Modularity and Composition in Propositional Statecharts

H.R. Dunn-Davies\(^1\), R.J. Cunningham\(^1\), and S. Paurobally\(^2\)

\(^1\) Department of Computing, Imperial College, London, UK, SW7 2BZ
\(^2\) Department of Computer Science, University of Liverpool, Liverpool, UK, L69 7ZF

Abstract. Propositional Statecharts, described in \[3\], are a variation of David Harel’s Statechart formalism \[6\] intended to enable both diagrammatic description of an agent interaction protocol, and interpretation as a theory in a dynamic logic. Here we provide an informal description of a diagrammatic extension to enable modular representation.

1 Introduction

Several diagrammatic methods have been proposed for the description of agent interaction protocols (\[4\], \[2\], \[6\], \[1\]). Although their visual basis allows a protocol to be described in a relatively easy-to-follow manner, it has been shown that these semi-formal methods of expressing protocols can lead to errors and ambiguities \[7\], with different interpretations among agent designers, so that different agents participating in the same interaction may in fact be using slightly different protocols. The Propositional Statechart formalism is intended to enable the visual intuition of a diagram to be underpinned by formal inferences from its interpretation as a theory in a dynamic logic such as ANML \[7\]. This is an on-going project, but it seems that Propositional Statecharts are capable of expressing many interaction protocols completely and unambiguously.

One noticeable drawback of the version of the Propositional Statechart formalism described in \[3\] is the lack of a mechanism for modularity, whereby Statecharts can be defined in terms of separate modular components. The Propositional Statechart formalism is intended to represent interaction protocols in their entirety, so can benefit greatly from this capability. In contrast, Harel’s statecharts have chiefly been used for component description, for instance in an object oriented systems. Nevertheless, the modularity mechanism we illustrate appears to be of wider applicability.

The need for a modular approach to protocol definition has been highlighted by Vitteau and Huget \[9\], who point out that many real world interactions are composed of sub-interactions, which may be repeated during the main interaction. The ability to define a protocol in a modular fashion enables the designer to split large protocols into smaller, more manageable components, and provide a single definition of any part of the protocol that occurs more than once. This helps make definitions concise and makes it possible to create libraries of components, so that protocols do not always have to be built from scratch.
A further advantage of modularity is that it becomes possible to represent recursively defined protocols which could not be described otherwise. This is an important step since many protocols, such as some versions of the multilateral protocol discussed in section 3, cannot be described adequately in any other way. Here we describe an extension to the Propositional Statechart formalism that allows the modular representation of interaction protocols.

2 Adding Modularity to Propositional Statecharts

The mechanism we propose to enable modular definitions takes advantage of Statecharts’ use of topological enclosure of contours to represent state hierarchy. The principle behind it is as follows: since the superstate of any collection of atomic states represents the disjoint union of those states, any given group of contours describing states can be replaced by a contour representing their superstate, without loss of precision, as long as a complete description of all of the states contained within the superstate is provided in a separate Statechart.

This approach allows us to define recursive protocols, as superstates can contain skeleton representations of themselves. It is not in general possible to finitely enumerate all of the states of a recursive protocol, as the number of states may not be bounded. However, a modular representation may provide as complete a description as is required, provided more complete representations can be generated by replacing each skeleton representation with its definition.

The mechanism is shown in figures 1 and 2. Figure 1 is a simple Statechart, and figure 2 provides a modular definition with the same intended behaviour. Figure 2 can be generated from figure 1 by the following procedure. Firstly, add a superstate, named \( H \), enclosing states \( D \), \( E \), \( F \), and \( G \), a process which is trivially behaviour preserving. Secondly, copy state \( H \) and its contents (all of the labelled contours and arrows completely enclosed by the contour representing state \( H \)), to form a separate modular component. Finally, replace the representation of state \( H \) in statechart \( A \) with a skeleton representation which includes only the states linked by a single transition to a state outside \( H \) (we refer to these as interface states). The recombination of the two components can easily be performed, essentially by reversing the above steps.

In order to ensure that a Propositional Statechart designer can refer to a component more than once in the same superstate while preserving uniqueness of the state labels we treat a component Statechart as a class of superstates.

1 Thence it appears more expressive than the graphical representations cited above, which seem not to provide adequate descriptions of recursive protocols.
2 In some cases it may be necessary to specify explicitly which of a number of components represents the top level of the combined statechart. In this example it is not, as there is only one possible combination of the two components.
3 We have also shaded each interface state in \( A \) to mark \( H \) as a skeleton representation, but have neglected to shade any states in the modular component representing \( H \). This is because we aim to create modular components which can be used in different contexts, and to provide definitions which are as general as possible.