

<http://didawiki.di.unipi.it/doku.php/magistraleinformatica/psc/>

PSC 2020/21 (375AA, 9CFU)

Principles for Software Composition

Roberto Bruni

<http://www.di.unipi.it/~bruni/>

19 - Hennessy-Milner Logic

CCS syntax

p, q	$::=$	nil	inactive process
		x	process variable (for recursion)
		$\mu.p$	action prefix
		$p \setminus \alpha$	restricted channel
		$p[\phi]$	channel relabelling
		$p + q$	nondeterministic choice (sum)
		$p q$	parallel composition
		rec $x. p$	recursion

(operators are listed in order of precedence)

CCS op. semantics

$$\begin{array}{c}
 \text{Act)} \frac{}{\mu.p \xrightarrow{\mu} p} \quad \text{Res)} \frac{p \xrightarrow{\mu} q \quad \mu \notin \{\alpha, \bar{\alpha}\}}{p \setminus \alpha \xrightarrow{\mu} q \setminus \alpha} \quad \text{Rel)} \frac{p \xrightarrow{\mu} q}{p[\phi] \xrightarrow{\phi(\mu)} q[\phi]} \\
 \\
 \text{SumL)} \frac{p_1 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q} \quad \text{SumR)} \frac{p_2 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q} \\
 \\
 \text{ParL)} \frac{p_1 \xrightarrow{\mu} q_1}{p_1 | p_2 \xrightarrow{\mu} q_1 | p_2} \quad \text{Com)} \frac{p_1 \xrightarrow{\lambda} q_1 \quad p_2 \xrightarrow{\bar{\lambda}} q_2}{p_1 | p_2 \xrightarrow{\tau} q_1 | q_2} \quad \text{ParR)} \frac{p_2 \xrightarrow{\mu} q_2}{p_1 | p_2 \xrightarrow{\mu} p_1 | q_2} \\
 \\
 \text{Rec)} \frac{p[\mathbf{rec} \ x. \ p / x] \xrightarrow{\mu} q}{\mathbf{rec} \ x. \ p \xrightarrow{\mu} q}
 \end{array}$$

HML

Hennessy-Milner Logic

Logical equivalence

Let us take another approach to equivalence

we define some logic (set of formulas)

a process may or may not satisfy a formula

two processes are (logically) equivalent

when they satisfy exactly the same formulas

formulas must describe behavioural properties of processes

the ability / inability to perform transitions

(modal logic: possibly, necessarily)

then, we can compose formulas with usual operators

Hennessy-Milner Logic

We present the core operators

multi-modal:

modal operators are parameterised by actions

no negation:

the converse of a formula can also be written as a formula

no recursion:

each formula express properties about finite steps ahead

denotational semantics of a formula (postponed):

set of processes that satisfy the formula

HML: syntax

F, G	$::=$	tt	true	
		ff	false	
		$\bigwedge_{i \in I} F_i$	conjunction	
		$\bigvee_{i \in I} F_i$	disjunction	
		$\diamond_{\mu} F$	diamond operator	$\langle \mu \rangle F$
		$\square_{\mu} F$	box operator	$[\mu] F$

\mathcal{L} set of all formulas

HML: semantics

$p \models F$ reads “ p satisfies F ”

defined inductively on the structure of the formula

$p \models \text{tt}$ any process satisfies true
(no process satisfies false)

$p \models \bigwedge_{i \in I} F_i$ iff $\forall i \in I. p \models F_i$ p satisfies all F_i

$p \models \bigvee_{i \in I} F_i$ iff $\exists i \in I. p \models F_i$ p satisfies one of the F_i

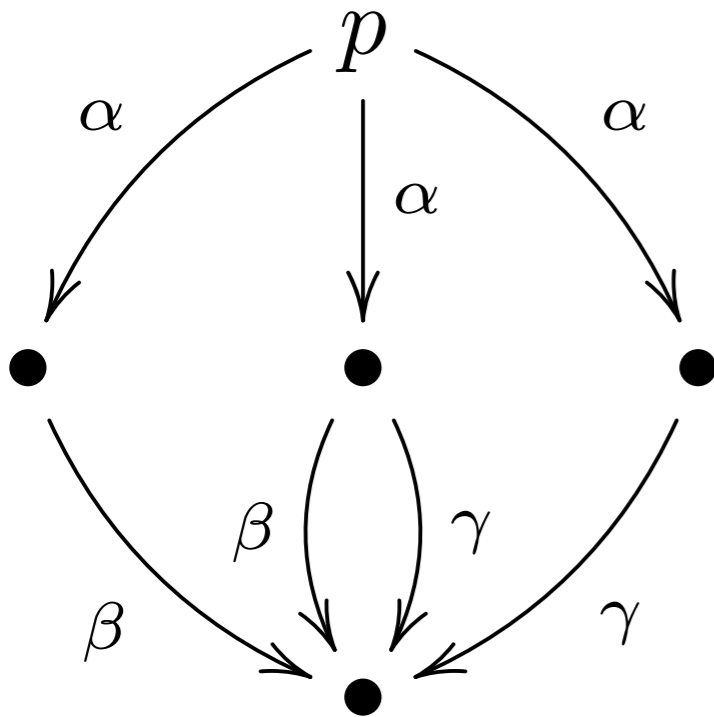
$p \models \diamond_{\mu} F$ iff $\exists p'. p \xrightarrow{\mu} p' \wedge p' \models F$ p can make one μ -step and then satisfy F

