



PSC 2024/25 (375AA, 9CFU)

Principles for Software Composition

Roberto Bruni

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

[http://didawiki.di.unipi.it/doku.php/
magistraleinformatica/psc/start](http://didawiki.di.unipi.it/doku.php/magistraleinformatica/psc/start)

18c - bisimilarity as a fixpoint

CCS syntax

p, q	$::=$	nil	inactive process
		x	process variable (for recursion)
		$\mu.p$	action prefix
		$p \backslash \alpha$	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

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

Strong bisimilarity

$\mathcal{P}$ set of processes

$\mathbf{R} \subseteq \mathcal{P} \times \mathcal{P}$ a binary relation

$\mathbf{R}$ is a strong bisimulation if

$$\forall p, q. (p, q) \in \mathbf{R} \Rightarrow \left\{ \begin{array}{l} \forall \mu, p'. p \xrightarrow{\mu} p' \Rightarrow \exists q'. q \xrightarrow{\mu} q' \wedge p' \mathbf{R} q' \\ \wedge \\ \forall \mu, q'. q \xrightarrow{\mu} q' \Rightarrow \exists p'. p \xrightarrow{\mu} p' \wedge p' \mathbf{R} q' \end{array} \right.$$

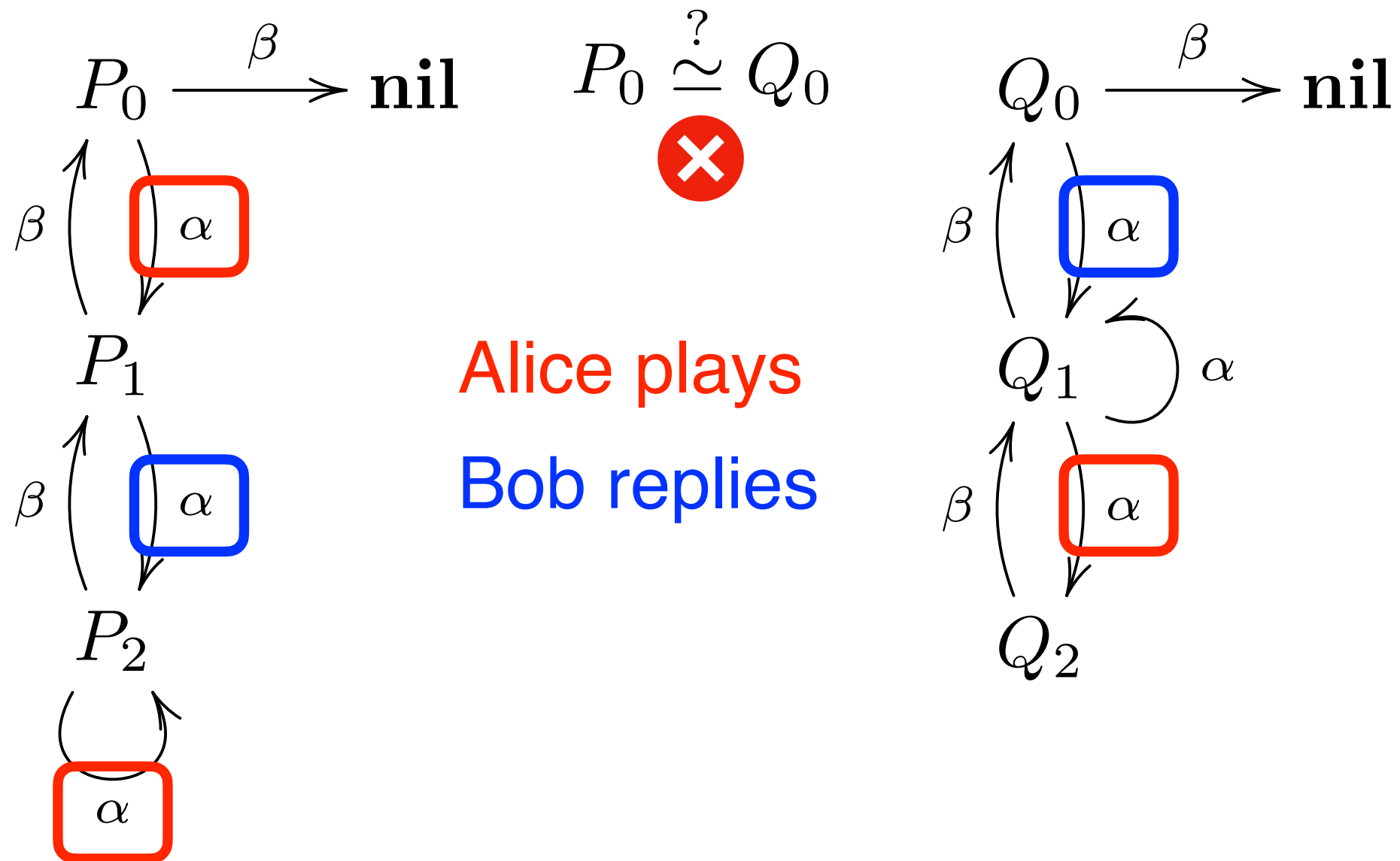
Alice plays
Bob replies

strong bisimilarity $\simeq \triangleq \bigcup_{\mathbf{R} \text{ s.b.}} \mathbf{R}$ is an equivalence

is a strong bisimulation

$$\forall p, q. p \simeq q \Leftrightarrow \left\{ \begin{array}{l} \forall \mu, p'. p \xrightarrow{\mu} p' \Rightarrow \exists q'. q \xrightarrow{\mu} q' \wedge p' \simeq q' \\ \wedge \\ \forall \mu, q'. q \xrightarrow{\mu} q' \Rightarrow \exists p'. p \xrightarrow{\mu} p' \wedge p' \simeq q' \end{array} \right.$$

Bisimulation game



CCS

Bisimilarity as a fixpoint

Strong bis as fix

