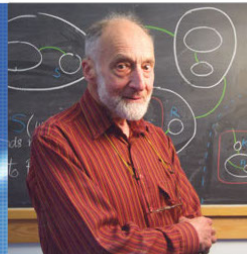


Robin Milner, 1934-2010

Concurrency: interaction, bisimulation, naming



Alan Jeffrey

Bell Labs, Enabling Computing Technologies Research

January 2011, ACM Principles of Programming Languages

PROOF TABLE FOR RELATIONAL COMPOSITION

Form	B'	B	B	B
Identity	$R \subseteq B'$	$L(R) \subseteq L(B')$	$L(R) \subseteq L(B)$	$PV(R) \subseteq PV(B')$
Projection	$B = B'$	$L(B) \subseteq L(B)$	$L(B) \subseteq L(B)$	$PV(B) = \{b_1, \dots, b_n\}$ $PV(B) \subseteq \{b_1, \dots, b_n\}$
Action	$a_1, \dots, a_n \cdot B$ $a_1, \dots, a_n \cdot B$ $\leq B$	$L(B) \subseteq L(B)$	$L(B) \subseteq L(B)$	$PV(B) \subseteq PV(B)$
Composition	$B \circ B'$	$L(B) \subseteq L(B')$	$L(B) \subseteq L(B)$	$PV(B) \subseteq PV(B')$
Restriction	$B \upharpoonright A$	$L(B) \subseteq L(B)$	$L(B) \subseteq L(B)$	$PV(B)$
Relabelling	RE	$L(B) \subseteq L(B)$	$L(B) \subseteq L(B)$	$\{b_1, \dots, b_n\}$
Identification	$b_1, \dots, b_n \cdot B$	$L(B) \subseteq L(B)$	$L(B) \subseteq L(B)$	$PV(B) \subseteq PV(B)$
Conditioned	$\{ \mid B \text{ then } B' \text{ else } B'' \}$	$L(B) \subseteq L(B')$	$L(B) \subseteq L(B'')$	$PV(B) \subseteq PV(B') \cup PV(B'')$

The table shows how B' of each L(R) may be built from B' of some $L(B)$. Furthermore one can be used to prove parallel composition, or to explain situations to avoid excessive use of parentheses we write the operator precedences

(Restriction) \circ Action \circ Composition \circ Restriction \circ Relabelling

Thus for example $B \upharpoonright (B' \circ B'')$ means $\{ \mid (B \upharpoonright (B' \circ B'')) \}$


Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

92

Robin Milner

A Calculus of Communicating Systems



Springer-Verlag
Berlin Heidelberg New York

79

Definition $B \subseteq C$ is always true
 $B \subseteq C$ iff for all x
 (i) if $x \in B$ then $x \in C$
 (ii) if $x \in C$ then for some $C', C \subseteq C'$ and $B' \subseteq C'$
 $B - C$ iff $\exists x \in B, x \notin C$ (i.e. $\neg \forall x (x \in B \Rightarrow x \in C)$)

We leave out the simple proofs that each \subseteq is an equivalence relation, and that $B \subseteq C$ implies $B \subseteq C$ (i.e. the reflexive of \subseteq is transitive).

Proposition 3.1 Show that $B \subseteq C$ implies $B \subseteq C$ for each k , and hence implies $B = C$.

Proposition 3.2 \subseteq is a congruence relation.
 Now particularly, $B = B$ implies
 $B_1 \subseteq B_2 \subseteq C_1 \subseteq C_2 \subseteq B_3 \subseteq C_3 \subseteq B_4 \subseteq C_4 \subseteq B_5 \subseteq C_5 \subseteq B_6 \subseteq C_6 \subseteq B_7 \subseteq C_7 \subseteq B_8 \subseteq C_8 \subseteq B_9 \subseteq C_9 \subseteq B_{10} \subseteq C_{10}$
 and $B_1 \subseteq B_2 \subseteq C_1 \subseteq C_2 \subseteq B_3 \subseteq C_3 \subseteq B_4 \subseteq C_4 \subseteq B_5 \subseteq C_5 \subseteq B_6 \subseteq C_6 \subseteq B_7 \subseteq C_7 \subseteq B_8 \subseteq C_8 \subseteq B_9 \subseteq C_9 \subseteq B_{10} \subseteq C_{10}$ (for all i) implies $B_{10} \subseteq C_{10}$

