Abstract

In this paper we present an algebraic approach to statecharts as they are used in the STATEMATE tool in the style of "Petri-Nets are Monoids" for place-transition nets developed by Meseguer and Montanari. We apply the framework of high-level-replacement systems, a categorical generalization of graph transformation systems, in order to define union as horizontal as well as transformation and refinement as vertical structuring techniques for statecharts. The first main result shows compatibility of union and transformation in a suitable category of statecharts. We present an algorithm for the computation of all transitions enabled within one step. The second main result shows the correctness of this algorithm. We define refinement morphisms for statecharts, which allow refinement of arbitrary states, in contrast to concepts in the literature where only basic and root states are subject of refinement. The third main result shows that refinement morphisms are compatible with the behavior of statecharts as defined in the formal semantics.

1 Introduction

The great success of statecharts [5,10] as a visual specification technique for concurrent and reactive systems is mainly due to its compact representation of concurrency and hierarchy. The existing STATEMATE tool [7] is offering sophisticated simulation and prototyping facilities and has thus led to a broad acceptance of this formalism. Similar to other specification techniques for concurrent systems horizontal and vertical structuring techniques are most important for the development of large and complex systems, but the theory of statecharts offers only little support for this problem. In this paper we present horizontal and vertical structuring techniques and results for statecharts from two points of view: On one hand we study union and transformation in analogy to approaches for graph transformation systems and Petri nets recently developed in the literature [2,13,4]. On the other hand we present a formal semantics of (a subset of) STATEMATE statecharts [8] and a new notion of refinement, which is compatible with this formal semantics and allows refinement of arbitrary states in contrast to some other concepts of refinement for statecharts.

As pointed out horizontal and vertical structuring techniques have been studied for various kinds of specification techniques in the literature. They are
new for statecharts. Other notions for formal semantics of statecharts are given in [9,11,12,14], but they are not conform with the STATEMATE semantics [8]. An overview of several statechart variants is given in [17]. For different kinds of object oriented statecharts [1,6,15,16] refinement techniques have been studied allowing root and basic state refinement, but not refinement of general states as in our case. Moreover no formal semantics has been presented for these approaches.

In section 2 we present the basic notions of abstract statecharts and statecharts. Union and transformation of abstract statecharts and the first main result are presented in section 3. The behavior of statecharts in the sense of STATEMATE is given in section 4. The correctness of the presented algorithm is the second main result. The definition of a behavior compatible refinement of statecharts in section 5 is the third main result. Because of space restrictions we are only able to present short proof ideas. A more detailed version is available from the authors.

2 Statecharts

Statecharts are automata equipped with hierarchy and concurrency. Hierarchy is achieved by embedding one statechart in a state of another one. Concurrency is expressed by the parallel composition of two statecharts. These structuring mechanisms are described by the notion of hierarchical state space.

**Definition 2.1 (Hierarchical State Space)** Given a set of states $S$, a distinguished state $\text{root} \in S$, a function $\text{substates} : S \rightarrow \mathbb{P}S$, a function $\text{decomp} : S \rightarrow \{\text{and, xor, basic}\}$, applying to each state its decomposition type, such that $\forall s \in S$, $s' \in \text{substates}(s)$ $\bullet$ $\text{decomp}(s) = \text{and}$ $\Rightarrow$ $\text{decomp}(s') = \text{xor}$. We call $\mathcal{HS} = (S, \text{root}, \text{substates}, \text{decomp})$ a hierarchical state space.

A configuration contains all states a statechart resides in at a moment.

**Definition 2.2 (Configuration)** Given a hierarchical state space $\mathcal{HS}$. We call a set of states $C \subseteq S$ a partial configuration w.r.t. a state $s$ if the following holds:

- $s \in C$ $\land$ $\forall s' \in C$ $\bullet$ $s' \triangleleft s$
- $\forall s' \in S$ $\bullet$ $s' \notin C$ $\Rightarrow$ $C \cap \text{substates}^+(s') = \emptyset$
- $\forall s' \in C$ $\bullet$ $\text{decomp}(s') = \text{and}$ $\Rightarrow$ $\text{substates}(s') \subseteq C$
- $\forall s' \in C$ $\bullet$ $\text{decomp}(s') = \text{xor}$ $\Rightarrow$ $\exists s'' \in \text{substates}(s')$ $\bullet$ $s'' \in C$ $\lor$
  $\text{substates}^+(s') \cap C = \emptyset$

A partial configuration $C$ w.r.t. $s$ is called total, if $\forall s' \in C$ $\bullet$ $\text{decomp}(s') = \text{xor}$ $\Rightarrow$ $\exists s'' \in \text{substates}(s')$ $\bullet$ $s'' \in C$. For a state $s$ the set of all total resp. partial configurations w.r.t. $s$ is denoted by $C(s)$ resp. $C_p(s)$. We write $C$ for $C(\text{root})$ resp. $C_p$ for $C_p(\text{root})$. 

$^b\mathbb{F}$, resp. $\mathbb{F}$ stand for (nonempty) finite powerset.

$^c$\text{substates} and $\text{substates}$ denote the irreflexive, resp. the reflexive and transitive closure of $\text{substates}$.