$$\forall p, q. (p, q) \in \mathbf{R} \Rightarrow \left\{ \begin{array}{l} \forall \mu, p'. p \xrightarrow{\mu} p' \Rightarrow \exists q'. q \xrightarrow{\mu} q' \wedge p' \mathbf{R} q' \\ \wedge \\ \forall \mu, q'. q \xrightarrow{\mu} q' \Rightarrow \exists p'. p \xrightarrow{\mu} p' \wedge p' \mathbf{R} q' \end{array} \right.$$

$\Phi : \wp(\mathcal{P} \times \mathcal{P}) \rightarrow \wp(\mathcal{P} \times \mathcal{P})$ maps relations to relations

$$\Phi(\mathbf{R}) \triangleq \left\{ (p, q) \left| \begin{array}{l} \forall \mu, p'. p \xrightarrow{\mu} p' \Rightarrow \exists q'. q \xrightarrow{\mu} q' \wedge p' \mathbf{R} q' \\ \wedge \\ \forall \mu, q'. q \xrightarrow{\mu} q' \Rightarrow \exists p'. p \xrightarrow{\mu} p' \wedge p' \mathbf{R} q' \end{array} \right. \right\}$$

$$\mathbf{R} \subseteq \Phi(\mathbf{R})$$

a strong bisimulation

Strong bis as fix

$$\forall p, q. p \simeq q \Leftrightarrow \left\{ \begin{array}{l} \forall \mu, p'. p \xrightarrow{\mu} p' \Rightarrow \exists q'. q \xrightarrow{\mu} q' \wedge p' \simeq q' \\ \wedge \\ \forall \mu, q'. q \xrightarrow{\mu} q' \Rightarrow \exists p'. p \xrightarrow{\mu} p' \wedge p' \simeq q' \end{array} \right.$$

$\Phi : \wp(\mathcal{P} \times \mathcal{P}) \rightarrow \wp(\mathcal{P} \times \mathcal{P})$ maps relations to relations

$$\Phi(\mathbf{R}) \triangleq \left\{ (p, q) \left| \begin{array}{l} \forall \mu, p'. p \xrightarrow{\mu} p' \Rightarrow \exists q'. q \xrightarrow{\mu} q' \wedge p' \mathbf{R} q' \\ \wedge \\ \forall \mu, q'. q \xrightarrow{\mu} q' \Rightarrow \exists p'. p \xrightarrow{\mu} p' \wedge p' \mathbf{R} q' \end{array} \right. \right\}$$

$$\simeq = \Phi(\simeq)$$

strong bisimilarity is a fixpoint

Fixpoint: which CPO?

