Algebra, Logic and Applications Series Volume 11

Models for Concurrency

Uri Abraham

Gordon and Breach Science Publishers

MODELS FOR CONCURRENCY

Uri Abraham
Department of Mathematics and Computer Science
Ben Gurion University, Be'er Sheva, Israel

GORDON AND BREACH SCIENCE PUBLISHERS

Australia • Canada • China • France • Germany • India
Japan • Luxembourg • Malaysia • The Netherlands • Russia
Singapore • Switzerland

Amsteldijk 166
1st Floor
1079 LH Amsterdam
The Netherlands

British Library Cataloguing in Publication Data
A catalogue record for this book is available
from the British Library.
ISBN 90–5699–199–X

MODELS FOR CONCURRENCY

Preface

The title 'Models for Concurrency' indicates that models (in the sense of model theory) are applied here to the analysis of concurrent protocols, and not that the diverse paradigms for modeling concurrency are explained.

The dominating tenet in concurrency research today takes the notion of a state as the basic unit, and uses it to describe systems and processes' behavior. A state is an assignment of values to a set of variables, and analysis of a system involves defining this set of variables and determining the rules-of-change that relate states to their successors. A 'history' is a sequence of states, possibly infinite, governed by these rules. Correctness proofs are essentially proofs 'by induction', whereby an assertion is proved to hold in each state in a history if it holds in the initial state and then the rules of change will not falsify it. I find that decomposing reality as a sequence of states is often inadequate and unnatural because we usually think of events – not of states – in forming our view of reality. These events overlap in time in such a complex manner that the state rendering of reality often becomes artificial.

When thinking and arguing informally about concurrent systems, computer scientists often use what has been called behavioral reasoning (''before sending the message, process *P* read register *R* and realized that ...''). This type of reasoning was found by many to be unreliable, and formal methods were suggested for proving protocols' correctness. For example, L. Lamport and F. B. Schneider wrote ([21] page 203):

> Most computer scientists find it natural to reason about a concurrent program in terms of its behavior – the sequence of events generated by its execution. Experience has taught us that such reasoning is not reliable; we have seen too many convincing proofs of incorrect algorithms. This has led to assertional proof methods, in which one reasons about the program's state instead of its behavior. Unlike behavioral reasoning, assertional proofs can be formalized – i.e., reduced to a series of precise steps that can, in principle, be machine-verified.

I agree that it is natural to reason about a concurrent program in terms of its behavior, but I believe that behavioral reasoning can be formalized too. Using first-order sentences and models that describe events and their attributes, one can transform behavioral reasoning into a formal mathematical proof and still keep the main features of the informal argument. The purpose of the book is to describe this approach and its background with many examples.

The basic mathematical notion used here is that of a model for a first-order theory. The first chapter introduces the required concepts from model theory (only the very elementary concepts are needed). The second chapter concentrates on system executions: the structures employed here to model concurrency. The third chapter, the heart of the book, describes the semantics of a simple language that allows concurrent executions of sequential programs. Chapter 4 explains how this semantics can be used to prove the correctness of protocols, and Chapter 5 deals with the question of resolving executions into higher-level and lower-level granularities.

The second part of the book explains the producer/consumer problem and describes two (known) solutions. The first, in Chapter 6, uses semaphores, and the second, in Chapter 7, uses a circular buffer.

The third part of the book deals with message passing rather than shared memory. The first chapter there specifies some notions connected with channels (such as non-lossy channels) which are used in the following chapters, and introduces an axiomatic treatment of sliding window protocols. Chapter 9 picks up the producer/consumer problem again, but now with messages through a channel. The purpose of this chapter is to prove the correctness of the Sliding Window Protocol. Chapter 10 discusses basic concepts (such as causality) in networks. Chapter 11 is about uniform deliveries; it is centered around a very interesting protocol: the Early Delivery Protocol of Dolev, Kramer, and Malki [14].

Not everybody is interested in completely formal correctness proofs that are written out according to strict first-order rules. The designers of a system or the programmer seeking to improve her algorithm mostly want to be convinced that the protocol is correct and to find precise descriptions of its behavior. The second and third parts of the book are written in everyday mathematical parlance that can 'in principle' be fully formalized. This approach can be valuable to those who wish to express concurrency arguments in a rigorous language (that of model theory) but do not desire to transform them into a fully formal first-order proof. (It will take more work to provide tools that help to make such transformations – an important issue though beyond the scope of this book and its methods.)

