can obtain a language TLZ for specifying reactive systems by using Z as the logic of actions in TLA. Most of a TLA specification consists of action specifications, so most of a TLZ specification will consist of ordinary Z.

TLA specifications are logical formulas, and they are combined with logical operators. Implementation is implication; a specification S_2 implements a specification S_1 iff the TLA formula $S_2 \Rightarrow S_1$ is valid. Parallel composition is conjunction; $S_1 \wedge S_2$ is the composition of specifications S_1 and S_2. TLZ can be viewed as Z with the schema calculus extended with the operators $\Box$ and $\boldsymbol{\exists}$. We can also take the opportunity to simplify Z before adding the "TL".