Can we reuse Kleene's fix point theorem?

we want to find the **coarsest** relation,
not the **least** relation

Idea: reverse the usual order (inclusion)!

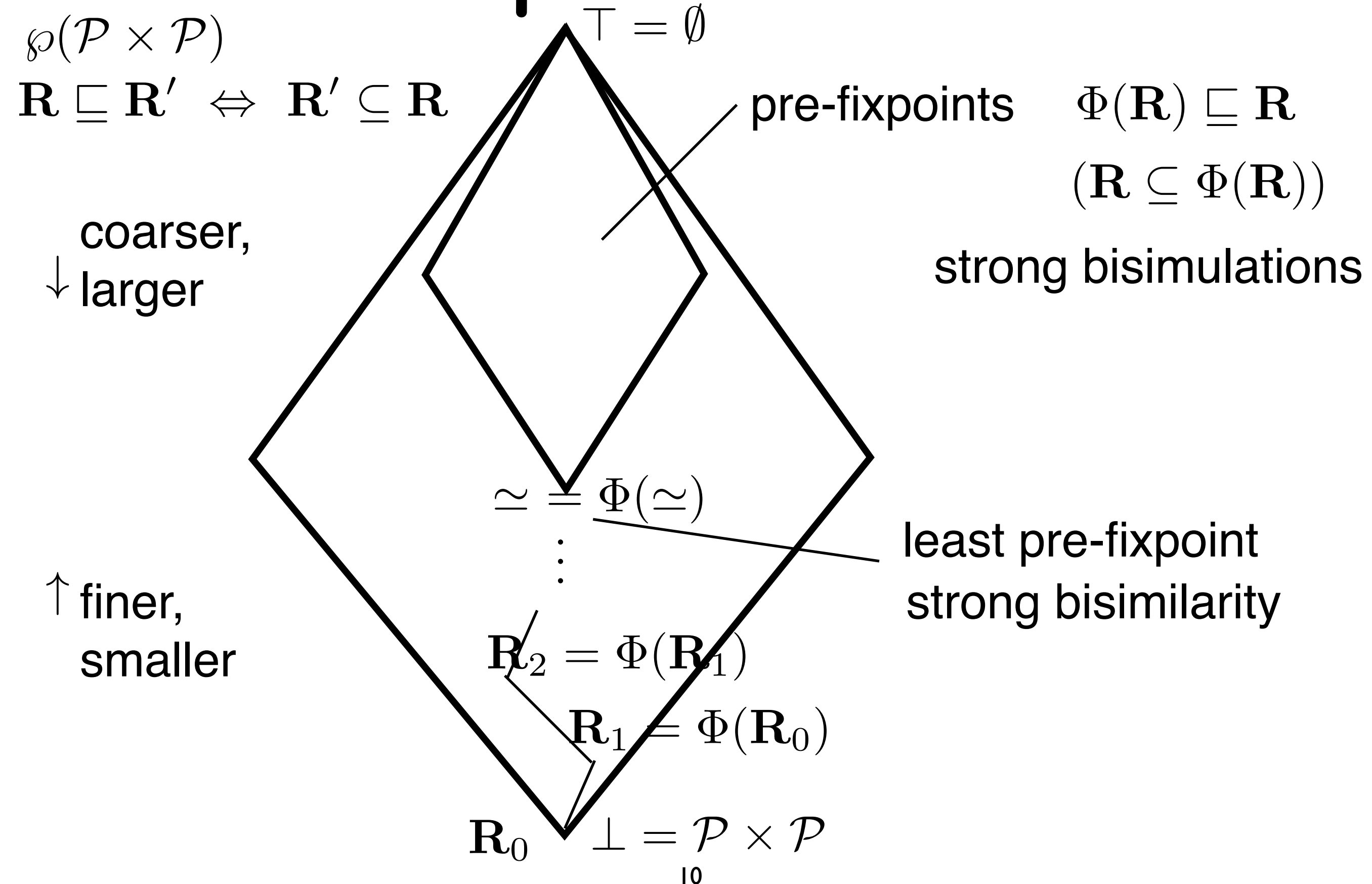
a relation with more pairs is
smaller than one with less pairs