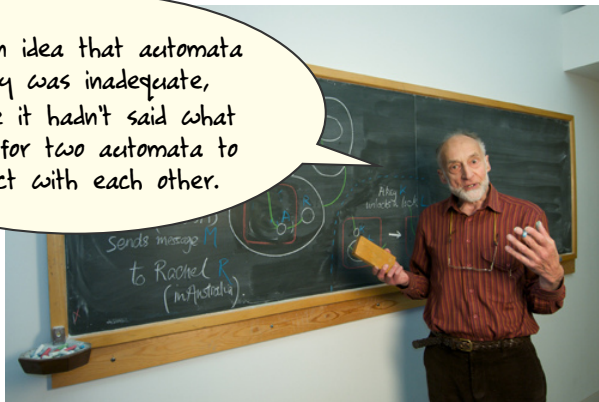
Prop We omit the proof only for composition. We prove by induction on k that

$B_1 \subseteq B_2 \subseteq C_1 \subseteq C_2 \subseteq B_3 \subseteq C_3 \subseteq B_4 \subseteq C_4 \subseteq B_5 \subseteq C_5 \subseteq B_6 \subseteq C_6 \subseteq B_7 \subseteq C_7 \subseteq B_8 \subseteq C_8 \subseteq B_9 \subseteq C_9 \subseteq B_{10} \subseteq C_{10}$
 for $k = 0$ it is trivial. We assume $B_1 \subseteq B_2 \subseteq C_1 \subseteq C_2 \subseteq B_3 \subseteq C_3 \subseteq B_4 \subseteq C_4 \subseteq B_5 \subseteq C_5 \subseteq B_6 \subseteq C_6 \subseteq B_7 \subseteq C_7 \subseteq B_8 \subseteq C_8 \subseteq B_9 \subseteq C_9 \subseteq B_{10} \subseteq C_{10}$ for all i such that $B_i \subseteq C_i$

There are three cases:
 (i) $B_1 \subseteq B_2 \subseteq C_1 \subseteq C_2 \subseteq B_3 \subseteq C_3 \subseteq B_4 \subseteq C_4 \subseteq B_5 \subseteq C_5 \subseteq B_6 \subseteq C_6 \subseteq B_7 \subseteq C_7 \subseteq B_8 \subseteq C_8 \subseteq B_9 \subseteq C_9 \subseteq B_{10} \subseteq C_{10}$ by (i) and (ii)
 (ii) $B_1 \subseteq B_2 \subseteq C_1 \subseteq C_2 \subseteq B_3 \subseteq C_3 \subseteq B_4 \subseteq C_4 \subseteq B_5 \subseteq C_5 \subseteq B_6 \subseteq C_6 \subseteq B_7 \subseteq C_7 \subseteq B_8 \subseteq C_8 \subseteq B_9 \subseteq C_9 \subseteq B_{10} \subseteq C_{10}$ for some $B'_1 \subseteq C'_1$
 where $B_1 \subseteq B'_1 \subseteq C_1 \subseteq C_2 \subseteq B_3 \subseteq C_3 \subseteq B_4 \subseteq C_4 \subseteq B_5 \subseteq C_5 \subseteq B_6 \subseteq C_6 \subseteq B_7 \subseteq C_7 \subseteq B_8 \subseteq C_8 \subseteq B_9 \subseteq C_9 \subseteq B_{10} \subseteq C_{10}$ by (i) and (ii)
 (iii) $B_1 \subseteq B_2 \subseteq C_1 \subseteq C_2 \subseteq B_3 \subseteq C_3 \subseteq B_4 \subseteq C_4 \subseteq B_5 \subseteq C_5 \subseteq B_6 \subseteq C_6 \subseteq B_7 \subseteq C_7 \subseteq B_8 \subseteq C_8 \subseteq B_9 \subseteq C_9 \subseteq B_{10} \subseteq C_{10}$ by inductive hypothesis.
 But $B_1 \subseteq B_2 \subseteq C_1 \subseteq C_2 \subseteq B_3 \subseteq C_3 \subseteq B_4 \subseteq C_4 \subseteq B_5 \subseteq C_5 \subseteq B_6 \subseteq C_6 \subseteq B_7 \subseteq C_7 \subseteq B_8 \subseteq C_8 \subseteq B_9 \subseteq C_9 \subseteq B_{10} \subseteq C_{10}$ by inductive hypothesis.

Concurrency = automata + interaction

I had an idea that automata theory was inadequate, because it hadn't said what it was for two automata to interact with each other.



All quotes from Martin Berger's *An Interview With Robin Milner*, 2003.

A Calculus of Communicating Systems

- Existing and simultaneous work on communication and processes.
 - Petri Nets (1962).
 - Dijkstra (1968) Cooperating Sequential Processes.
 - Owicki-Gries (1976) Hoare logic for parallel programs.
 - Lamport (1978) partially ordered events.
 - Pnueli (1979) temporal logic.
 - Hoare (1979) Communicating Sequential Processes.
 - Hennessy and Plotkin (1979) resumptions.
 - ...