These notes originated with a course given in the summer of 1995 at Ben-Gurion University to a group of students from the industry, whose definite expectations and interests largely determined the form and content of my lectures. More practical examples were required to keep the students' interest, and no obvious background could be assumed. I am grateful to all of them. including the one who kept asking why so much effort is invested in proving obvious facts.

The book was typeset with the AMS LATEX program. I wish to express my admiration to the generosity of those who developed such programs and made them available to all. I was lucky to have two experts among my friends, James Cummings and Martin Goldstern, who knew how to adapt the program to my needs. I am also grateful to the staff of Gordon and Breach for editorial help and cooperation. Additional thanks go to Bob Constable who has made valuable suggestions concerning Chapter 2.

Contents

PART 3
Message communication

PART 1

Semantics of distributed protocols

The first part introduces some elementary notions of model theory and uses them to explain the semantics of concurrently communicating processes. We formally define what an execution of a protocol is and what it means to prove that a protocol satisfies some correctness statements.

1
Elements of model theory

First-order languages and their interpretations (models) are used in this book to specify systems and describe the behavior of programs. This chapter introduces some of the elementary notions from model theory that are needed, such as structure, first-order language, and the satisfaction relation. We study these notions informally and only to such a degree as required to read the book.

1. Structures

In this chapter we shall be concerned with language and models (structures)—two interrelated concepts that play a central role in logic and in this book. To define a language its signature is needed first.

DEFINITION 1.1 (SIGNATURE). *We say that L is a* signature *if L is a four-tuple $\langle \overline{P}, \overline{F}, \overline{c}, arity \rangle$ where:*

(1) *$\overline{P}$ is a finite sequence $\langle P_1, \dots, P_k \rangle$ of symbols called predicates. For each predicate P_i in the sequence, $arity(P_i)$ is a non-zero natural number called "the arity of P_i".*
(2) *$\overline{F} = \langle F_1, \dots, F_\ell \rangle$ is a finite sequence of "function symbols", and, again, a non-zero natural number $arity(F_i)$ is associated with each function symbol.*
(3) *$\overline{c} = \langle c_1, \dots \rangle$ is (a possibly infinite) sequence of "constants".*

For example, to describe the natural numbers, a signature L may be formed by taking a single predicate $<^*$ with arity 2, two function symbols, $+^*$ and $\times^*$, with arity 2, and a single constant, the symbol 0^* (it would also be natural to add all constants n^* to represent the natural numbers $n \in \mathbb{N}$). L is thus just a set of symbols with associated arities. The asterisk, for example $+^*$, is to emphasize that this is just a symbol rather than the familiar addition operation $+$. In later chapters I will not be that careful and the reader will have to find the status of the symbol from the context. The standard interpretation for this signature is obtained by taking as the universe of discourse the set of natural numbers $\mathbb{N}$, the familiar ordering relation as an interpretation of $<^*$, the addition and multiplication operations as interpretations of $+^*$ and $\times^*$, and the number zero to interpret 0^*.

Another natural interpretation for L is obtained by taking the real numbers. But arbitrary interpretations are also possible: Indeed any non-empty set with a binary relation, two functions, and a constant can interpret L. Of course the choice of a symbol may indicate the intentions of the users, but these intentions cannot replace a definition.

We are now going to define interpretations in general.

DEFINITION 1.2 (INTERPRETATION). *Let $L = \langle \overline{P}, \overline{F}, \overline{c}, arity\rangle$ be a signature. $\mathcal{M}$ is called an interpretation (or a structure) for L if $\mathcal{M} = \langle A, \overline{P}^{\mathcal{M}}, \overline{F}^{\mathcal{M}}, \overline{c}^{\mathcal{M}}\rangle$ consists of the following.*

(1) *A is a non-empty set denoted $|\mathcal{M}|$ and called the* universe *of $\mathcal{M}$. Members of A are called "individuals" of $\mathcal{M}$.*
(2) *$\overline{P}^{\mathcal{M}} = \langle P_1^{\mathcal{M}}, \dots, P_k^{\mathcal{M}}\rangle$ associates with each predicate P_i in L of arity $m = arity(P_i)$ an m-ary relation $P_i^{\mathcal{M}}$ on $|\mathcal{M}|$. That is $P_i^{\mathcal{M}} \subseteq A^m$. (For any set A, A^m denotes the collection of all m-tuples from A.) In particular, every unary predicate symbol is associated with a subset of the universe.*
(3) *$\overline{F} = \langle F_1^{\mathcal{M}}, \dots, F_\ell^{\mathcal{M}}\rangle$ is an interpretation of all function symbols. For each function symbol F_j, of arity m, $F_j^{\mathcal{M}}$ is an m-place function*