$$(\wp(\mathcal{P} \times \mathcal{P}), \sqsubseteq)$$

$$\mathbf{R} \sqsubseteq \mathbf{R}' \quad \Leftrightarrow \quad \mathbf{R}' \subseteq \mathbf{R}$$

$$\perp = \mathcal{P} \times \mathcal{P}$$

Least fixpoint... reversed



Computing fixpoints

can we reuse Kleene's fix point theorem to compute $\simeq$?

$$\simeq \stackrel{?}{=} \bigsqcup \Phi^n(\mathcal{P} \times \mathcal{P})$$

intersection

start from the universal relation (all pairs, a unique partition)

all processes are equivalent

we apply Φ to distinguish more and more processes

$\mathbf{R}_1$ distinguishable in one step

$\mathbf{R}_2$ distinguishable in two steps

$\vdots$

the number of partitions increases at each step

TH. Φ is monotone

proof.

take $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2$ we need to prove $\Phi(\mathbf{R}_1) \sqsubseteq \Phi(\mathbf{R}_2)$

$$\mathbf{R}_2 \subseteq \mathbf{R}_1 \qquad \Phi(\mathbf{R}_2) \subseteq \Phi(\mathbf{R}_1)$$

take $(p, q) \in \Phi(\mathbf{R}_2)$ we need to prove $(p, q) \in \Phi(\mathbf{R}_1)$

take $p \xrightarrow{\mu} p'$ we want to find $q \xrightarrow{\mu} q'$ with $(p', q') \in \mathbf{R}_1$

since $(p, q) \in \Phi(\mathbf{R}_2)$ we have $q \xrightarrow{\mu} q'$ with $(p', q') \in \mathbf{R}_2 \subseteq \mathbf{R}_1$

take $q \xrightarrow{\mu} q'$ we want to find $p \xrightarrow{\mu} p'$ with $(p', q') \in \mathbf{R}_1$

analogous to the previous case

hence $(p, q) \in \Phi(\mathbf{R}_1)$

TH. Φ is continuous (for finitely branching processes)

proof.

take a chain $\{\mathbf{R}_n\}_{n \in \mathbb{N}}$

$$\mathbf{R}_0 \sqsubseteq \mathbf{R}_1 \sqsubseteq \dots \sqsubseteq \mathbf{R}_n \sqsubseteq \dots$$

$$\mathbf{R}_0 \supseteq \mathbf{R}_1 \supseteq \dots \supseteq \mathbf{R}_n \supseteq \dots$$

we need to prove $\Phi \left(\bigsqcup_{n \in \mathbb{N}} \mathbf{R}_n \right) = \bigsqcup_{n \in \mathbb{N}} \Phi(\mathbf{R}_n)$

$$\Phi \left(\bigsqcup_{n \in \mathbb{N}} \mathbf{R}_n \right) \supseteq \bigsqcup_{n \in \mathbb{N}} \Phi(\mathbf{R}_n)$$

follows from monotonicity

$$\Phi \left(\bigsqcup_{n \in \mathbb{N}} \mathbf{R}_n \right) \sqsubseteq \bigsqcup_{n \in \mathbb{N}} \Phi(\mathbf{R}_n)$$

$$\Phi \left(\bigcap_{n \in \mathbb{N}} \mathbf{R}_n \right) \supseteq \bigcap_{n \in \mathbb{N}} \Phi(\mathbf{R}_n)$$

take $(p, q) \in \bigcap_{n \in \mathbb{N}} \Phi(\mathbf{R}_n)$ we want to prove $(p, q) \in \Phi \left(\bigcap_{n \in \mathbb{N}} \mathbf{R}_n \right)$
 $\forall n. (p, q) \in \Phi(\mathbf{R}_n)$ (continue)

