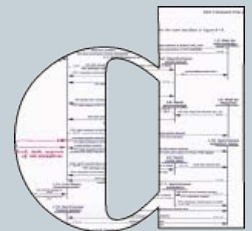


Applied Concurrency Theory

Lecture 4 : bisimulations, CCS, and pi-calculus



Hubert Garavel
Alexander Graf-Brill



Bisimulations

2

Do we need equivalences at all?

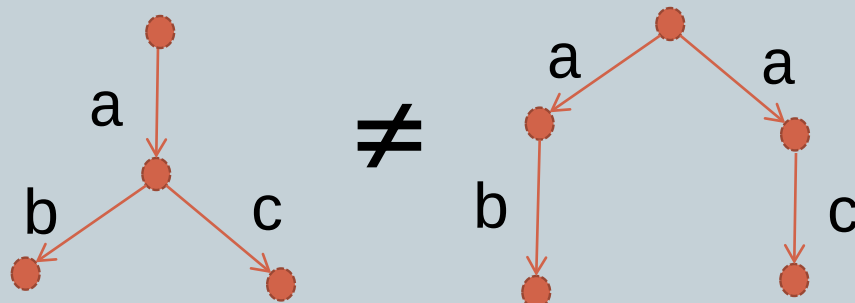
3

- Process algebraists use equivalences because this is the only way for them to verify programs
- With operational semantics:
 - ▶ we translate (well, not too large) programs into graphs
 - ▶ we can do visual checking
 - ▶ we can do model checking
 - ▶ also, equivalences are more expensive than model checking -- roughly: $O(n \log n)$ vs $O(n)$
 - ▶ do we still need equivalences?
- Yes. Equivalences are useful
 - ▶ to minimize LTSs (e.g. before visual or model checking)
 - ▶ to avoid writing complex temporal logic formulas
 - ▶ to check if certain traces are accepted by an LTS
 - ▶ to fight state explosion (compositional minimization)

Why not using automata equivalence?

4

- Automata equivalence checks whether two automata accept the same language
 - ▶ same language = same set of accepted words (or traces)
 - ▶ this is perfect for regular expressions and compiler scanners
- This is not suitable for studying concurrency
 - ▶ comparing languages is not enough
 - ▶ two LTS may have the same language but behave differently



'coffee-vending machine' example

both LTSs recognize the same traces $\{a.b, a.c\}$ but putting them in parallel with $a.b$ generates a deadlock in the 2nd case

Do we need so many equivalences?

5

- In the literature, there are nearly 50 different equivalences for LTSs
- In practice, only two or three are needed:
 - ▶ strong bisimulation: preserves all properties on LTSs (well, not the number of states nor the branching factor)
 - ▶ weak bisimulation: try to eliminate or collapse sequences of τ -transitions which are not observable anyway. Branching bisimulation is a suitable weak bisimulation.
 - ▶ some divergence-preserving bisimulation
- Also useful:
 - ▶ equivalences taking time and/or probabilities into account

A critical look at CCS

6

Syntax of CCS

7

(channel, port) names: $a, b, c, \dots$

co-names: $\bar{a}, \bar{b}, \bar{c}, \dots$

silent action: τ

actions, prefixes: $\mu ::= a \mid \bar{a} \mid \tau$

processes: P, Q	$::=$	0	inaction
		$\mid \mu.P$	prefix
		$\mid P \mid Q$	parallel
		$\mid P + Q$	(external) choice
		$\mid (\nu a)P$	restriction
		$\mid \text{rec}_K P$	process P with definition $K = P$
		$\mid K$	(defined) process name

Dynamic semantics of CCS

8

■ A very small number of rules

$$[\text{Act}] \frac{}{\mu.P \xrightarrow{\mu} P}$$

$$[\text{Res}] \frac{P \xrightarrow{\mu} P' \quad \mu \neq a, \bar{a}}{(\nu a)P \xrightarrow{\mu} (\nu a)P'}$$

$$[\text{Sum1}] \frac{P \xrightarrow{\mu} P'}{P+Q \xrightarrow{\mu} P'}$$

$$[\text{Sum2}] \frac{Q \xrightarrow{\mu} Q'}{P+Q \xrightarrow{\mu} Q'}$$

$$[\text{Par1}] \frac{P \xrightarrow{\mu} P'}{P|Q \xrightarrow{\mu} P'|Q}$$

$$[\text{Par2}] \frac{Q \xrightarrow{\mu} Q'}{P|Q \xrightarrow{\mu} P|Q'}$$

$$[\text{Com}] \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$[\text{Rec}] \frac{P[\text{rec}_K P/K] \xrightarrow{\mu} P'}{\text{rec}_K P \xrightarrow{\mu} P'}$$

