

0.1 On the notion of behavior

The assertion logic presented in the previous section allows us to reason about the behavior of a system without explicitly generating the possible sequences of states resulting from the execution of the program. However, underlying the inference rules of our assertion logic we need a mathematical model for the operational behavior of a system.

An operational model is needed to prove the soundness of the inference rules. Further, an operational model may aid in understanding the meaning of particular language constructs and their associated correctness rules. In the following we will sketch the construction of a *transition system* modeling the behavior of an object-based program. Studying the formal semantics is relevant to understanding object orientation only in so far as it provides a means with which to characterize the desired behavior of object creation and message passing in an unambiguous manner.

Transition system A transition system for a program is a collection of rules that collectively describe the effect of executing the statements of the program. A labeled transition system is one that enables us to label the transition from one state to another by some label indicating the observable behavior of a program step.

In the transition system defined below, we will employ states ϕ , which may be decorated by object identifiers α , as in ϕ^α . Object identifiers are created when creating a new instance of an object type τ . We assume newly created object identifiers to be unique.

We assume that each object type τ has a constructor (which is a, possibly empty, statement that we write as S_τ) and an arbitrary number of methods m . Each method m is assumed to be defined by some statement, which we write as $S_m(e)$, for method calls of the form $m(e)$. Also we allow an object α of type τ to have attributes or instance variables v that may be accessed (read-only) as $\alpha.v$ for an object identifier α or $x.v$ for an object variable x (which must have α as its value).

To determine the visible behavior of a program, we will employ labels of the form α (to denote the creation of an object α) and m_α (to indicate the invocation of a method m for object α). We allow transitions to be labeled by sequences of labels that we write as λ and which are concatenated in the usual way.

We will define a transition system for a simple language of which the syntax is defined in slide 0-1.

Expressions are either local variables v or object instance variables that we write as $x.v$, where x is an object variable. As elementary statements we have $v = e$ (indicating the assignment of (the value of) an expression e to a local variable v), $x = \text{new } \tau$ (which stands for the creation of a new object of type τ), and $x.m(e)$ (which calls a method m with arguments e for object x). The object variable x is associated with an object identifier α by the state ϕ in which the statement in which x occurs is executed. As compound statements we have an empty statement ε (which is needed for technical reasons), an elementary