TH. Φ is continuous (for finitely branching processes)

proof. (continue) $(p, q) \in \bigcap_{n \in \mathbb{N}} \Phi(\mathbf{R}_n)$

$$\forall n. (p, q) \in \Phi(\mathbf{R}_n) \Rightarrow (p, q) \in \Phi \left(\bigcap_{n \in \mathbb{N}} \mathbf{R}_n \right)$$

take $p \xrightarrow{\mu} p'$ we want to find $q \xrightarrow{\mu} q'$ with $(p', q') \in \bigcap_{n \in \mathbb{N}} \mathbf{R}_n$

$$\forall n. (p', q') \in \mathbf{R}_n$$

since $\forall n. (p, q) \in \Phi(\mathbf{R}_n)$ then $\forall n. \exists q_n. q \xrightarrow{\mu} q_n$ with $(p', q_n) \in \mathbf{R}_n$

$$\mathbf{R}_0 \supseteq \mathbf{R}_1 \supseteq \dots \supseteq \mathbf{R}_n \supseteq \dots \quad \forall k \leq n. (p', q_n) \in \mathbf{R}_k$$

q is finitely branching: $\{q' \mid q \xrightarrow{\mu} q'\}$ is finite

thus $\exists m \in \mathbb{N}$ such that $\{n \mid q_n = q_m\}$ is infinite

hence $\forall n. (p', q_m) \in \mathbf{R}_n$ and we take $q' = q_m$

take $q \xrightarrow{\mu} q'$ we want to find $p \xrightarrow{\mu} p'$ with $(p', q') \in \bigcap_{n \in \mathbb{N}} \mathbf{R}_n$

analogous to the previous case

Strong bis as fix

$\mathcal{P}_f$ finitely branching processes

$$\simeq = \bigsqcup_{n \in \mathbb{N}} \Phi^n(\mathcal{P}_f \times \mathcal{P}_f)$$

how do we know a process is finitely branching?

we can restrict the syntax: guarded processes

Example

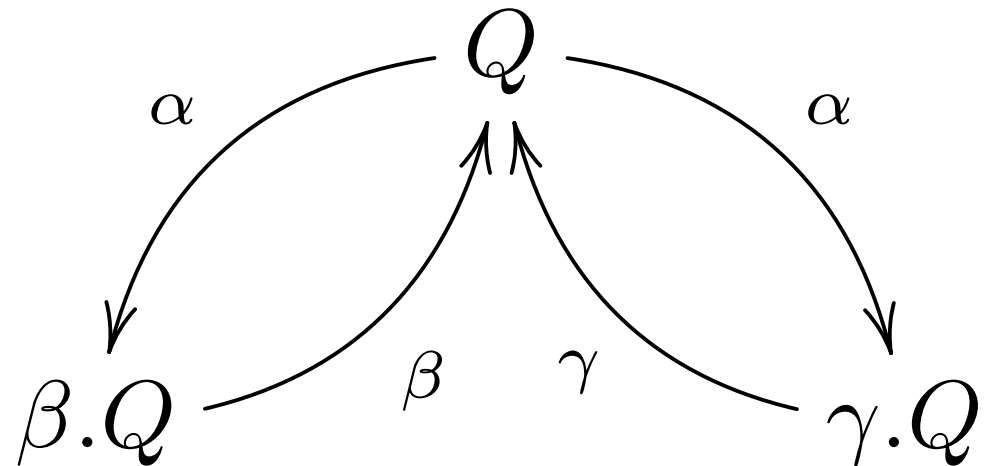
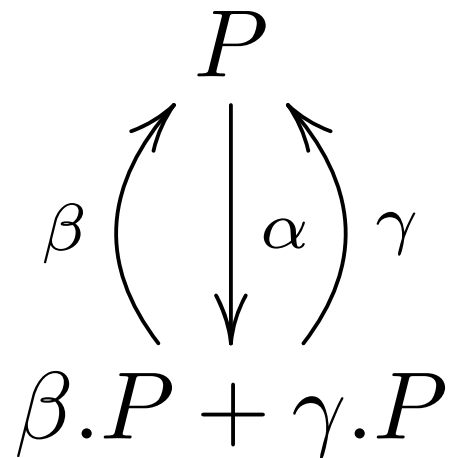
Guarded!