A Calculus of Communicating Systems

- Existing and simultaneous work on communication and processes.
- Labelled transition systems.
 - Automata where every state is an acceptor.
 - Alphabet of actions ℓ , coactions $\bar{\ell}$ and τ .
 - Support *interaction*:

$$\frac{P \xrightarrow{\ell} P' \quad Q \xrightarrow{\bar{\ell}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

A Calculus of Communicating Systems

- Existing and simultaneous work on communication and processes.
- Labelled transition systems.
- Process language.
 - Small, inductively defined syntax with recursive processes.
 - Inspired by language of regular expressions; includes interaction.
 - Semantics given operationally (cf. Plotkin) as an LTS.

A Calculus of Communicating Systems

- Existing and simultaneous work on communication and processes.
- Labelled transition systems.
- Process language.
- Inductively defined equivalence of processes.
 - $P \sim_0 Q$.
 - $P \sim_{n+1} Q$ whenever, for all α :
 - If $P \xrightarrow{\alpha} P'$ then, for some Q' , $Q \xrightarrow{\alpha} Q'$ and $P' \sim_n Q'$.
 - If $Q \xrightarrow{\alpha} Q'$ then, for some P' , $P \xrightarrow{\alpha} P'$ and $P' \sim_n Q'$.
 - $P \sim_\omega Q$ whenever, for all n , $P \sim_n Q$.

Now called stratified bisimulation.

A Calculus of Communicating Systems

- Existing and simultaneous work on communication and processes.
- Labelled transition systems.
- Process language.
- Inductively defined equivalence of processes.
 - $P \sim_0 Q$.
 - $P \sim_{n+1} Q$ whenever, for all α :
 - If $P \xrightarrow{\alpha} P'$ then, for some Q' , $Q \xrightarrow{\alpha} Q'$ and $P' \sim_n Q'$.
 - If $Q \xrightarrow{\alpha} Q'$ then, for some P' , $P \xrightarrow{\alpha} P'$ and $P' \sim_n Q'$.
 - $P \sim_\omega Q$ whenever, for all n , $P \sim_n Q$.

Now called stratified bisimulation. Implicit in earlier work:

- Moore's algorithm for DFA minimization (1956).
- Proof of Myhill-Nerode Theorem (1958).

But now a first-class citizen, not a proof or algorithm technicality.

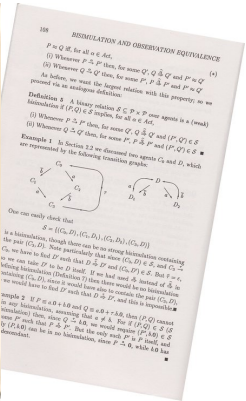
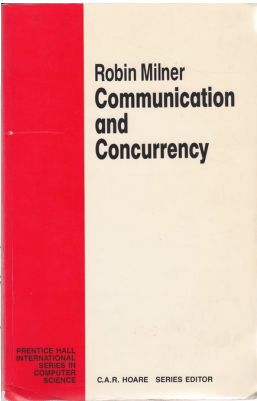
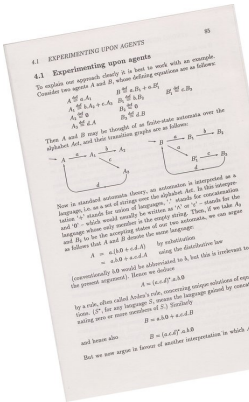
A Calculus of Communicating Systems

- Existing and simultaneous work on communication and processes.
- Labelled transition systems.
- Process language.
- Inductively defined equivalence of processes.
- Results.
 - Stratified bisimulation is a congruence.
 - Sound and complete axiomatization.
 - Logical characterization (Hennessy-Milner logic).

A Calculus of Communicating Systems

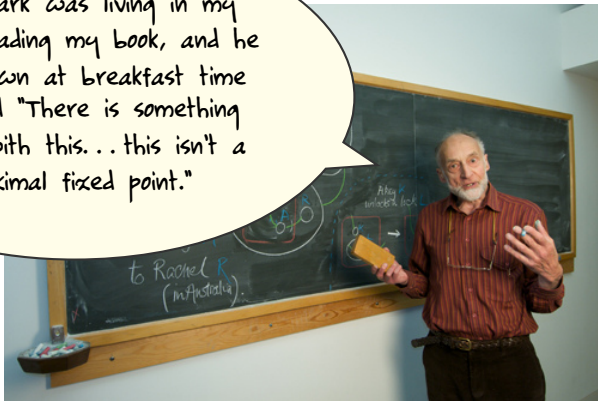
- Existing and simultaneous work on communication and processes.
- Labelled transition systems.
- Process language.
- Inductively defined equivalence of processes.
- Results.
- Spawned a new research area.
 - Papers, books, conferences, research networks. . .
 - Helped establish operational reasoning.

