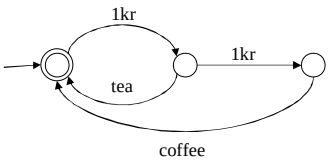
CCS: Processes and Equivalences

Reading: Peled 8.1, 8.2, 8.5

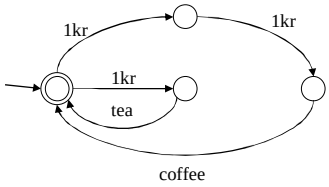
Mads Dam

Finite State Automata

- Coffee machine A_1:



- Coffee machine A_2:



- Are the two machines ”the same”?

CCS

Calculus of concurrent processes

Main issues:

- How to specify concurrent processes in an abstract way?
- Which are the basic relations between concurrency and non-determinism?
- Which basic methods of construction (= operators) are needed?
- When do two processes behave differently?
- When do they behave the same?
- Rules of calculation:
  - Replacing equals for equals
  - Substitutivity
- Specification and modelling issues

Process Equivalences

Sameness of behaviour = equivalence of states

Many process equivalences have been proposed (cf. Peled 8.5)

For instance: $q_1 \sim q_2$ iff

- q_1 and q_2 have the same paths, *or*
- q_1 and q_2 may always refuse the same interactions, *or*
- q_1 and q_2 pass the same tests, *or*
- q_1 and q_2 satisfy the same temporal formulas, *or*
- q_1 and q_2 have identical branching structure

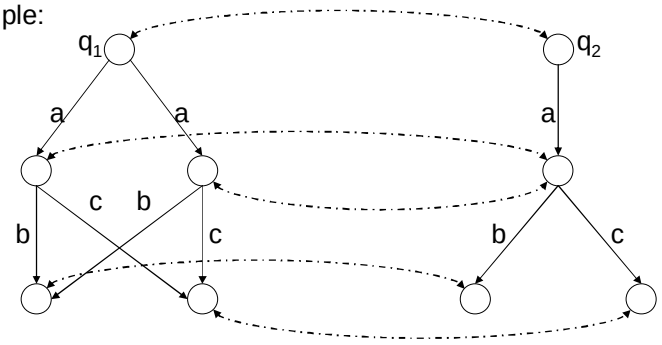
CCS: Focus on bisimulation equivalence

Bisimulation Equivalence

Intuition: $q_1 \sim q_2$ iff q_1 and q_2 have same branching structure

Idea: Find relation which will relate two states with the same transition structure, and make sure the relation is preserved

Example:



Strong Bisimulation Equivalence

Given: Labelled transition system $T = (Q,\Sigma,R)$

Looking for a relation $S \subseteq Q \times Q$ on states

S is a *strong bisimulation relation* if whenever q_1 S q_2 then:

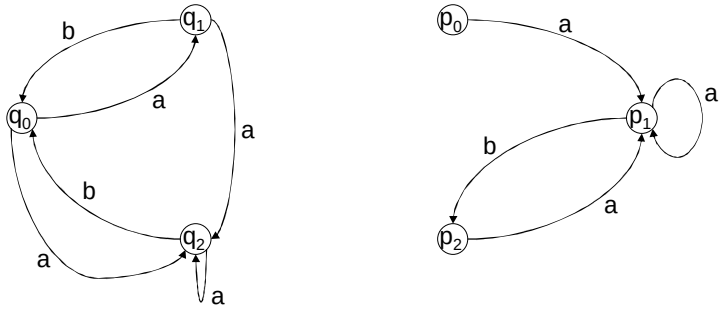
- $q_1 \rightarrow^{\alpha} q_1'$ implies $q_2 \rightarrow^{\alpha} q_2'$ for some q_2' such that q_1' S q_2'
- $q_2 \rightarrow^{\alpha} q_2'$ implies $q_1 \rightarrow^{\alpha} q_1'$ for some q_1' such that q_1' S q_2'

q_1 and q_2 are *strongly bisimilar* iff q_1 S q_2 for some strong bisimulation relation S

$q_1 \sim q_2$: q_1 and q_2 are strongly bisimilar

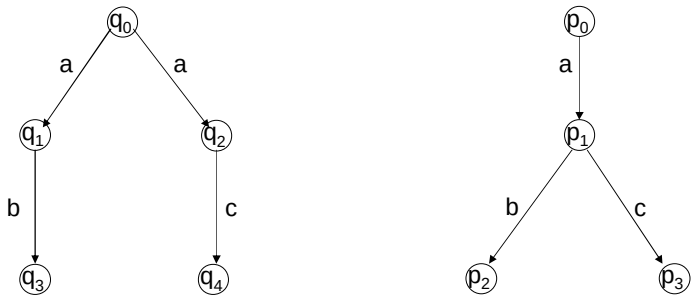
Peled uses $\equiv_{bis}$ for $\sim$

Example



Does $q_0 \sim p_0$ hold?

