
Preface

In the world we live in concurrency is the norm. For example, the human body is a massively concurrent system, comprising a huge number of cells, all simultaneously evolving and independently engaging in their individual biological processing. In addition, in the biological world, truly sequential systems rarely arise. However, they are more common when manmade artefacts are considered. In particular, computer systems are often developed from a sequential perspective. Why is this? The simple reason is that it is easier for us to think about sequential, rather than concurrent, systems. Thus, we use sequentiality as a device to simplify the design process.

However, the need for increasingly powerful, flexible and usable computer systems mitigates against simplifying sequentiality assumptions. A good example of this is the all-powerful position held by the Internet, which is highly concurrent at many different levels of decomposition. Thus, the modern computer scientist (and indeed the modern scientist in general) is forced to think about concurrent systems and the subtle and intricate behaviour that emerges from the interaction of simultaneously evolving components.

Over a period of 25 years, or so, the field of concurrency theory has been involved in the development of a set of mathematical techniques that can help system developers to think about and build concurrent systems. These theories are the subject matter of this book.

Our motivation in writing this book was twofold. (1) We wished to synthesise into a single coherent story, a body of research that is scattered across a set of journal and conference publications. (2) We have also sought to highlight newer research (mainly undertaken by the authors) on concurrency theory models of real-time systems. The first of these aspects yields the text book style of the first three parts of the book, whereas the second has motivated the approach of the fourth part, which has more of the flavour of a research monograph.

There are other books on concurrency theory, but these have tended to have a different focus from this book. Most relevant in this respect are classic works by Milner on the Calculus of Communicating Systems (CCS) [148],

Hoare on first generation Communicating Sequential Processes (CSP) [96], Roscoe on the mature CSP theory [171] and Schneider on Timed CSP [176]. However, all of these have a tighter focus than this book, being directed at specific theories. Although one point of major focus in this book is the process calculus LOTOS (which, by the way, has not previously been presented in book format), our approach is broader in scope than these earlier texts. For example, we consider both untimed and timed approaches (in the same book), we highlight the process calculus approach along with communicating finite and infinite state automata and we present a spectrum of different semantic theories, including traces, transition systems, refusals and true concurrency models. The latter of these semantic models being particularly noteworthy, because the bundle event structure true concurrency theory we consider is not as well known as it should be.

Another difference with previous concurrency theory texts is that this book is less focused on proof systems. There are a number of reasons for this. First, proof systems are not as well behaved in LOTOS as they are in CCS and CSP; e.g. testing equivalence is not a congruence in LOTOS. Second, we would argue that the issue of finding complete proof systems has actually turned out to be less important than once seemed to be the case. This is because of the development of powerful state-space exploration methods, such as model-checking and equivalence-checking, which are not proof system dependent. As a reflection of this trend, we also consider finite and infinite state communicating automata approaches, which have recently taken a prominent place in concurrency theory, because of their amenability to formal verification. These techniques were not considered in the previous process calculus texts.

Due to the breadth of scope that we have sought in this book, by necessity, certain topics have had to be treated in less depth than would be optimum. As just discussed, one of these is the topic of proof systems. In addition, when, in a denotational style, we interpret recursive definitions semantically, we do not present the full details of the fixed point theories that we use. However, at all such points in the text, we give pointers to the required definitions and include references to complete presentations of the necessary theory.

In terms of target readership, this book is partially a textbook and partially a research monograph. It is particularly suitable for masters and doctoral level programmes with an emphasis on parallel processing, distributed systems, networks, formal methods and/or concurrency in general. We assume a basic knowledge of set theory, logic and discrete mathematics, as found in a textbook such as [86]. However, we do include a list of notation to help the reader.

The material presented here has partially grown out of a set of course notes used in an MSc-level course on formal methods taught in the Computing Laboratory at the University of Kent. Consequently, we would like to thank the students who have taken this course over a number of years. The feedback from these students has helped these notes to be refined, which has, in turn, benefited this book.

