Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 280 (2011) 35–46

www.elsevier.com/locate/entcs

# Generating Hierarchical State Based Representation From Event-B Models

Dipak L. Chaudhari[1,2] Om P. Damani[3]

*Department of Computer Science and Engineering,*
*Indian Institute of Technology, Bombay,*
*Mumbai, India*

**Abstract**

Many properties of a system may not be obvious just by a quick inspection of the corresponding Event-B model. Users typically rely on animation, scenario analysis, and inspection of state transition graphs for discovering certain behavior of the system. We propose a methodology for generating a hierarchical representation of the system for visualising Event-B models. Our representation is succinct and it provides multiple views to aid in better comprehension of the Event-B models.

*Keywords:* Event-B, Model visualization, Hierarchical state based representation

## 1 Introduction

In Event-B, desired global properties of the system are specified in the form of *invariants* and the *invariant preservation proofs* ensure that these properties are maintained by the system after execution of any enabled event[1]. However, after execution of an enabled event, it is not obvious which events will be enabled or disabled next. Users typically rely on animation, scenario analysis, and inspection of state transition graphs to grasp the behavioral aspects of the system. The ProB animator[9], with the aid of a model checker, can generate graphical visualization of the state space of a B machine. However, because of the flat (non-hierarchical) nature of the ProB state space representation, it becomes difficult to reduce the complexity of the state space graphs even after employing the state space reduction techniques[10]. In general, hierarchical state transition diagrams are found to be useful in reducing the complexity of the state transition diagrams [6].

[1] This work was supported in part by the Ministry of Human Resources Development, Government of India and by the Tata Research Development and Design Center (TRDDC).

[2] Email: dipakc@cse.iitb.ac.in

[3] Email: damani@cse.iitb.ac.in


doi:10.1016/j.entcs.2011.11.016

We propose a hierarchical representation, similar to the statechart diagrams, for visualising Event-B models. We present a top-down methodology for constructing an abstract representation of desired granularity directly from the given Event-B model.

## 2 Hierarchical Abstract State Transition Machine

To represent a discrete event system, we use a Hierarchical Abstract State Transition Machine (HASTM) representation which uses the concepts of hierarchical states and guarded transitions similar to those in statechart diagrams [6]. In HASTM, state-space is arranged in the form of a tree (which we call a state-space partition tree) and the root node of the tree represents all the valid states of the system, i.e., the states defined by the conjunction of all the invariants. The root node is partitioned into substates based on some predicate. The substates are in turn partitioned further using appropriate predicates. Figure 1 shows the state-space partition tree generated by our method (Algorithm 1) for the Lift Event-B model [4] shown in Figure 2, given the predicates $(cf = topFloor)$, $(cf = botFloor)$, $(doorOpen = T)$, and $(dirUp = T)$. The algorithm starts constructing the tree from the root node and at each node selects a partitioning predicate that minimizes the number of transitions in the generated HASTM. This reduces the complexity of the generated HASTM. While partitioning the tree, the algorithm also computes the pre-states, transition guards, and the post states (defined in Section 2.1 ) for the transitions. The final HASTM for the Lift model is shown in Figure 4.

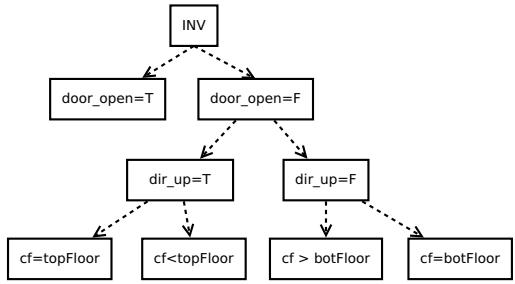


Fig. 1: State-space partition tree for the Lift example. The hierarchy relation is shown by dotted arrow.

### 2.1 Structure and Semantics of HASTM

If $v$ denotes the variables of a system then the set $\Phi = \{v|\text{True}\}$ is the entire state space of the system. We use the term *abstract state* to represent any subset of $\Phi$ and the term *concrete state* or just *state* to represent a particular element of $\Phi$. [5]

Abstract states are usually specified using predicates. If $Q(v)$ is a predicate with free variables in $v$ then we represent by $Q$ the set of all concrete states satisfying $Q(v)$, i.e., $Q = \{v|Q(v)\}$. If a system is in a concrete state $q$, and $q \in Q$ where $Q$ is an abstract state then the system is said to be in the abstract state $Q$.

HASTM is a tuple $\mathcal{H} = \langle v, \mathcal{S}, \succ, \Sigma, T, t_0 \rangle$ , where

[4] The Lift Event-B model is adapted from the B model that comes with the ProB tool [9].

[5] The terms *abstract state* and *concrete state* should not be confused with the terms *abstract model* and *concrete model* which are used in context of refinements.