$p \models \square_{\mu} F$ iff $\forall p'. p \xrightarrow{\mu} p' \Rightarrow p' \models F$ F is satisfied after any μ -step of p

Examples

- $\diamond_{\alpha} \mathbf{tt}$ satisfied by any process that can make an α -step
- $\square_{\beta} \mathbf{ff}$ satisfied by any process that cannot make a β -step
- $\diamond_{\alpha} \mathbf{ff}$ same as \mathbf{ff}
 - if a process cannot do α the modality is missed
 - if a process can do α its continuation cannot satisfy \mathbf{ff}
- $\square_{\beta} \mathbf{tt}$ same as \mathbf{tt}
 - if a process cannot do β the modality holds trivially
 - if a process does β its continuation will satisfy \mathbf{tt}
- $\diamond_{\alpha} (\diamond_{\beta} \mathbf{tt} \wedge \square_{\gamma} \mathbf{ff})$ satisfied by any process the can do α and reach a process that can do β but not γ

Examples



$$p \stackrel{?}{\models} \diamond_{\alpha} \mathbf{tt}$$



$$p \stackrel{?}{\models} \square_{\alpha} \diamond_{\beta} \mathbf{tt}$$



$$p \stackrel{?}{\models} \diamond_{\alpha} \square_{\beta} \mathbf{ff} \wedge \diamond_{\alpha} \square_{\gamma} \mathbf{ff}$$



$$p \stackrel{?}{\models} \square_{\alpha} (\diamond_{\beta} \mathbf{tt} \vee \diamond_{\gamma} \mathbf{tt})$$



$$p \stackrel{?}{\models} \square_{\alpha} (\diamond_{\beta} \mathbf{tt} \wedge \diamond_{\gamma} \mathbf{tt})$$



$$p \stackrel{?}{\models} \diamond_{\alpha} (\diamond_{\beta} \mathbf{tt} \wedge \diamond_{\gamma} \mathbf{tt})$$



Negation

not present in the syntax, but not needed

any formula F has a *converse* formula F^c such that

$$\forall p. p \models F \quad \text{iff} \quad p \not\models F^c$$

F^c can be defined by structural induction

$$\mathbf{tt}^c \triangleq \mathbf{ff}$$

$$\mathbf{ff}^c \triangleq \mathbf{tt}$$

$$\left(\bigwedge_{i \in I} F_i \right)^c \triangleq \bigvee_{i \in I} F_i^c$$

$$\left(\bigvee_{i \in I} F_i \right)^c \triangleq \bigwedge_{i \in I} F_i^c$$

$$\left(\diamond_{\mu} F \right)^c \triangleq \square_{\mu} F^c$$

$$\left(\square_{\mu} F \right)^c \triangleq \diamond_{\mu} F^c$$

example

$$\left(\diamond_{\alpha} \mathbf{tt} \right)^c = \square_{\alpha} \mathbf{tt}^c = \square_{\alpha} \mathbf{ff} \quad (\text{can do } \alpha)^c = \text{cannot do } \alpha$$

Extended syntax

$$A = \{\mu_1, \dots, \mu_n\}$$

$$\begin{aligned} \diamond_A F &\triangleq \diamond_{\mu_1} F \vee \dots \vee \diamond_{\mu_n} F & \square_A F &\triangleq \square_{\mu_1} F \wedge \dots \wedge \square_{\mu_n} F \\ &= \bigvee_{i \in [1, n]} \diamond_{\mu_i} F & &= \bigwedge_{i \in [1, n]} \square_{\mu_i} F \end{aligned}$$