$$P \triangleq \alpha.(\beta.P + \gamma.P)$$

$$P \stackrel{?}{\simeq} Q$$

Guarded!

$$Q \triangleq \alpha.\beta.Q + \alpha.\gamma.Q$$

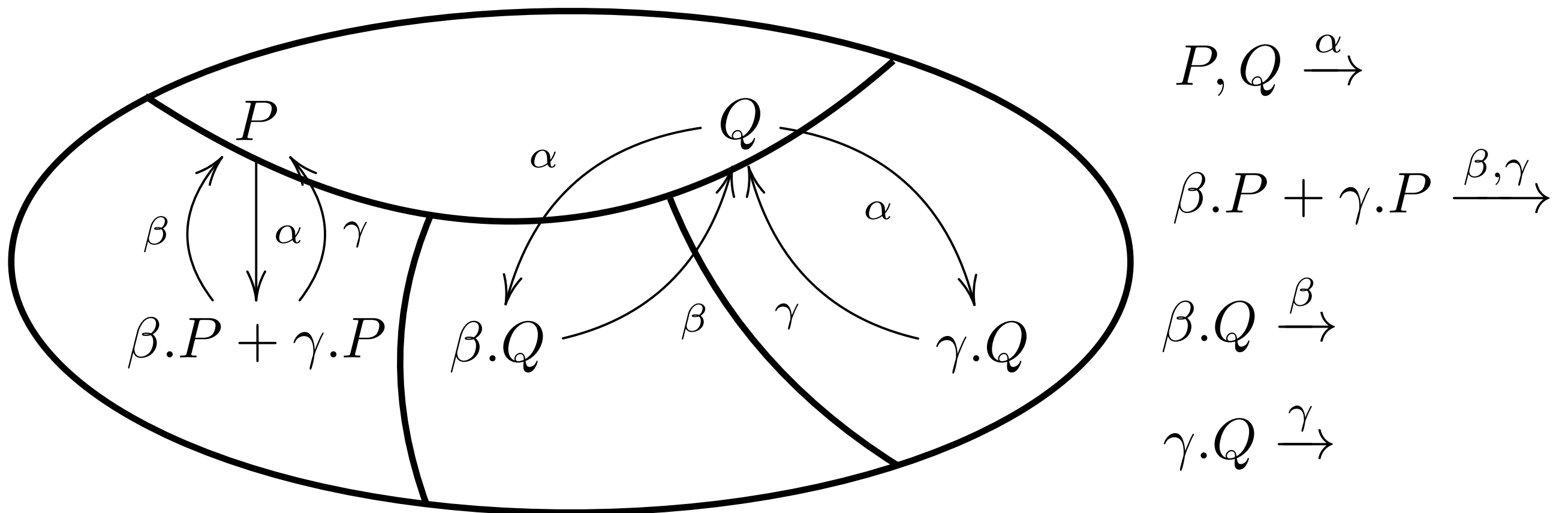


$$\mathbf{R}_0 = \{ \{P, Q, \beta.P + \gamma.P, \beta.Q, \gamma.Q\} \}$$

Example

$$P \triangleq \alpha.(\beta.P + \gamma.P) \qquad P \stackrel{?}{\simeq} Q \qquad Q \triangleq \alpha.\beta.Q + \alpha.\gamma.Q$$