$$F_j^{\mathcal{M}} : |\mathcal{M}|^m \to |\mathcal{M}|.$$

(The notation $f : X \to Y$ means that f is a function from X to Y. Thus $F_j^{\mathcal{M}}$ is defined on $|\mathcal{M}|^m$, the set of m-tuples of individuals, and takes values in $|\mathcal{M}|$.)
(4) *Finally, $\overline{c}^{\mathcal{M}} = \langle c^{\mathcal{M}} \mid c \text{ a constant}\rangle$ interprets the constants of L: For every constant c in L, $c^{\mathcal{M}}$ is an individual of $\mathcal{M}$. That is, $c^{\mathcal{M}} \in |\mathcal{M}|$.*

Reality often forces us to consider objects of different natures. There are two possible ways to model two (or more) sorts of individuals in a structure. The first is to assume that there is a single universe containing a mixture of individuals of all sorts, and to distinguish them by different predicates. The second is the way adopted here: to assume not just a single universe of discourse, but several domains, called sorts, in a single structure. Such structures are said to be multi-sorted. I give an example for this in the following subsection, but multi-sorted signatures are defined first.

DEFINITION 1.3 (MULTI-SORTED SIGNATURE). *A multi-sorted signature is a sequence of the form*

$$L = \langle S_1, \dots, S_n;\ \overline{P}, \overline{F}, \overline{c}, arity, sort\rangle$$

where $\langle \overline{P}, \overline{F}, \overline{c}, arity\langle$ is a signature, $S_1, \dots, S_n$ is a list of symbols called sorts, and sort is a function that associate with each predicate, function, or constant its sort as follows.

If P is a predicate of arity k, then $sort(P)$ is a k-tuple of sorts. That is $sort(P) = \langle X_1, \dots, X_k\rangle$ where each X_i is some S_j (repetitions are allowed).

The intention is that the question of whether $P(x_1, \dots, x_k)$ holds or not can be asked only if each x_i is of sort X_i. Similarly, with each k-ary function symbol F, sort(F) is a $k+1$ tuple of sorts defining the sorts both of the domain and the range of F. Finally sort(c) for a constant c gives the sort of c. We often introduce a semicolon (;) after the list of sorts to separate them from the predicates and functions.

DEFINITION 1.4 (MULTI-SORTED INTERPRETATION). *An interpretation $\mathcal{M}$ for a multi-sorted signature L consists of*

(1) *a universe $|\mathcal{M}| = S_1^{\mathcal{M}} \cup S_2^{\mathcal{M}} \cdots \cup S_n^{\mathcal{M}}$, which is now a union of the sorts (not necessarily a disjoint union), and*
(2) *the predicates, function symbols and constants, which are interpreted in accordance with their sorts.*

This means, for example, that if F is a two-place relation symbol and *sort*(F) $= \langle S_1, S_2 \rangle$, then $F^{\mathcal{M}} \subseteq S_1^{\mathcal{M}} \times S_2^{\mathcal{M}}$. Similarly, if g is a unary function with *sort*(g) $= \langle S_1, S_2 \rangle$, then $g^{\mathcal{M}} : S_1^{\mathcal{M}} \to S_2^{\mathcal{M}}$.

DEFINITION 1.5 (ISOMORPHISM). *Let L be a signature with sorts $S_1, \dots, S_n$, and let $\mathcal{M}_1$, $\mathcal{M}_2$ be two interpretations of L. We say that $f : |\mathcal{M}_1| \to |\mathcal{M}_2|$ is an* isomorphism *of $\mathcal{M}_1$ and $\mathcal{M}_2$ if f is one-to-one from the universe of $\mathcal{M}_1$ and onto the universe of $\mathcal{M}_2$ such that:*

(1) *For every sort S_i, $m \in S_i^{\mathcal{M}_1}$ iff $f(m) \in S_i^{\mathcal{M}_2}$. (*iff *stands for "if and only if".)*
(2) *For every n-ary predicate symbol P and n-tuple $m_1, \dots, m_n \in |\mathcal{M}_1|$,*

$$\langle m_1, \dots, m_n \rangle \in P^{\mathcal{M}_1} \textit{ iff } \langle f(m_1), \dots, f(m_n) \rangle \in P^{\mathcal{M}_1}.$$

(3) *For every n-place function symbol F,*

$$f(F^{\mathcal{M}_1}(m_1, \dots, m_n)) = F^{\mathcal{M}_2}(f(m_1), \dots f(m_n)).$$

Similarly, if c is any constant then $f(c^{\mathcal{M}_1}) = c^{\mathcal{M}_2}$.