Example



Does $q_0 \sim p_0$ hold?

Weak Transitions

What to do about internal activity?

τ: Transition label for activity which is not externally visible

- $q \Rightarrow^{\varepsilon} q'$ iff $q = q_0 \rightarrow^{\tau} q_1 \rightarrow^{\tau} \ldots \rightarrow^{\tau} q_n = q'$, $n \geq 0$
- $q \Rightarrow^{\tau} q'$ iff $q \Rightarrow^{\varepsilon} q'$
- $q \Rightarrow^{\alpha} q'$ iff $q \Rightarrow^{\varepsilon} q_1 \rightarrow^{\alpha} q_2 \Rightarrow^{\varepsilon} q'$ ($\alpha \neq \tau$)

Beware that $\Rightarrow^{\tau} = \Rightarrow^{\varepsilon}$ (non-standard notation)

Observational equivalence, v.1.0: Bisimulation equivalence with $\Rightarrow$ in place of $\rightarrow$

Let $q_1 \approx' q_2$ iff $q_1 \sim q_2$ with $\Rightarrow^{\alpha}$ in place of $\rightarrow^{\alpha}$

Cumbersome definition: Too many transitions $q \Rightarrow^{\alpha} q'$ to check

Observational Equivalence

Let $S \subseteq Q \times Q$. The relation S is a *weak bisimulation relation* if whenever $q_1 \, S \, q_2$ then:

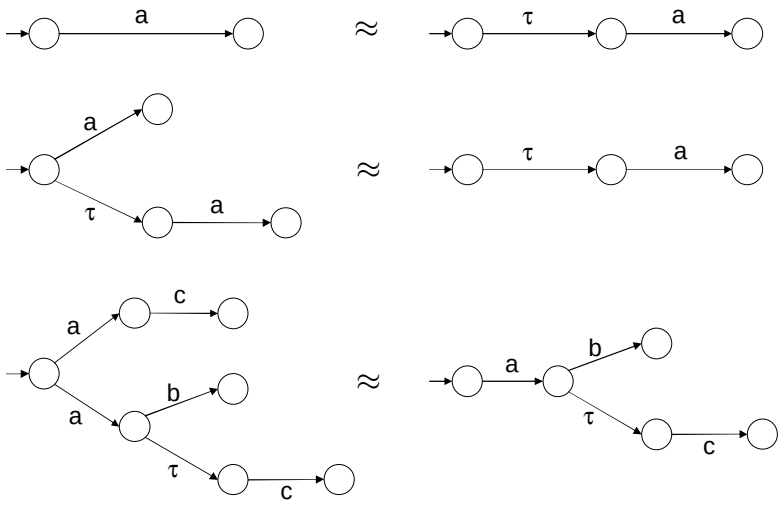
- $q_1 \rightarrow^{\alpha} q_1'$ implies $q_2 \Rightarrow^{\alpha} q_2'$ for some q_2' such that $q_1' \, S \, q_2'$
- $q_2 \rightarrow^{\alpha} q_2'$ implies $q_1 \Rightarrow^{\alpha} q_1'$ for some q_1' such that $q_1' \, S \, q_2'$

q_1 and q_2 are *observationally equivalent,* or *weakly bisimulation equivalent,* if $q_1 \, S \, q_2$ for some weak bisimulation relation S

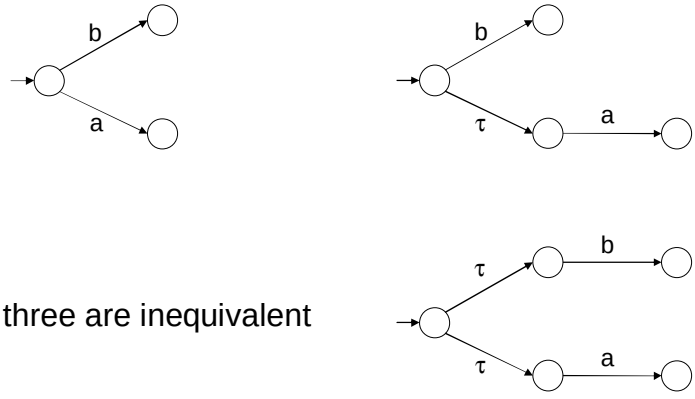
$q_1 \approx q_2$: q_1 and q_2 are observationally equivalent/weakly bisimilar