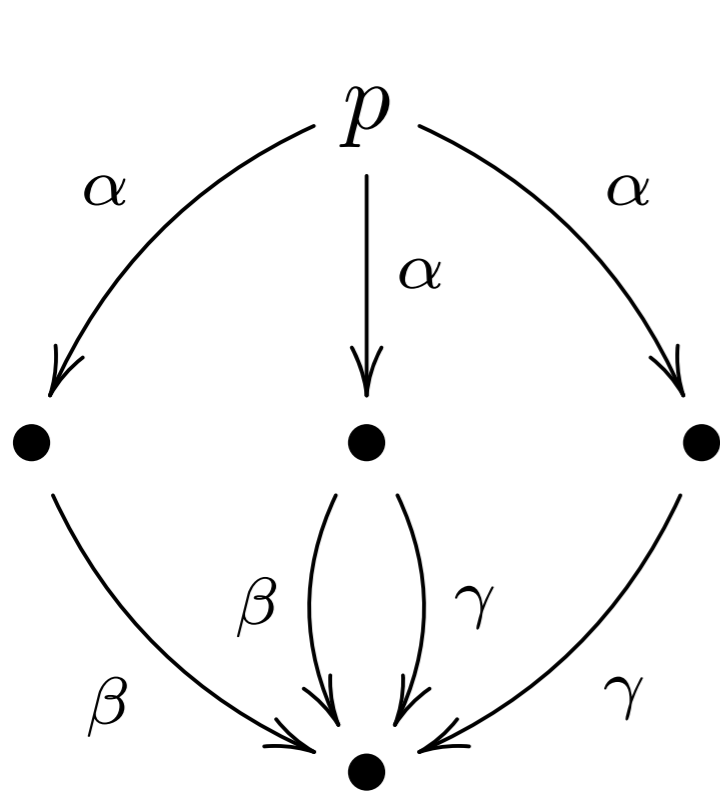
$$\diamond_{\emptyset} F \triangleq \mathbf{ff}$$

$$\square_{\emptyset} F \triangleq \mathbf{tt}$$

HML: logical equivalence

two processes are equivalent iff they satisfy the same formulas

$$p \equiv_{\text{HM}} q \quad \text{iff} \quad \forall F \in \mathcal{L}. (p \models F \Leftrightarrow q \models F)$$



$$p \models F$$

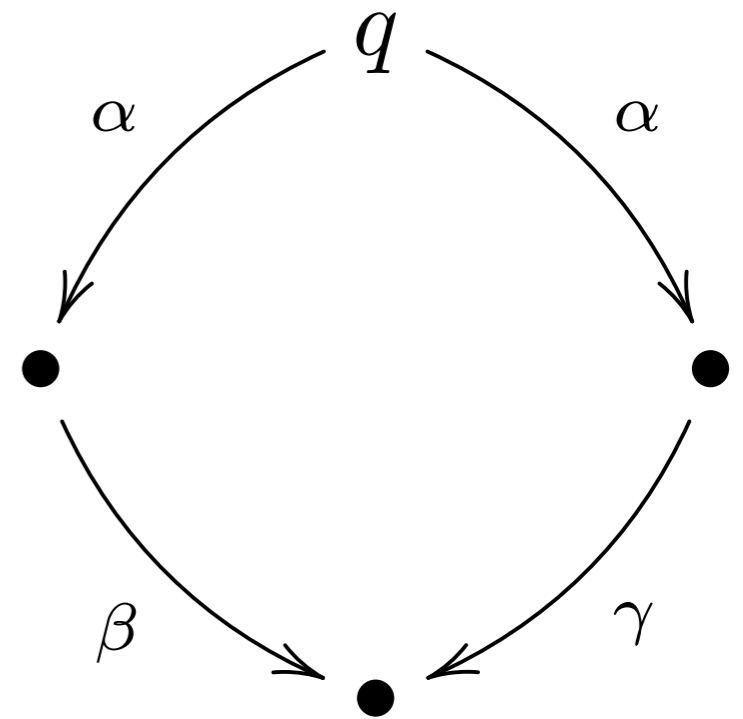
$$p \not\models F^c$$

$$p \stackrel{?}{\equiv}_{\text{HM}} q$$



$$F \triangleq \diamond_{\alpha} (\diamond_{\beta} \mathbf{tt} \wedge \diamond_{\gamma} \mathbf{tt})$$

$$F^c \triangleq \square_{\alpha} (\square_{\beta} \mathbf{ff} \vee \square_{\gamma} \mathbf{ff})$$



$$q \not\models F$$

$$q \models F^c$$

Strong bis as logic equiv

TH. for any finitely branching processes p, q

$$p \simeq q \quad \text{iff} \quad p \equiv_{\text{HM}} q$$

(proof omitted)