We would also like to thank a number of our academic colleagues with whom we have discussed concurrency theory and who have contributed to the development of our understanding of this field. We would particularly like to mention Juan Carlos Augusto, Gordon Blair, Lynne Blair, Eerke Boiten, Tommaso Bolognesi, Jeremy Bryans, Amanda Chetwynd, John Derrick, Giorgio Faconti, Holger Hermanns, Joost-Pieter Katoen, Rom Langerak, Diego Latella, Su Li, Peter Linington, Mieke Massink, Tim Regan, Steve Schneider, Marteen Steen, Ben Strulo, Simon Thompson, Stavros Tripakis and Frits Vaandrager.

In addition, we would like to acknowledge the contribution of the following funding bodies who have provided financial support for our concurrency theory research over the last ten years: the UK Engineering and Physical Sciences Research Council, British Telecom, the European Union, under the Marie Curie and ERCIM programmes, Universities UK, through the Overseas Research Fund, and the Computing Laboratory at the University of Kent.

Finally, we would like to thank Catherine Drury and the Springer publishing team for their efficiency and patience with us.

Canterbury, Kent, UK,
June 2005

Howard Bowman
Rodolfo Gomez

Notation

The following is an account of some symbols commonly found in this book.

Numbers

- \mathbb{N} : the natural numbers
- \mathbb{Z} : the integer numbers
- \mathbb{R} : the real numbers
- \mathbb{R}^+ : the positive real numbers
- \mathbb{R}^{+0} : the positive real numbers, including zero

Sets and Functions

- $|S|$: cardinality of (i.e. number of elements in) S
- $\mathcal{P}(S)$: powerset of S (i.e. the set of all possible sets containing elements of S)
- \subseteq (\subset) : set inclusion (proper set inclusion)
- \cup (\bigcup) : set union (generalised union)
- \cap (\bigcap) : set intersection (generalised intersection)
- \setminus : set difference
- \times : Cartesian product: $S_1 \times S_2 \times \cdots \times S_n = \{(s_1, s_2, \dots, s_n) \mid s_i \in S_i\}$
- $^{-1}$: inverse relation: $b R^{-1} a$ iff $a R b$
- \lceil : domain restriction: given $f : \mathcal{D} \rightarrow \mathcal{R}$, then $f \lceil S$ is s.t. $f \lceil S : \mathcal{D} \cap S \rightarrow \mathcal{R}$ and $f(x) = f \lceil S(x)$ for all $x \in \mathcal{D} \cap S$

where S, S_1, S_2, \dots, S_n are sets, R is a binary relation and f is a function.

Logic

- \wedge (\bigwedge) : conjunction (generalised conjunction)
- \vee (\bigvee) : disjunction (generalised disjunction)
- \neg : negation
- \implies : implication
- \iff : double implication
- \forall : universal quantification
- \exists : existential quantification
- \models : satisfiability relation

General Abbreviations and Acronyms

- \triangleq : “defined as”
- iff : “if and only if”

s.t.	: “such that”
w.r.t.	: “with respect to”
LTS	: “Labelled Transition System”
BES	: “Bundle Event Structure”
TBES	: “Timed Bundle Event Structure”
TTS	: “Timed Transition System”
CA	: “(finite state) Communicating Automata”
ISCA	: “Infinite State Communicating Automata”
TA	: “Timed Automata”
DTA	: “Discrete Timed Automata”
RSL	: “Ready Simulation Logic”
HML	: “Hennessy-Milner Logic”
LOTOS	: “Language of Temporal Ordering Specification”
CCS	: “Calculus of Communicating Systems”
CSP	: “Communicating Sequential Processes”
pomset	: “partially ordered multiset”
lposet	: “labelled partially ordered set”

Process Calculi (Chapters 2,3,4,5,6,9 and 10)

