Multivalued Possibilities Mappings *

Nancy A. Lynch
Laboratory for Computer Science
MIT
Cambridge, MA 02139

Abstract: Abstraction mappings are one of the major tools used to construct correctness proofs for concurrent algorithms. Several examples are given of situations in which it is useful to allow the abstraction mappings to be multivalued. The examples involve algorithm optimization, algorithm distribution, and proofs of time bounds.

Keywords: abstraction mapping, mapping, possibilities mapping, safety property, Alternating Bit Protocol, transaction processing, garbage collection, distributed algorithms, time bounds, history variables

Contents
1 Introduction
2 A Formal Framework
3 Algorithm Optimization
   3.1 Alternating Bit Protocol
      3.1.1 Problem Statement
      3.1.2 Architecture
      3.1.3 Alternating Bit Protocol
      3.1.4 Redundant Protocol
      3.1.5 Possibilities Mapping Proof
      3.1.6 Remarks
   3.2 Transaction Processing
   3.3 Garbage Collection
   3.4 Remarks
3 Distribution
5 Proving Time Bounds
   5.1 Overview
   5.2 Formal Model
      5.2.1 Timed Automata
      5.2.2 The Automaton time(A)
   5.2.3 Strong Possibilities Mappings

*This work was supported by ONR contract N0014-85-K-0168, by NSF contract CCR-8611442, and by DARPA contract N00014-83-K-0125.
Introduction

Abstraction mappings are one of the major tools that my colleagues and I use to construct correctness proofs for concurrent (including distributed) algorithms. In this paper, I will try to make one major point about such mappings: that it is useful to allow them to be multivalued. That is, often when one maps a "low-level" algorithm L to a "high-level" algorithm H, one would like to allow several states of H to correspond to a single state of L. I believe that any useful framework for describing abstraction mappings should include the ability to describe multivalued mappings.

I don't know if this point is especially controversial. I have been using multivalued mappings since I started carrying out such proofs in 1981, and the popular notion of bisimulation proposed by Milner [20] also permits multiple values (although bisimulation is a stronger notion than I advocate here, since it requires simulation relationships between L and H in both directions). However, work on history variables, tracing its roots to [22], takes pains to avoid the use of multivalued mappings by adding extra information to the state of L, and there are also some recent papers (e.g., [13, 12, 1]) that restrict the notion of mapping to be single-valued.

I will describe some situations in which multivalued abstraction mappings are useful. The examples I consider involve

1. algorithm optimization,
2. distribution, and
3. proving time bounds.

I will illustrate the first of these situations in some detail, using one familiar example (the Alternating Bit Protocol) and two less familiar examples, just touch on the second, and spend the remaining time on the third - it's the newest use I have found and possibly the most interesting.

In my work, abstraction mapping seem most useful for proving safety properties; although I have been involved in some work that proves liveness properties using such mappings (e.g., [17, 27]), these efforts are still somewhat ad hoc. Note that timing properties are more like safety properties than like liveness properties; because of this, mappings are useful for proving timing properties as well. In this paper, I will restrict attention to safety and timing properties.

A Formal Framework

To be concrete, I will describe the work in terms of I/O automata [17, 18], since that is what I've actually used. The precise choice of model is not very important for most of what I will discuss here (timing proofs excepted); other state machine models would probably do as well. Here, I will review the definition of an I/O automaton and will give the usual notion of mapping, called a possibilities mapping, that I use for defining a correspondence between I/O automata.

Recall that an I/O automaton consists of states, start states, actions classified by a signature as output, input and internal, and steps, which are (state, action, state) triples. So far, that makes them