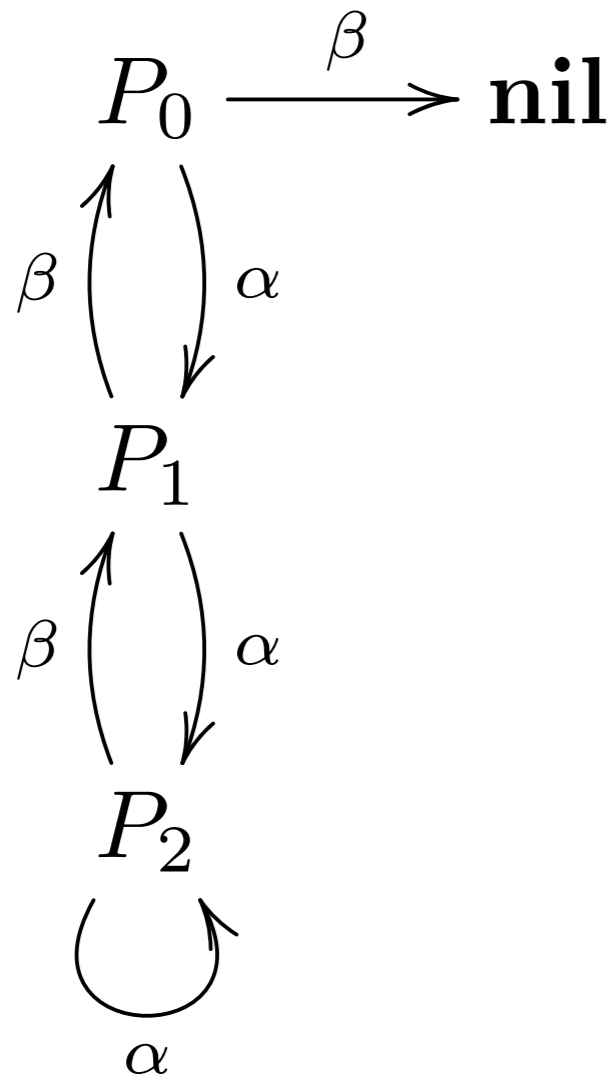
consequences:

to show that two processes are strong bisimilar:
exhibit a strong bisimulation relation that relates them

to show that two processes are not strong bisimilar:
exhibit a HML formula that distinguishes between them