Sets

In LOTOS

<i>Defs</i>	: the set of LOTOS definitions
<i>DefList</i>	: the set of lists of LOTOS definitions
<i>tDeflist</i>	: the set of tLOTOS definition lists
<i>PIdent</i>	: the set of process identifiers
<i>Beh</i>	: the set of LOTOS behaviours
<i>tBeh</i>	: the set of tLOTOS behaviours
<i>Der(B)</i>	: the set of behaviours that can be derived from B
<i>Act</i>	: the set of actions
\mathcal{L}	: the set of actions occurring in a given specification
$\mathcal{A}(B)$: the set of actions which arise in B
<i>Gate</i>	: the set of gates
<i>SN</i>	: the set of semantic notations
<i>SM</i>	: the set of semantic mappings
<i>DEV</i>	: the set of development relations
\mathcal{T}	: the set of traces
A^*	: the set of traces from actions in A
<i>Tr(S)</i>	: the set of traces that can be derived from S
\mathcal{LTS}	: the set of labelled transition systems
\mathcal{TTS}	: the set of timed transition systems
$Ref_B(\sigma)$: the set of refusals of B after σ

XII Preface

$S(B)$: the set of (RSL) observations that B can exhibit
 Ξ : the set of time intervals

where B is a behaviour, σ is a trace and S is an LTS.

In Bundle Event Structures

BES : the set of bundle event structures
 $TBES$: the set of timed bundle event structures
 \mathcal{U}_E : the universe of events
 $\$ \rho$: the set of events underlying ρ
 $\mathbf{cfl}(\rho)$: the set of events that are disabled by some event in ρ
 $\mathbf{sat}(\rho)$: the set of events that have a causal predecessor in ρ
 for all incoming bundles
 $\mathbf{en}(\rho)$: the set of events enabled after ρ
 $\mathcal{PS}(\varepsilon)$: the set of proving sequences of ε
 $\mathcal{CF}(\varepsilon)$: the set of configurations of ε
 $\mathbf{L}(X)$: the multiset of labels of events in X
 $Tr^{st}(\varepsilon)$: the set of step traces of ε
 $\mathcal{LP}(\varepsilon)$: the set of lposets of ε
 $PoS(\varepsilon)$: the set of pomsets of ε

where ρ is a proving sequence, ε is a BES and X is a set of events.

Relations (Functional and Nonfunctional)

In Traces and Labelled Transition Systems

$\llbracket \cdot \rrbracket$: a semantic map
 $\llbracket \cdot \rrbracket_{tr}$: the trace semantic map
 $\llbracket \cdot \rrbracket_{lts}$: the LTS semantic map
 $\llbracket \cdot \rrbracket_{tts}$: the TTS semantic map
 \leq_{tr} : trace preorder
 \asymp : equivalence
 \asymp_{tr} : trace equivalence
 \prec : simulation
 \asymp_{\prec} : simulation equivalence
 \prec_R : ready simulation
 \sim_R : ready simulation equivalence
 \sim : strong bisimulation
 \sim_t : timed strong bisimulation
 \approx : weak bisimulation (or observational) equivalence
 \approx_c : weak bisimulation congruence
 \approx_t : timed weak bisimulation
 \approx_r^t : timed rooted weak bisimulation

In Bundle Event Structures

- $\llbracket \cdot \rrbracket_{be}$: the BES semantic mapping
- $\llbracket \cdot \rrbracket_{tbe}$: the TBES semantic mapping
- ψ : a mapping from BES to \mathcal{LTS}
- \sim_{sq} : sequential strong bisimulation
- \sim_{st} : step strong bisimulation
- \preceq : the causality partial order induced by bundles
- \preceq_C : the causality partial order \preceq restricted to C
- \otimes_C : the independence relation between events w.r.t. C
- \prec_{st} : step trace equivalence
- \prec_{PoS} : pomset equivalence
- \prec_{PS} : proving sequence isomorphism
- \simeq : the isomorphism between lposets

where C is a configuration.