A cold look at CCS

9

■ Minimality

- ▶ appealing in academia, but does not scale up to real problems
- ▶ the LOTOS ISO committee added the required extensions

■ Sequential composition

- ▶ CCS action-prefix proved to be a bad language design decision
- ▶ see Lecture 3 for a discussion (LOTOS vs LOTOS NT)

■ Parallel composition

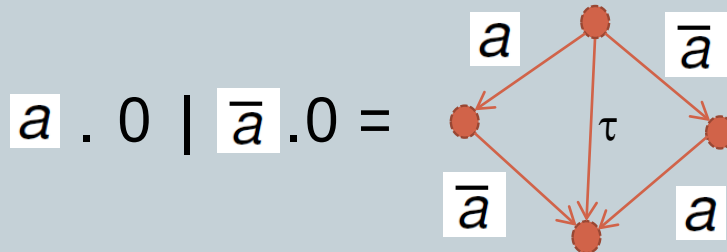
- ▶ CCS parallel composition is worse than the one of CSP/LOTOS
- ▶ only supports binary rendez-vous (co-names are a mistake)
- ▶ even the binary communication is badly designed

CCS parallel composition

10

$$\begin{array}{ll}
 [\text{Par1}] \quad \frac{P \xrightarrow{\mu} P'}{P|Q \xrightarrow{\mu} P'|Q} & [\text{Par2}] \quad \frac{Q \xrightarrow{\mu} Q'}{P|Q \xrightarrow{\mu} P|Q'} \\
 [\text{Com}] \quad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'} & [\text{Res}] \quad \frac{P \xrightarrow{\mu} P' \quad \mu \neq a, \bar{a}}{(\nu a)P \xrightarrow{\mu} (\nu a)P'}
 \end{array}$$

- ▶ No list of gates on which to synchronize or not
- ▶ [Par1] and [Par2]: each parallel process can always evolve alone and ignore the rendez-vous!
- ▶ [Com]: the rendezvous is immediately renamed into τ impossible to observe in the LTS $\Rightarrow$ verification impossible



a restriction on a is required to force the rendezvous

CCS parallel composition: limitations

11

- Limitation of binary synchronization:
how to specify $(P \parallel Q) ; R$? (LOTOS NT semantics)
- This is a 3-party rendez-vous: P and Q wait each other to terminate and R waits to start
- CCS requires 2 additional rendezvous δ_1 and δ_2 :
 $((P . \delta_1 \mid Q . ' \delta_1 . \delta_2) \setminus \delta_1 \mid ' \delta_2 . R) \setminus \delta_2$
this creates two τ -transitions in the LTS (too bad)

The pi-calculus

12

Motivation (1/3)

13

- In ‘classical’ process calculi (CCS, CSP, LOTOS...):
 - ▶ one often describes a finite set of concurrent actors
 - ▶ these actors can be (recursively) nested
 - ▶ the communication topology (i.e., gates) is fixed
 - ▶ well-adapted to hardware design, data transmission protocols

- In fact, ‘classical’ process calculi can do more:
 - ▶ dynamic creation/destruction of actors and channels
Example: $A ; \text{hide } G \text{ in } (B \mid [G] \mid C) ; D$
 - ▶ unbounded dynamic creation of actors
Example: $\text{process } P(N) := \text{if } N=0 \text{ then } Q \text{ else } (P(N-1) \mid \mid Q)$

(mixing LOTOS and LOTOS NT syntaxes)

Motivation (2/3)