$$\mathbf{R}_0 = \{ \{P, Q, \beta.P + \gamma.P, \beta.Q, \gamma.Q\} \}$$

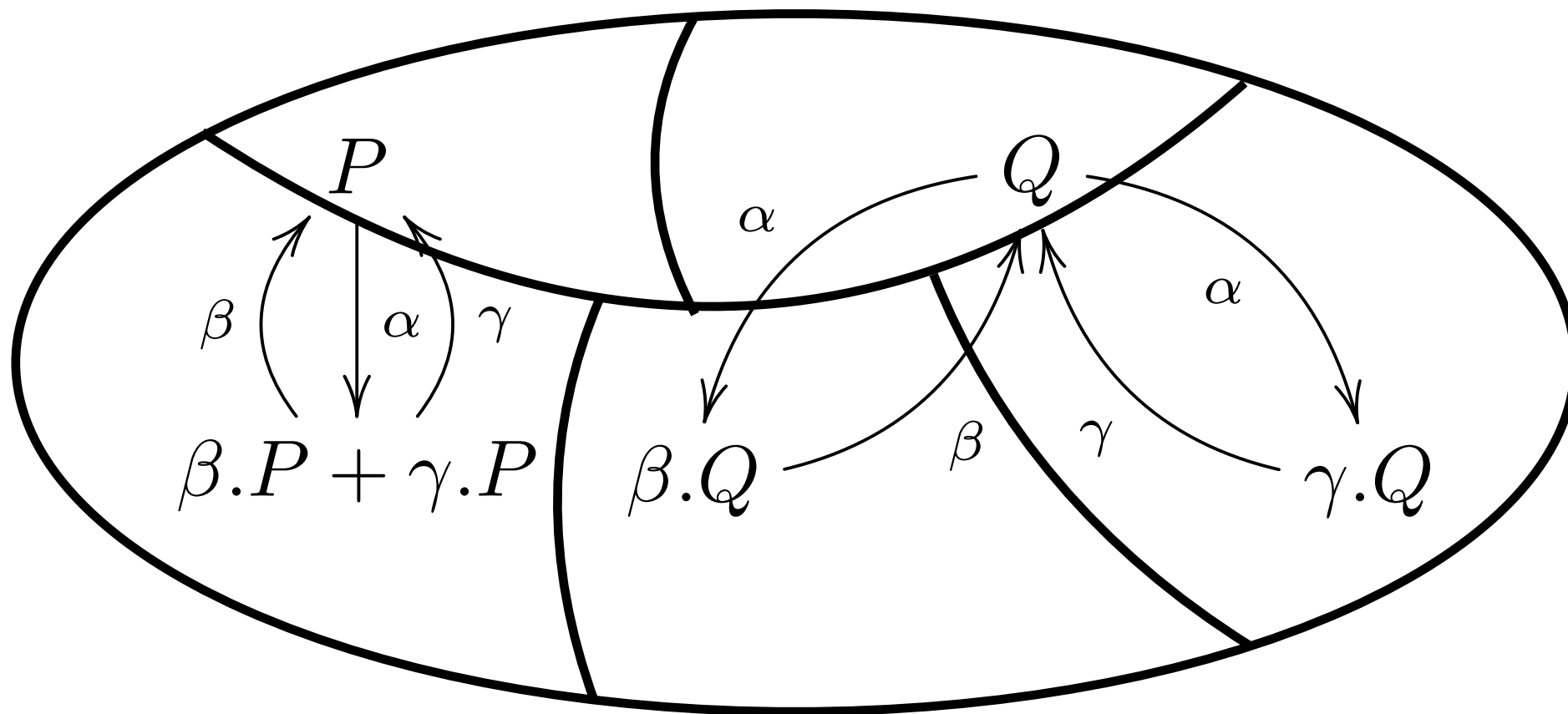


Processes with different capabilities must be distinguished

Example

$$P \triangleq \alpha.(\beta.P + \gamma.P) \qquad P \stackrel{?}{\simeq} Q \qquad Q \triangleq \alpha.\beta.Q + \alpha.\gamma.Q$$

$$\mathbf{R}_1 = \{ \{P, Q\}, \{\beta.P + \gamma.P\}, \{\beta.Q\}, \{\gamma.Q\} \}$$