Exercise: Show that $\approx' = \approx$

Examples



Examples



All three are inequivalent

Calculus of Communicating Systems - CCS

Language for describing communicating transition systems

Behaviours as algebraic terms

Calculus: Centered on observational equivalence

Elegant mathematical treatment

Emphasis on process structure and modularity

Recent extensions to security and mobile systems

- CSP - Hoare: Communicating Sequential Processes (85)
- ACP - Bergstra and Klop: Algebra of Communicating Processes (85)
- CCS - Milner: Communication and Concurrency (89)
- Pi-calculus – Milner (99), Sangiorgi and Walker (01)
- SPI-calculus – Abadi and Gordon (99)
- Many recent successor for security and mobility (more in 2G1517)

CCS - Combinators

The idea: 7 elementary ways of producing or putting together labelled transition systems

Pure CCS:

- Turing complete – can express any Turing computable function

Value-passing CCS:

- Additional operators for value passing
- Definable
- Convenient for applications

Here only a taster

Actions

Names a,b,c,d,...

Co-names: $\overline{a},\overline{b},\overline{c},\overline{d},...$
- Sorry: Overbar not good in texpoint!
- $\overline{\overline{a}} = a$

In CCS, names and co-names synchronize

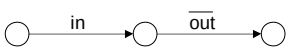
Labels l: Names ∪ co-names

$\alpha \in$ Actions = Σ = Labels ∪ $\{\tau\}$

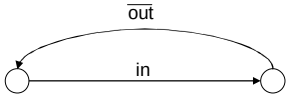
Define $\overline{\alpha}$ by:
- $\overline{l} = \overline{l}$, and
- $\overline{\tau} = \tau$

CCS Combinators, II

Nil	0	No transitions
Prefix	$\alpha.P$	$in.\overline{out}.0 \rightarrow^{in} out.0 \rightarrow^{\overline{out}} 0$



Definition	$A == P$	Buffer == $in.\overline{out}$.Buffer Buffer $\rightarrow^{in}$ out.Buffer $\rightarrow^{\overline{out}}$ Buffer



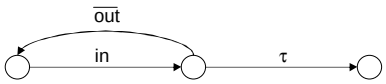
CCS Combinators, Choice

Choice $P + Q$ BadBuf == in.(τ.0 + $\overline{\text{out}}$.BadBuf)

BadBuf $\rightarrow^{\text{in}}$ τ.0 + $\overline{\text{out}}$.BadBuf

$\rightarrow^{\tau}$ 0 **or**

$\rightarrow^{\overline{\text{out}}}$ BadBuf



Obs: No priorities between τ's, a's or $\overline{a}$'s

CCS doesn't "know" which labels represent input, and which output

May use Σ notation: $\Sigma_{i\in\{1,2\}}\alpha_i.P_i = \alpha_1.P_1 + \alpha_2.P_2$

Example: Boolean Buffer

2-place Boolean Buffer

Buf^2: Empty 2-place buffer
Buf^2_0: 2-place buffer holding a 0
Buf^2_1: Do. holding a 1
Buf^2_{00}: Do. Holding 00
... etc. ...

$Buf^2 == in_0.Buf^2_0 + in_1.Buf^2_1$
$Buf^2_0 == out_0.Buf^2 +$
$\quad in_0.Buf^2_{00} + in_1.Buf^2_{01}$
$Buf^2_1 == ...$
$Buf^2_{00} == out_0.Buf^2_0$
$Buf^2_{01} == out_0.Buf^2_1$
$Buf^2_{10} == ...$
$Buf^2_{11} == ...$

Example: Scheduler

a_i: start $task_i$

b_i: stop $task_i$

Requirements:

1. $a_1,...,a_n$ to occur cyclically
2. a_i/b_i to occur alternately beginning with a_i
3. Any a_i/b_i to be schedulable at any time, provided 1 and 2 not violated

Let $X \subseteq \{1,...,n\}$

$Sched_{i,X}$:

- i to be scheduled
- X pending completion

$Scheduler == Sched_{1,\emptyset}$

$Sched_{i,X}$

$== \Sigma_{j \in X} b_j.Sched_{i,X-\{j\}}$, if $i \in X$

$== \Sigma_{j \in X} b_j.Sched_{i,X-\{j\}} + a_i.Sched_{i+1,X \cup \{i\}}$, if $i \notin X$

Example: Counter

Basic example of infinite-state system

$Count == Count_0$

$Count_0 == zero.Count_0 + inc.Count_1$

$Count_{i+1} == inc.Count_{i+2} + dec.Count_i$

Can do stacks and queues equally easy – try it!