14

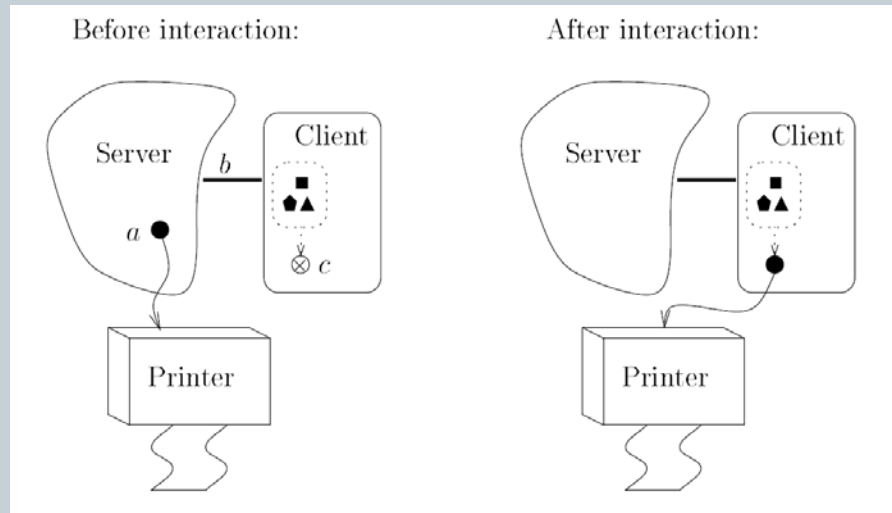
- ‘Mobile process calculi’ : a more radical approach
 - ▶ dynamically evolving networks
 - ▶ actors can be created/deleted dynamically
 - ▶ channels (communication links) also
 - ▶ actors can discover each other, and then communicate
 - ▶ often, they are put in relation by a third-party (‘trader’)

- Real-life examples:
 - ▶ plug-and-play devices on a network
 - ▶ mobile phones and base stations
 - ▶ object-oriented software

Motivation (3/3)

15

■ The printer discovery example (J. Parrow):



■ One approach to mobility: sending channels

- ▶ impossible in 'classical' process calculi, where offers sent or received on gates only contain data values (but not gates)
- ▶ sending processes is similar to sending channels

The pi-calculus

16

- Proposed by R. Milner, J. Parrow, D. Walker in the early 90s (see References)
- Defined as an extension of CCS
- Two main changes:
 - ▶ channels can be sent on channels
 - ▶ the restriction operator of CCS is technically modified
- A very influential model in academia:
 - ▶ many variants
 - ▶ some tools, such as the Mobility Workbench
<http://www.it.uu.se/research/group/mobility/mwb>
 - ▶ some applications – basis for defining BPEL
 - ▶ see <http://move.to/mobility>

Syntax

17

Prefixes	$\alpha ::=$	$\bar{a}x$	Output (noted a !x in LOTOS)
		$a(x)$	Input (noted a ?x in LOTOS)
		τ	Silent
			also written 'a<x>
Agents	$P ::=$	0	Nil
		$\alpha . P$	Prefix
		$P + P$	Sum
		$P \mid P$	Parallel
		if $x = y$ then P	Match
		if $x \neq y$ then P	Mismatch
		$(\nu x)P$	Restriction
		$A(y_1, \dots, y_n)$	Identifier
added later			
initially noted $P \setminus a$ as in CCS			
Definitions	$A(x_1, \dots, x_n) \stackrel{\text{def}}{=} P \quad (\text{where } i \neq j \Rightarrow x_i \neq x_j)$		

Static semantics

18

- A single ‘type’ of data, merging values and channels
- Variables are defined (‘bound’) only at 3 places:
 - ▶ $x(y).P$: variable y contains the data received on x
 y is visible only in P
 - ▶ $(\nu y)P$: a new channel is created and assigned to variable y
 y is visible only in P , but P may send y to other agents
(this is called ‘scope extrusion’ – tricky rules)
 - ▶ $A(x_1, \dots, x_n) = P$: parameters $x_1, \dots, x_n$ are visible in P
- $bn(P) :=$ bound variables defined in P : $x(y)$ or (νy)
- $fn(P) :=$ all other variables used in P (*free variables*)