In Testing Theory

- $te (te^s)$: testing equivalence (stable testing equivalence)
- $conf (conf^s)$: conformance (stable conformance)
- $red (red^s)$: reduction (stable reduction)
- $ext (ext^s)$: extension (stable extension)

Transitions

In Labelled Transition Systems

- $B \xrightarrow{a} B'$: B evolves to B' after a
- $B \xRightarrow{\sigma} B'$: B evolves to B' after σ ($\sigma \neq i$)
- $B \xRightarrow{\sigma} B'$: B evolves to B' after σ ($\sigma = i$ is allowed)
- $B \xRightarrow{\sigma} B'$: B evolves to B' after σ ($\sigma = i$ is allowed but $B \neq B'$)
- $B \overset{a}{\rightsquigarrow} B'$: B evolves to B' after a (considers undefinedness)

where B, B' are LOTOS behaviours, a is an action and σ is a trace.

In Bundle Event Structures

- $\bullet \xrightarrow{a}$: denotes a sequential transition generated from a BES
- $\blacktriangleright \xrightarrow{A}$: denotes a step transition generated from a BES

In Timed Transition Systems

$B \xrightarrow{t} B'$: B evolves to B' after t
$B \xrightarrow{a} B'$: B evolves to B' after a
$s \xrightarrow{a} s'$ ($s \xrightarrow{t} s'$)	: a TTS action (time) transition
$B \xrightarrow{v} B'$: B evolves to B' after v
$B \xRightarrow{\sigma}_t B'$: B evolves to B' after σ

where B, B' are tLOTOS behaviours, s, s' are states, t is a delay, a is an action, v is either an action or a delay, and σ is either a trace, an internal action, or a delay.

Other symbols and acronyms

In LOTOS

pbLOTOS	: <i>primitive basic LOTOS</i> , a subset of bLOTOS
bLOTOS	: <i>basic LOTOS</i> , the complete language without data types
fLOTOS	: <i>full LOTOS</i> , the complete language with data types
i	: the internal action
δ	: the successful termination action
$ $: independent parallel composition
$ $: fully synchronised parallel composition
$[[G]]$: parallel composition with synchronisation set G
\square	: choice
$;$: action prefix
$>>$: enabling
$[>$: disabling
ϵ	: the empty trace
Ω	: a LOTOS process with completely unpredictable behaviour
\models_{RSL}	: satisfiability under RSL
\models_{HML}	: satisfiability under HML
$[t, t']$: time interval (also $[t, \infty)$, $[t]$ and (t))
\oplus	: time interval addition
\ominus	: time interval subtraction
$initI(a, B)$: the set of intervals where B can initially perform a
$initI\downarrow(A, B)$: the smallest instant where B can initially perform an action in A
$initI\uparrow(A, B)$: the smallest of the set of all maximum time points where B can initially perform an action in A

where B is a tLOTOS behaviour, a is an action and A is a set of actions.

In Bundle Event Structures

E	: the set of events of a given BES
$\#$: the set of conflicts of a given BES
\mapsto	: the set of bundles of a given BES
l	: the labelling function of a given BES
$\varepsilon[C]$: the remainder of ε after C
\overline{C}	: the lposet corresponding to C
$[\overline{C}]_{\sim}$: the pomset corresponding to the lposet \overline{C}
\mathcal{A}	: the event-timing function
\mathcal{R}	: the bundle-timing function
$init(\Psi)$: the set of initial events of Ψ
$exit(\Psi)$: the set of successful termination events of Ψ
$res(\Psi)$: the events of Ψ whose timing is restricted
$rin(\Psi)$: the set of initial and time restricted events of Ψ
$X \xrightarrow{I} e$: a timed bundle
$\mathcal{Z}(\sigma, e)$: the set of instants where (enabled event) e could happen, after σ

where ε is a BES, C is a configuration, Ψ is a TBES, X is a set of events, I is a time interval, e is an event and σ is a timed proving sequence.