Bisimulation



The most important breakfast in computer science?

David Park was living in my house, reading my book, and he came down at breakfast time and said "There is something wrong with this... this isn't a maximal fixed point."



Communication and Concurrency

- Technical problem: stratified bisimulation is not a fixed point.
 - \sim_ω is not always $\sim_{\omega+1}$.
 - Technical fix: transfinite induction.

Communication and Concurrency

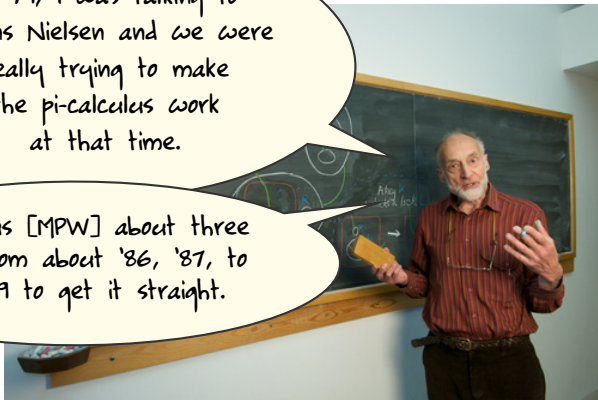
- Technical problem: stratified bisimulation is not a fixed point.
- More importantly, missed an important proof technique.
 - A relation \mathcal{R} is a *bisimulation* whenever, for all $P \mathcal{R} Q$ and α :
 - If $P \xrightarrow{\alpha} P'$ then, for some Q' , $Q \xrightarrow{\alpha} Q'$ and $P' \mathcal{R} Q'$.
 - If $Q \xrightarrow{\alpha} Q'$ then, for some P' , $P \xrightarrow{\alpha} P'$ and $P' \mathcal{R} Q'$.
 - *Bisimilarity*, written $P \sim Q$, is the largest bisimulation.
 - Bisimilarity comes with a proof technique: establish a bisimulation.

Communication and Concurrency

- Technical problem: stratified bisimulation is not a fixed point.
- More importantly, missed an important proof technique.
- Bisimulation has impact far outside concurrency.
 - Coinduction and coinductive datatypes (e.g. streams).
 - Applicative bisimulation for λ -calculi.
 - Coinduction in mechanized proof systems (e.g. Coq and Agda).
 - Non-wellfounded set theory.

In '79, I was talking to Mogens Nielsen and we were really trying to make the pi-calculus work at that time.

It took us [MPW] about three years from about '86, '87, to '88, '89 to get it straight.



- Old problem: gensym.
 - In the Calculus of Communicating Systems, channel scope is static.
 - Cannot model systems with *link mobility*, where channels escape.
 - Causes problems modelling functions, objects, references. . .

- Old problem: gensym.
- Nielsen and Engberg (1986) *Extended CCS*.
 - Name generation modelled by α -equivalence.
 - Scope extrusion: $P \mid \nu(x)Q \sim \nu(x)(P \mid Q)$ when $x \notin \text{fn}(P)$.
 - Complex: names, variables, constants.

- Old problem: gensym.
- Nielsen and Engberg (1986) *Extended CCS*.
- Berry and Boudol (1990) *Chemical Abstract Machine*.
 - Reduction semantics up to a structural congruence:

$$\frac{P \equiv Q \rightarrow Q' \equiv P'}{P \rightarrow P'}$$

- Old problem: gensym.
- Nielsen and Engberg (1986) *Extended CCS*.
- Berry and Boudol (1990) *Chemical Abstract Machine*.
- The π -calculus: Milner, Parrow & Walker (1989), Milner (1992).
 - In 1989, an LTS semantics, with scope extrusion as a bisimilarity.
 - In 1992, a reduction semantics up to a structural congruence.
 - Scope extrusion and α -equivalence *in the structural congruence*.
 - Natural embedding of a λ -calculus with explicit substitutions.

- Old problem: gensym.
- Nielsen and Engberg (1986) *Extended CCS*.
- Berry and Boudol (1990) *Chemical Abstract Machine*.
- The π -calculus: Milner, Parrow & Walker (1989), Milner (1992).
- Wide applicability of π -like techniques:
 - Business processes.
 - Distributed systems.
 - Molecular biology.
 - Nominal set theory.
 - Objects.
 - References.
 - Security.
 - ...

Thank you Robin!