CCS Combinators, Composition

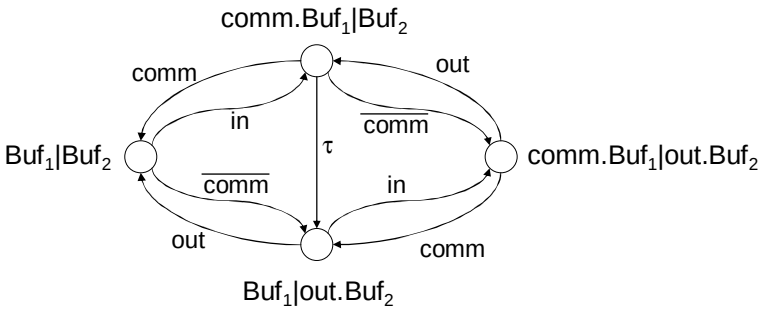
Composition $P \mid Q$

$Buf_1 == in.comm.Buf_1$
$Buf_2 == \overline{comm}.out.Buf_2$
$Buf_1 \mid Buf_2$
$\rightarrow^{in} comm.Buf_1 \mid Buf_2$
$\rightarrow^{\tau} Buf_1 \mid out.Buf_2$
$\rightarrow^{out} Buf_1 \mid Buf_2$

But also, for instance:
$Buf_1 \mid Buf_2$
$\rightarrow^{\overline{comm}} Buf_1 \mid out.Buf_2$
$\rightarrow^{out} Buf_1 \mid Buf_2$

Composition, Example

$Buf_1 == in.comm.Buf_1$
$Buf_2 == \overline{comm}.out.Buf_2$
$Buf_1 \mid Buf_2$:



CCS Combinators, Restriction

Restriction P\L

$Buf_1 == in.comm.Buf_1$

$Buf_2 == \overline{comm}.out.Buf_2$

$(Buf_1 \mid Buf_2)\backslash\{comm\}$

$\rightarrow^{in} comm.Buf_1 \mid Buf_2$

$\rightarrow^{\tau} Buf_1 \mid out.Buf_2$

$\rightarrow^{out} Buf_1 \mid Buf_2$

But *not*:

$(Buf_1 \mid Buf_2)\backslash\{comm\}$

$\rightarrow^{\overline{comm}} Buf_1 \mid out.Buf_2$

$\rightarrow^{out} Buf_1 \mid Buf_2$

CCS Combinators, Relabelling

Relabelling *P*[f]

$Buf == in.\overline{out}.Buf_1$

$Buf_1 == Buf[comm/out]$

$= in.\overline{comm}.Buf_1$

$Buf_2 == Buf[comm/in]$

$= comm.out.Buf_2$

Relabelling function f must preserve complements:

$f(\bar{a}) = \overline{f(a)}$

And τ:

$f(\tau) = \tau$

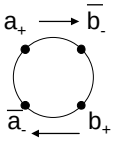
Relabelling function often given by name substitution as above

Example: 2-way Buffers

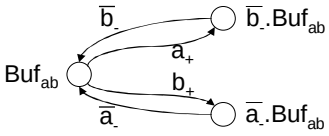
1-place 2-way buffer:

$Buf_{ab} == a_+.\overline{b}_-.Buf_{ab} + b_+.\overline{a}_-.Buf_{ab}$

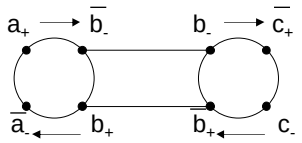
Flow graph:



LTS:



$Buf_{bc} ==$
$Buf_{ab}[c_+/b_+,c_-/b_-,b_-/a_+,b_+/a_-]$
(Obs: Simultaneous substitution!)
$Sys = (Buf_{ab} \mid Buf_{bc})\backslash\{b_+,b_-\}$
Intention:



What went wrong?