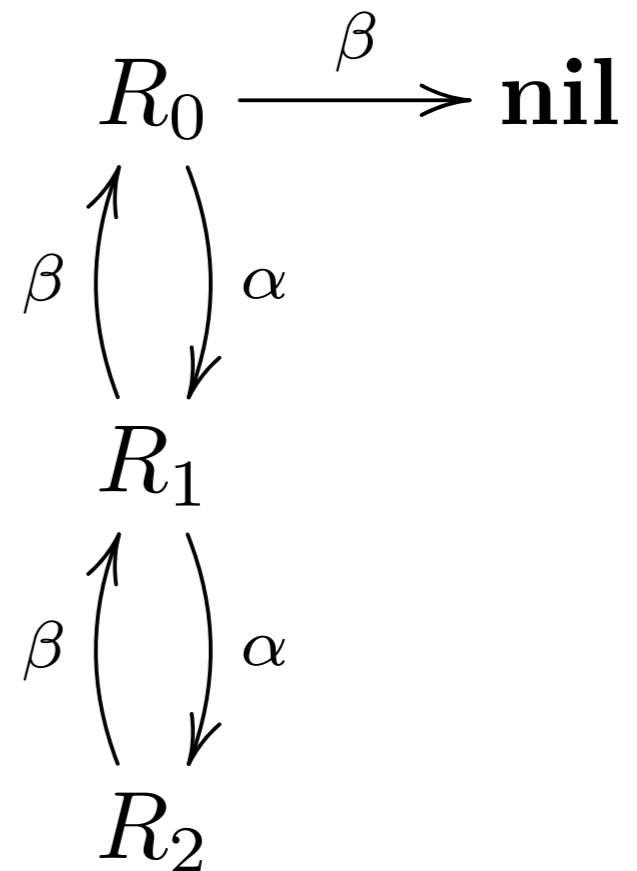
Exercise

find a HML formula that distinguishes the two processes



$$P_0 \models F$$

$$P_0 \neq R_0$$



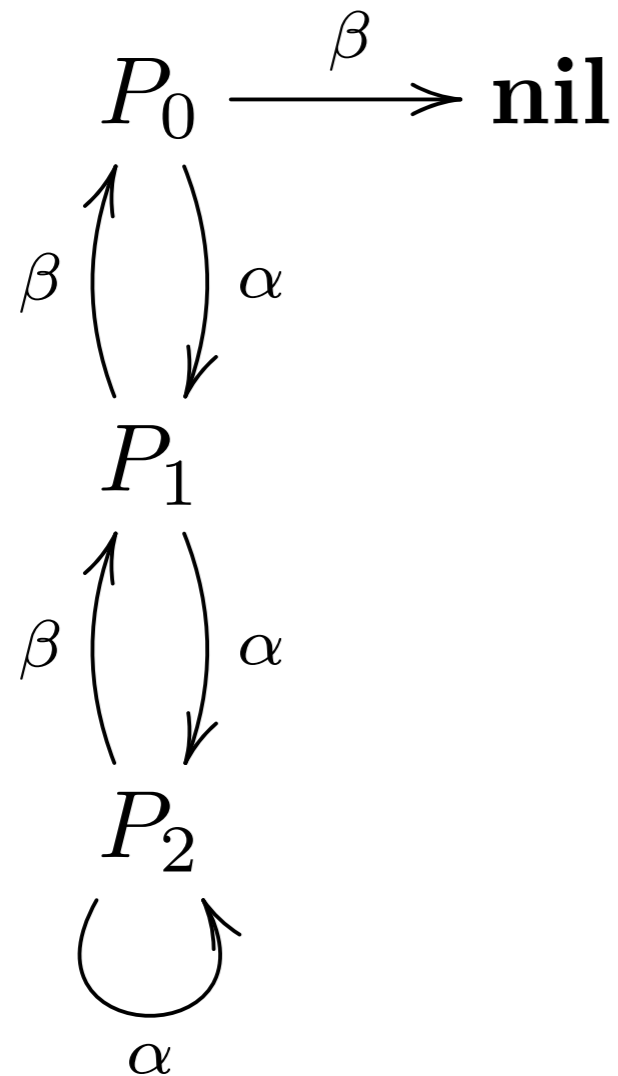
$$R_0 \not\models F$$

$$F \triangleq \diamond_{\alpha} \diamond_{\alpha} \diamond_{\alpha} \mathbf{tt}$$

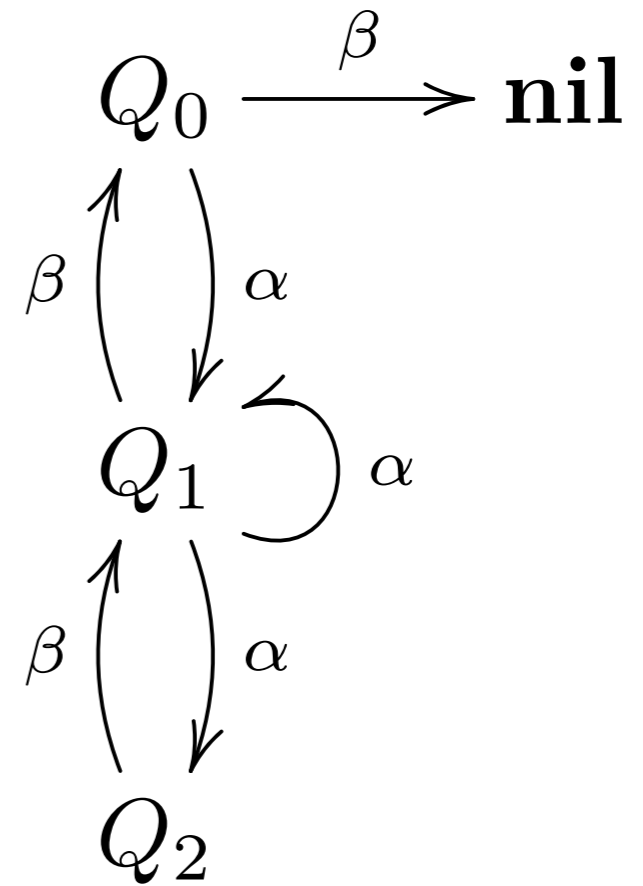


Exercise

find a HML formula that distinguishes the two processes



$$P_0 \neq Q_0$$



$$P_0 \models F$$

$$F \triangleq \diamond_{\alpha} \square_{\alpha} \diamond_{\alpha} \mathbf{tt}$$

$$Q_0 \not\models F$$