Dynamic semantics

19

$$\begin{array}{ll}
 \text{TAU} \quad \tau.P \xrightarrow{\tau} P & \text{OUT} \quad \bar{x}y.P \xrightarrow{\bar{x}y} P \quad \text{IN} \quad x(y).P \xrightarrow{xz} P\{z/y\} \\
 \text{SUM} \quad \frac{P_1 \xrightarrow{\alpha} P'_1}{P_1 + P_2 \xrightarrow{\alpha} P'_1} & \text{PAR} \quad \frac{P_1 \xrightarrow{\alpha} P'_1}{P_1 | P_2 \xrightarrow{\alpha} P'_1 | P_2} \text{ if } bn(\alpha) \cap fn(P_2) = \emptyset \\
 \text{COM} \quad \frac{P_1 \xrightarrow{\bar{x}y} P'_1 \quad P_2 \xrightarrow{xy} P'_2}{P_1 | P_2 \xrightarrow{\tau} P'_1 | P'_2} & \text{CLOSE} \quad \frac{P_1 \xrightarrow{\bar{x}(y)} P'_1 \quad P_2 \xrightarrow{xy} P'_2}{P_1 | P_2 \xrightarrow{\tau} (\nu y)(P'_1 | P'_2)} \text{ if } y \notin fn(P_2) \\
 \text{RES} \quad \frac{P \xrightarrow{\alpha} P'}{(\nu x)P \xrightarrow{\alpha} (\nu x)P'} \text{ if } x \notin n(\alpha) & \text{OPEN} \quad \frac{P \xrightarrow{\bar{x}y} P'}{(\nu y)P \xrightarrow{\bar{x}(z)} P'\{z/y\}} \text{ if } x \neq y, z \notin fn((\nu y)P') \\
 \text{MATCH} \quad \frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'} & \text{MISMATCH} \quad \frac{P \xrightarrow{\alpha} P'}{[x \neq y]P \xrightarrow{\alpha} P'} \text{ if } x \neq y \\
 \text{IDE} \quad \frac{P\{y_1/x_1, \dots, y_{r(A)}/x_{r(A)}\} \xrightarrow{\alpha} P'}{A(y_1, \dots, y_{r(A)}) \xrightarrow{\alpha} P'} \text{ if } A(x_1, \dots, x_{r(A)}) \stackrel{\text{def}}{=} P
 \end{array}$$

Example

20

Taken from Mateescu-Salaün IFM 2010 paper
(see references)

$$\begin{aligned} Main &= (\nu req, a, b, c) (Client(req, a, b, c) \mid Dispatcher(req) \mid \\ &\quad Server(a) \mid Server(b) \mid Server(c)) \\ Client(req, a, b, c) &= (\nu x) (\overline{request} a. \overline{req} \langle a, x \rangle. ClientAux(req, a, a, b, c, x)) + \\ &\quad (\nu x) (\overline{request} b. \overline{req} \langle b, x \rangle. ClientAux(req, b, a, b, c, x)) + \\ &\quad (\nu x) (\overline{request} c. \overline{req} \langle c, x \rangle. ClientAux(req, c, a, b, c, x)) \\ ClientAux(req, k, a, b, c, x) &= x(info). (\overline{x} purchase. \overline{purchase} k. 0 + \\ &\quad \overline{x} refuse. \overline{refuse} k. Client(req, a, b, c)) \\ Dispatcher(req) &= req(k, x). \overline{k} x. Dispatcher(req) \\ Server(k) &= k(x). \overline{x} info. x(decision). Server(k) \end{aligned}$$

The PIC2LNT tool

21

PIC2LNT (1 / 2)

22

- A recent translator developed at INRIA Grenoble
- Input language: PIC
 - ▶ pi-calculus
 - ▶ with a machine-readable syntax (from Mobility Workbench)
 - ▶ extended with data values (= ‘applied pi-calculus’)
- Output: LOTOS NT program
- A script named ‘pic2bcg’ automates the translation
PIC → LOTOS NT → LOTOS → Petri nets → LTS

PIC2LNT (2/2)

23

■ The PIC language

- ▶ defined in the PIC2LNT manual page (see References)
- ▶ the data types and value expressions are those of LOTOS NT

■ The translation approach:

- ▶ most pi-calculus tools do symbolic proofs on the terms
- ▶ pic2Int works by state space exploration
(= explicit-state enumeration = reachability analysis)
- ▶ limitation: only works for finite-state models
- ▶ $\Rightarrow$ bounding channels, data types, ‘!’ operator
- ▶ BUT enables to study non-trivial mobile programs

A few notes

24

- Caution: 't' means τ (contrary to 'i' in LOTOS/NT)
- The restriction operator ν must be written '**new**'
- Emissions $\bar{x}$ have to be noted 'x'
- Emittted parameters must be bracked with < and > even when there is only a single parameter
- Received parameters must be bracked with (and) even when there is only a single parameter
- There are no channel declarations: beware of typos
 - ▶ exploit: at any place, you can easily insert a 'debug event

More notes

25

- In the LTS obtained, the labels carry extra offers
 - ▶ for instance: !FALSE or !TRUE
 - ▶ this is an artefact of the translation to LOTOS NT
 - ▶ (perhaps the pic2bcg script could remove them)
- The translation implements the creation of new channels by giving unique numbers
 - ▶ example: (new y) 'x<y> may generate a transition: X !Y(41)
 - ▶ don't worry if the counter is not increasing one by one
- Restriction hides the synchronizations ☹
 - ▶ one cannot observe them in the LTS (only τ -transitions can be seen)
 - ▶ add extra events if needed

Today's challenge

26

Your first pi-calculus program (1/2)

27

- Find the paper about PIC2LNT published at IFM 2010 (see References below)
- Copy-and-paste in a file named 'disp.pic' the pi-calculus example given page 11
- Convert it to machine-readable notations:
 - ▶ replace each ν symbol by the **new** keyword
 - ▶ replace emissions $\bar{x}$ with 'x
 - ▶ restore the < and > symbols around emissions of multiple channels; add them for emissions of single channels
 - ▶ same with (and) for receptions
 - ▶ finally, replace the 0 with nil (0 is not documented in the manual page, yet seems to be accepted)

Your first pi-calculus program (2/2)

28

- Perform the translation $\text{PIC} \rightarrow \text{LOTOS NT} \rightarrow \text{LOTOS} \rightarrow \text{Petri nets} \rightarrow \text{LTS}$ by typing:
 - ▶ `$ pic2bcg disp.pic`
 - ▶ if it does not compile properly, fix the mistakes
- Visualize the file 'disp.bcg' obtained
 - ▶ `$ bcg_edit disp.bcg`
- Compare it to the picture given page 11
- Minimize it using strong bisimulation to remove 'duplicated' parts of the LTS
 - ▶ `$ bcg_min disp.bcg`
 - ▶ `$ bcg_edit disp.bcg`
- Send your file 'disp.pic' and the PostScript file to Alexander (possibly with comments if you observe a difference with the picture of the paper)

References

29

Pi-calculus bibliography

30

- J. Parrow. *An introduction to the pi-calculus*. Chapter of the Handbook of Process Algebra, 2001.
<http://user.it.uu.se/~joachim/intro.ps>
Especially sections 1, 2.1, (2.1), 2.3, 4, and 6.
- U. Nestmann. *Welcome to the Jungle: A subjective guide to mobile process calculi*, 200x.
<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.89.6712>

Pi-calculus bibliography

31

- R. Milner, J. Parrow, D. Walker. *A calculus of mobile processes (parts I and II)*. Information and Computation, vol. 100, num. 1, 1992.
- R. Milner. *Elements of interaction: Turing award lecture*. <http://dl.acm.org/citation.cfm?id=151240>
- On-line resources: <http://move.to/mobility>

Tools for the pi-calculus

32

PIC2LNT translator, by R. Mateescu and G. Salaün, 2010-12.
In your [VM](#), directory [\\$HOME/Desktop/PIC2LNT](#)

- ▶ Reference documentation:
The PIC2LNT manual page
in your [VM](#), directory [\\$HOME/Desktop/PIC2LNT/man/pdf](#)
- ▶ If you want details on the translation:
R. Mateescu and G. Salaün. *Translating Pi-Calculus into LOTOS NT*. IFM 2010
in your [VM](#), directory [\\$HOME/Desktop/PIC2LNT/doc/pdf](#)
(caution: their version of LOTOS NT is highly simplified)