Automata (Chapters 8, 11, 12 and 13)

Sets

In Communicating Automata, Timed Automata and Timed Automata with Deadlines

Act	: the set of action labels
$CAct$: the set of labels for completed actions
$HAct$: the set of labels for half actions
$CommsAut$: the set of product automata (CA)
TA	: the set of timed automata
L	: the set of locations in a given automaton
TL	: the set of transition labels of a given automaton
T	: the transition relation of a given automaton
\mathbb{C}	: the set of clocks
CC	: the set of clock constraints
C	: the set of clocks of a given automaton
CC_C	: the set of clock constraints restricted to clocks in C
$Clocks(\phi)$: the set of clocks occurring in the constraint ϕ
\mathbb{V}	: the space of clock valuations
\mathbb{V}_C	: the space of valuations restricted to clocks in C
$Runs(A)$: the set of runs of A
$ZRuns(A)$: the set of zeno runs of A

$Loops(A)$: the set of loops in A
$Loc(lp)$: the set of locations of lp
$Clocks(lp)$: the set of clocks occurring in any invariant of lp
$Trans(lp)$: the set of transitions of lp
$Guards(lp)$: the set of guards of lp
$Resets(lp)$: the set of clocks reset in lp
$Act(lp)$: the set of transition labels in lp
$HL(A)$: the set of pairs of matching half loops in $ A$
$CL(A)$: the set of completed loops in $ A$
$Esc(lp)$: the set of escape transitions of lp

where A is an automaton, $|A$ is a network of automata and lp is a loop.

In Infinite State Communicating Automata, Discrete Timed Automata and Fair Transition Systems

\mathcal{A}	: the set of actions of a given automaton
$COMP(\mathcal{A})$: the set of completed actions in \mathcal{A}
$IN(\mathcal{A})$: the set of input actions in \mathcal{A}
$OUT(\mathcal{A})$: the set of output actions in \mathcal{A}
V	: the set of variables in a given automaton or fair transition system
$V'(e)$: the set of variables modified by effect e
V_L	: the set of local variables of a given automaton
V_S	: the set of shared variables of a given automata network
Θ	: the initialisation formula
Θ_L	: the initialisation formula for variables in V_L
Θ_S	: the initialisation formula for variables in V_S

Transitions

In Communicating Automata, Timed Automata and Timed Automata with Deadlines

$l \xrightarrow{a} l'$: a CA transition
$s \xrightarrow{a} s'$: an LTS transition
$l \xrightarrow{a,g,r} l'$: a TA transition
$l \xrightarrow{a,g,d,r} l'$: a TAD transition
$s \xrightarrow{\gamma} s'$: a TTS transition from s to s'

where l, l' are automata locations, s, s' are states, γ is either an action or a delay, a is an action label, g is a guard, r is a reset set and d is a deadline.

Relations

Δ_1 : the mapping from product automata (CA) to pbLOTOS specifications
 Δ_2 : a mapping from product automata (CA) to CCS^{CAct} specifications

Other Symbols and Acronyms

CCS^{CAct} : a CCS variant with completed actions
 $|A$: a network of automata
 $\langle u_1, \dots, u_n \rangle$: a location vector
 $u[l \rightarrow j]$: substitution of locations (the j th component in u , by location l)
 \setminus^{CAct} : the CCS^{CAct} restriction operator
 Π^{CAct} : the CCS^{CAct} parallel composition operator
 E_V : an expression on variables in V
 $\llbracket v \rrbracket_s$ ($\llbracket E_V \rrbracket_s$) : the value of variable v (expression E_V) in state s
 l_0 : the initial location of a given automaton
 $I(l)$: the invariant of l , where l is a location of a given TA
 $[l, v]$: a state with location l and valuation v
 (l, Z) : a symbolic state with location l and zone Z
 $r(Z)$: reset of zone Z w.r.t. reset set r
 Z^\uparrow : forward projection of zone Z
 $norm(Z)$: normalisation of zone Z

Concurrency Theory

Calculi and Automata for Modelling Untimed and Timed
Concurrent Systems

Bowman, H.; Gomez, R.

2006, XX, 422 p. 126 illus., Hardcover

ISBN: 978-1-85233-895-4