Transition Semantics

To apply observational equivalence need a formalised semantics

Each CCS expression -> state in LTS derived from that expression

Compositionality: Construction of LTS follows expression syntax

Inference rules:

$$\frac{P_1 \rightarrow^{\alpha} P_2}{P_1 \mid Q \rightarrow^{\alpha} P_2 \mid Q}$$

Meaning: For all P_1, P_2, Q, α, if there is an α transition from P_1 to P_2 then there is an α transition from $P_1 \mid Q$ to $P_2 \mid Q$

CCS Transition Rules

(no rule for 0!)

Prefix $\dfrac{-}{\alpha.P \rightarrow^{\alpha} P}$

Def $\dfrac{P \rightarrow^{\alpha} Q}{A \rightarrow^{\alpha} Q}(A == P)$

Choice$_L$ $\dfrac{P \rightarrow^{\alpha} P'}{P+Q \rightarrow^{\alpha} P'}$

Choice$_L$ $\dfrac{Q \rightarrow^{\alpha} Q'}{P+Q \rightarrow^{\alpha} Q'}$

Com$_L$ $\dfrac{P \rightarrow^{\alpha} P'}{P|Q \rightarrow^{\alpha} P'|Q}$

Com$_R$ $\dfrac{Q \rightarrow^{\alpha} Q'}{P|Q \rightarrow^{\alpha} P|Q'}$

Com $\dfrac{P \rightarrow^{l} P' \quad Q \rightarrow^{\bar{l}} Q'}{P|Q \rightarrow^{\tau} P'|Q'}$

Restr $\dfrac{P \rightarrow^{\alpha} P'}{P \setminus L \rightarrow^{\alpha} P' \setminus L}$ $(\alpha, \bar{\alpha} \notin L)$

Rel $\dfrac{P \rightarrow^{\alpha} P'}{P[f] \rightarrow^{f(\alpha)} P'[f]}$

CCS Transition Rules, II

Closure assumption: $\rightarrow^{\alpha}$ is least relation closed under the set of rules

Example derivation:

$Buf_1 == in.\overline{comm}.Buf_1$

$Buf_2 == comm.\overline{out}.Buf_2$

$(Buf_1 \mid Buf_2) \setminus \{comm\}$

$\rightarrow^{in} \overline{comm}.Buf_1 \mid Buf_2$

$\rightarrow^{\tau} Buf_1 \mid \overline{out}.Buf_2$

$\rightarrow^{\overline{out}} Buf_1 \mid Buf_2$

Example: Semaphores

Semaphore:

Unary semaphore: 

$S^1 == p.S^1_1$

$S^1_1 == v.S^1$

Binary semaphore:

$S^2 == p.S^2_1$

$S^2_1 == p.S^2_2 + v.S^2$

$S^2_2 == v.S^2_1$

Result:

$S^1 \mid S^1 \sim S^2$

Proof: Show that
{($S^1 \mid S^1, S^2$),
($S^1_1 \mid S^1, S^2_1$),
($S^1 \mid S^1_1, S^2_1$),
($S^1_1 \mid S^1_1, S^2_2$)}
is a strong bisimulation relation

Example: Simple Protocol

$Spec == in.\overline{out}.Spec$

$Sender == in.Transmit$
$Transmit == \overline{transmit}.WaitAck$
$WaitAck == ack_+.Sender + ack_-.Transmit$

$Receiver == transmit.Analyze$
$Analyze == \tau.\overline{out}.\overline{ack_+}.Receiver + \tau.\overline{ack_-}.Receiver$

$Protocol == (Sender \mid Receiver) \setminus \{transmit, ack_+, ack_-\}$

Exercise: Prove $Spec \approx Protocol$

Example: Jobshop

i_E: input of easy job
i_N: input of neutral job
i_D: input of difficult job
O: output of finished product

$A == i_E.A' + i_N.A' + i_D.A'$
$A' == \overline{o}.A$

Spec = A | A

Hammer: H == gh.ph.H
Mallet: M == gm.pm.M
Jobber:
$J == \Sigma_{x\in\{E,N,D\}} i_x.J_x$
$J_E == o.J$
$J_N == \overline{gh}.\overline{ph}.J_E + \overline{gm}.\overline{pm}.J_E$
$J_D == \overline{gh}.\overline{ph}.J_E$
Jobshop ==
$(J \mid J \mid H \mid M)\setminus\{gh,ph,gm,pm\}$

Theorem:
Spec ≈ Jobshop

Exercise: Prove this.

Proving Equivalences

The bisimulation proof method:

To establish P ≈ Q:
1. Identify a relation S such that P S Q
2. Prove that S is a weak bisimulation relation

This is the canonical method

There are other methods for process verification:
- Equational reasoning
- Temporal logic specification/proof/model checking