Since we shall identify two isomorphic structures, this concept reflects our understanding that the inner composition of the elements of the universe of a structure is not relevant. These elements are just abstract "points" devoid of any inherent meanings, were it not for the functions and predicates defined on them.

1.1. Examples of multi-sorted structures. In the first example we model the situation in which several measurements of light intensity are made by some light-meter. A measurement evaluates the average intensity of light during the opening of the meter, which is by definition the integral of the intensity over the aperture interval divided by its length. We want to be able to express in our language the fact that distinct measurements may be made over different exposure intervals and may show different values. For this, every structure contains a set of "events" that represent measurements, and a value is assigned to each event to represent the result of the measurement. Hence two sorts of individuals are needed: measurement events and real numbers (if we want the results to be real numbers). Moreover, to evaluate the measurement error, we would like to have the "true" light intensity function in the structure, that is the function that gives the real intensity at each instant. We shall denote this function by I and let $I(t)$ denotes the light intensity at moment t. We also want the integral function of I. This integral function will allow us to express the average intensity.

In our example the measurement events are one sort and the real numbers are another. An advantage of this two-sorted approach is that functions and relations may be specific to certain sorts. The addition operation, for example, is defined on the real numbers, and it is meaningless to ask for the addition of events.

So we define first a signature K that contains two sorts: E and R (the elements of E are called events and those of R numbers). In addition, K contains the following.

(1) A binary predicate $<$ and binary function symbols $+$, $-$, $\times$, $/$ over sort R. (For typographical clarity, we no longer use the asterisks.) When we say that $+$, for example, is over R we mean that $sort(+) = \langle R, R, R\rangle$; that is to say that $+$ is interpreted as a function taking pairs of individuals of sort R into individuals of sort R.
(2) For each rational number q, a constant $\boldsymbol{q}$.
(3) Two function symbols, $Left\_End$ and $Right\_End$, are defined on E and give values in R. (The intention is to use them to give for every event e its temporal interval $(Left\_End(e), Right\_End(e))$).
(4) A unary function $Value$ defined on E and giving values in R. (The intention is to use it for the values of the measurement events.)
(5) Two unary function symbols I and A. $I(t)$ gives the true light intensity at time t, and $A(t)$ is the integral of I from some start-up time t_0 to t (so t_0 is a constant in this signature). A more familiar notation for $A(t)$ would be $\int_{t_0}^{t} I(x)dx$, but of course it is too far from our syntax.

A standard real number interpretation $\mathcal{M}$ of K is a structure containing the real numbers $\Re$ (interpreting the sort R) and a set $E^{\mathcal{M}}$ of "events". The interpretation of the arithmetical operations is the standard interpretation on $\Re$ (for example, $<^{\mathcal{M}}$ is the natural ordering on $\Re$). For each individual e in $E^{\mathcal{M}}$, $Left\_End^{\mathcal{M}}(e)$, $Right\_End^{\mathcal{M}}(e)$, $Value^{\mathcal{M}}(e)$ are real numbers with $Left\_End^{\mathcal{M}}(e) <^{\mathcal{M}} Right\_End^{\mathcal{M}}(e)$. The interpretation of I is an arbitrary (integrable) real-valued function $I^{\mathcal{M}}$, and $A^{\mathcal{M}}$ is its integral (from time $t_0^{\mathcal{M}} \in \Re$).

Of course there is nothing in the symbols to force this particular interpretation, and many others are possible. This example is brought here just to give some idea of the diversity of situations where multi-sorted structures can be used, and though it is used again in the next section it will not be used later on.

Two-sorted structures are useful to handle pairs and finite sequences, as the second example shows. This will be significant for later development in the book. Usually we write pairs with angled brackets $\langle a, b\rangle$, but for simplicity of expression we often use round brackets (a, b).

In set theory one learns how to form pairs as sets. A pair $\langle a, b\rangle$ can be defined as a set $\{\{a\}, \{a, b\}\}$. We do not use this (or any other) particular representation here as only the abstract properties of pairs and finite sequences are needed. We shall define now the language in which these abstract properties are expressed.