Applied Mathematics and Computation 215 (2009) 1386–1403

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Unifying computers and dynamical systems using the theory of synchronous concurrent algorithms

B.C. Thompson [a], J.V. Tucker [a], J.I. Zucker [b,*]

[a] Department of Computer Science, Swansea University, Singleton Park, Swansea SA2 8PP, Wales, United Kingdom
[b] Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada L8S 4K1

ARTICLE INFO

Keywords:
Synchronous concurrent algorithms
Dynamical systems
Many sorted algebras
Equational specifications
Streams
Computability on topological algebras
Computable physical systems

ABSTRACT

A *synchronous concurrent algorithm* (SCA) is a parallel deterministic algorithm based on a network of modules and channels, computing and communicating data in parallel, and synchronised by a global clock with discrete time. Many types of algorithms, computer architectures, and mathematical models of physical and biological systems are examples of SCAs. For example, conventional digital hardware is made from components that are SCAs and many computational models possess the essential features of SCAs, including systolic arrays, neural networks, cellular automata and coupled map lattices.

In this paper we formalise the general concept of an SCA equipped with a global clock in order to analyse precisely (i) specifications of their spatio-temporal behaviour; and (ii) the senses in which the algorithms are correct. We start the mathematical study of SCA computation, specification and correctness using methods based on computation on many-sorted topological algebras and equational logic. We show that specifications can be given equationally and, hence, that the correctness of SCAs can be reduced to the validity of equations in certain computable algebras. Since the idea of an SCA is general, our methods and results apply to each of the particular classes of algorithms and dynamical systems above.

1. Introduction

1.1. The concept

A *synchronous concurrent algorithm* (SCA) is an algorithm based on a network of modules and channels, computing and communicating data in parallel, and synchronised by a global clock with discrete time. The etymology of 'synchronous' is Greek: "at the same time". SCAs can process infinite streams of input data and return infinite streams of output data. Most importantly, an SCA is a *parallel deterministic algorithm*.

Many types of algorithms, computer architectures, and mathematical models of physical and biological systems are examples of SCAs. First and foremost, conventional digital hardware, including all forms of serial and parallel computers and digital controllers, are made from components that are SCAs. In many cases, complete specifications of computers at different levels of abstraction are SCAs. Interestingly, the structure of Charles Babbage's Analytical Engine (developed from 1833 onwards) is that of an SCA.

Further, many specialised models of computation possess the essential features of SCAs, including *systolic arrays*, *neural networks*, *cellular automata* and *coupled map lattices*.

* Corresponding author.
E-mail addresses: J.V.Tucker@swansea.ac.uk (J.V. Tucker), zucker@mcmaster.ca (J.I. Zucker).

0096-3003/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2009.04.058

The parallel algorithms, architectures and dynamical systems that comprise the class of SCAs have many applications, ranging from their use in special purpose devices (for communication and signal processing, graphics and process control) to computational models of biological and physical phenomena.

From the point of view of computing, an SCA can be considered to be a type of *deterministic* data flow network, in which time is explicit and enjoys a primary role. SCAs require a new specialised mathematical theory with applications of its own.

From the point of view of mathematical physics and biology, an SCA can be considered to be a type of spatially extensive discrete space, discrete time, deterministic dynamical system that is studied independently or as an approximation to continuous space, continuous time dynamical systems.

In most cases, SCAs are complicated and require extensive simulation and mathematical analysis to understand their operation, behaviour and verification. In fact, in the independent literatures on the above types of SCAs it is often difficult to formulate precisely

(i) specific SCAs and their operation in time;
(ii) specifications of their spatio-temporal behaviour; and
(iii) the senses in which the algorithms are correct.

In the case of neural networks, correctness is further complicated by the difficulty of writing problem specifications, the existence of a learning phase, and notions of approximate correctness. In the case of non-linear dynamical systems, correctness is concerned with properties such as chaotic, stable, emergent and coherent behaviour over time. Thus, SCAs constitute a wide ranging class of useful algorithms for which many basic questions concerning their structure and design remain unanswered.

In this paper, we formalise the general concept of an SCA equipped with a global clock and analyse precisely ideas about the specification and correctness of SCAs. Our mathematical study of SCA computation, specification and correctness provides a unified theory of deterministic parallel computing systems and deterministic, spatially extensive, non-linear dynamical systems.

The methods are based on abstract computability theory on many-sorted topological algebras and equational logic. We show how to define SCAs by equations over stream algebras in a simple way. We also show that specifications can be given equationally and, hence, that the correctness of SCAs can always be reduced to the validity of equations in certain algebras. Thus, a natural method for verification of SCAs is equational reasoning, although this is incomplete.

Our methods and results apply to each of the classes of algorithms and architectures listed above. In particular, they can be used in case studies and software tools for design and verification of specific classes of SCAs, and as a starting point for a general theoretical analysis of hardware verification.

1.2. The theory

Data is modelled by an algebra

$$A = (A, \mathbb{B}, \mathbb{T};\ F_1, \ldots, F_k)$$

with three carrier sets: the set A of data, $\mathbb{B}$ of Booleans and $\mathbb{T}$ of naturals $\{0, 1, 2, \ldots\}$ (written $\mathbb{T}$ instead of $\mathbb{N}$ because it represents the discrete time on the global clock), and functions $F_1, \ldots, F_k$ which include the standard Boolean operations (with possibly equality on A) and the arithmetic operations of 0 and successor $t + 1$.

The behaviour of SCAs in time is modelled using *streams* of elements of A, which are infinite sequences indexed by (discrete) time. Let $[\mathbb{T} \to A]$ be the set of all streams. The operations on data, time and streams are combined to form a *stream algebra*:

$$\overline{A} = (A, \mathbb{B}, \mathbb{T}, [\mathbb{T} \to A]; F_1, \ldots, F_k, \mathsf{eval}).$$

Typically, in models of hardware systems, SCAs compute with streams of bits, integers or terms. In dynamical systems, SCAs compute with streams of real and complex numbers. To prepare for this mathematical view, we provide some preliminaries on topological algebras in Section 2 and stream algebras and computable algebras in Section 3. We note that all stream algebras are topological algebras and often have certain dense subalgebras that are computable.

In Section 4, we define synchronous concurrent algorithms and architectures and formalise their semantics by means of functions defined by *simultaneous primitive recursion equations* over $\overline{A}$.

More specifically, an SCA based on a network N with m modules and p input streams is specified by a network state function

$$\mathsf{V}^N : A^m \times [\mathbb{T} \to A]^p \times \mathbb{T} \to A^m$$

in which $\mathsf{V}^N(\mathsf{a}, \mathsf{x}, t)$ denotes the state of the SCA on processing p input streams $\mathsf{x} \in [\mathbb{T} \to A]^p$ from initial state $\mathsf{a} \in A^m$ at time $t \in \mathbb{T}$.

In Section 5, we give a sketch of the broad range of types of SCAs (systolic arrays, neural networks, cellular automata and coupled map lattices) with an bibliography.

In Section 6, we consider specifications and correctness criteria for a simple form of the space-time behaviour of SCAs: correctness based on specifications with respect to a single system clock of the SCA. Other forms of correctness are possible, such as correctness based on specifications with respect to a second clock external to the SCA [30,60].