A Brief History of Process Algebra

J.C.M. Baeten
Technische Universiteit Eindhoven
The Netherlands

—

presented by

Robert Staudinger
University of Salzburg
Austria

Introduction

- What is Process Algebra?
- History, most important algebras
- Recent developments

What is Process Algebra? (1)

- Roots in automata theory (states, transitions)
- Equivalence: language equivalence
- Algebra: regular expressions *
- Lacking in concurrency and interaction

- Process: behaviour of a system
- P. algebra: axiomatic approach to describe processes *
- Axioms allow for calculations
- Equivalence: bisimilarity

What is Process Algebra? (2)

Basic operators

- $+$ alternative composition (weakest binding)
- $||$ parallel composition
- $;$ sequential composition (strongest binding)

Operator syntax differs among algebras

What is Process Algebra? (3)

Basic laws

- $x + y = y + x$ (commutativity)
- $x + (y + z) = (x + y) + z$ (associativity: alternative)
- $x + x = x$ (idempotency)
- $(x + y); z = x; z + y; z$ (right distributivity of + over ;)
- $(x; y); z = x; (y; z)$ (associativity: sequential)
- $xy = yx$ (commutativity of parallel composition)
- $(xy)z = x(yz)$ (associativity: parallel)

Interleaving vs. *true concurrency* systems (expansion theorem)

History

1. Operational semantics (McCarthy)
   - Computer: abstract machine
   - State: variables
   - Transition: elementary program instruction
2. Denotational semantics (Scott, Strachey)
   - More abstract
   - Program model is function turning input into output
3. Axiomatic semantics (Floyd, Hoare)
   - Emphasis on proving programs correct
   - Proof triple: precondition, statement, postcondition *

History: Bekič

- First to address semantics of *quasi-parallel execution*

$$
\begin{aligned}
&(A//B)\xi = \\
&\qquad (cases\ \ A\xi : null \longrightarrow B\xi \\
&\qquad\qquad\quad (f, A') \longrightarrow f, (A'//B)) \\
&\qquad \sqcup \\
&\qquad (cases\ \ B\xi : null \longrightarrow A\xi \\
&\qquad\qquad\quad (g, B') \longrightarrow g, (A//B'))
\end{aligned}
$$

History: CCS

- Calculus of Communicating Systems
- Robin Milner, central person on PA
- Uses * for sequential, ? for alternative, || for parallel composition
- Fomulates basic CCS with Henessy
- Observational- and strong equivalence defined inductively
- Henessy-Milner logic: logical characterisation of process equivalence

History: CSP

- Communicating Sequential Processes
- Tony Hoare, now at MSR
- First to introduce message passing instead of global variables *
- Prevents deadlocks, influenced Milner/CCS

? History: ACP

- Algebra of Communicating Processes
- Bergstra, Klop, 1982
- First to use term *process algebra*

Axioms ($+$ union, $\cdot$ composition, $\between$ left merge)

$$
\begin{aligned}
x + y &= y + x \\
x + (y + z) &= (x + y) + z \\
x + x &= x \\
(xy)z &= x(yz) \\
(x + y)z &= xz + yz \\
(x + y) \between z &= x \between z + y \between z \\
ax \between y &= a(x \between y + y \between x) \\
a \between y &= ay
\end{aligned}
$$

Developments: Theory

- Bisimulation is central notion of equivalence in PA.

- Def: (Strong Bisimulation) A binary relation $\mathcal{R}$ over the set of states $s_i \in S$ of an LTS is a *bisimulation* iff whenever $s_1 \mathcal{R} s_2$ and α is an action:
  - If $s_1 \stackrel{\alpha}{\rightarrow} s_1'$, then $\exists$ a transition $s_2 \stackrel{\alpha}{\rightarrow} s_2'$ such that $s_1' \mathcal{R} s_2'$
  - If $s_2 \stackrel{\alpha}{\rightarrow} s_2'$, then $\exists$ a transition $s_1 \stackrel{\alpha}{\rightarrow} s_1'$ such that $s_1' \mathcal{R} s_2'$

  Two bisimilar states are written $s_1 \sim s_2$

- Operational semantics of CCS given in terms of LTS whose states are process expression: definition applies.

Developments: Time

- Variants of CCS, CSP, ACP with quantitative notion of time
- Formulations of bisimulation taking time into account
- Used e.g. in protocol verification
- Very strict, notion of *approximation* desireable

Developments: Mobility

- Networks of mobile processes are being researched
- Examples: π calculus, ambient calculus
- Interesting area to watch, e.g. recent submission of *save boxed ambient calculus* implementation to DATE
- Is a VM for executing processes on networks, according to a behavioural spec in highlevel syntax

Conclusion

- Early work focused on programming languages, parallel constructs
- Breakthrough 1: from process as function to provision for intermediate states
- Breakthrough 2: replacement of global state with message passing
- PAs extended with data, time, mobility, probability, stochastics over time
- Yet challenges exist, would lead too far, get in touch with Ana