$$P \xrightarrow{\alpha} [\beta.P + \gamma.Q]$$

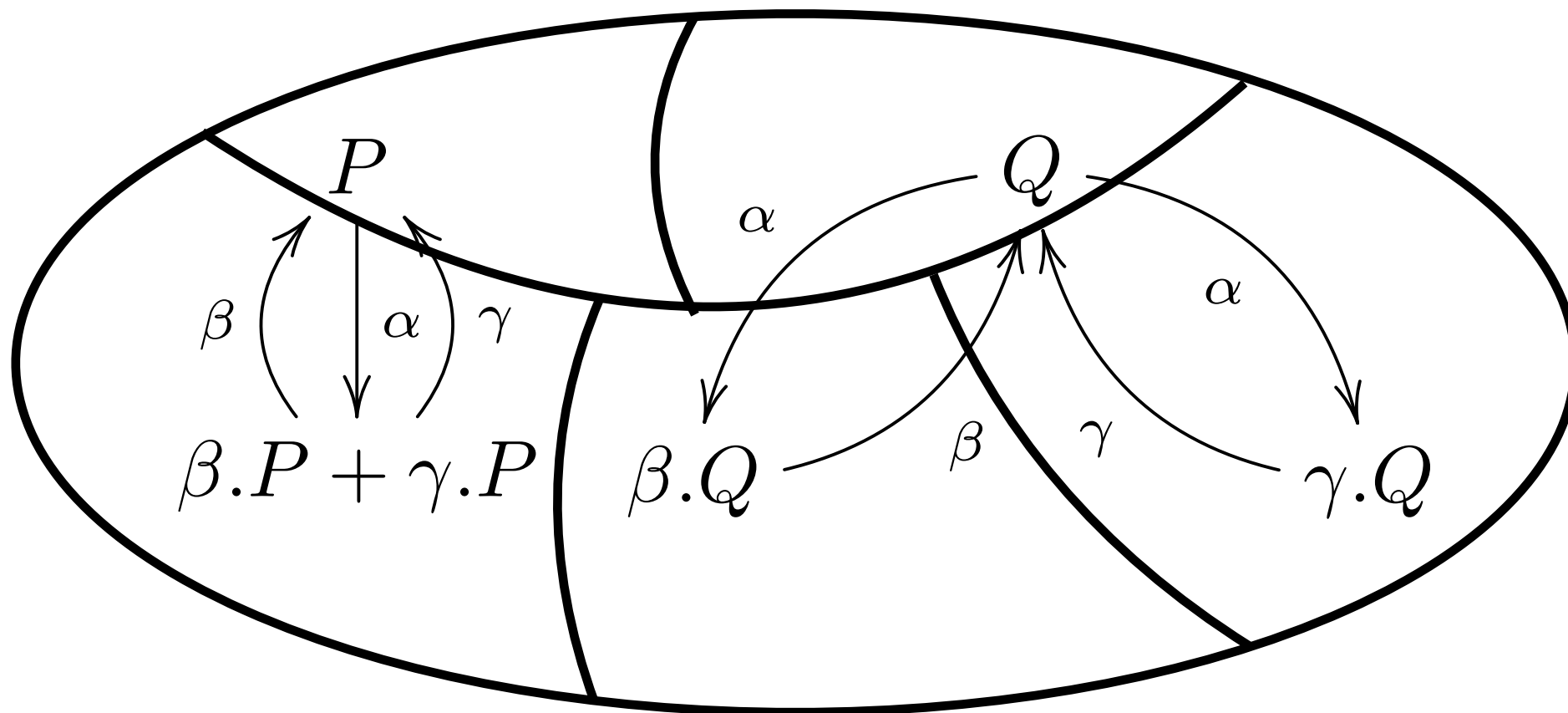
$$Q \xrightarrow{\alpha} [\beta.Q], [\gamma.Q]$$

The α transitions of P and Q ends in different partitions

Example

$$P \triangleq \alpha.(\beta.P + \gamma.P) \qquad P \stackrel{?}{\simeq} Q \qquad Q \triangleq \alpha.\beta.Q + \alpha.\gamma.Q$$

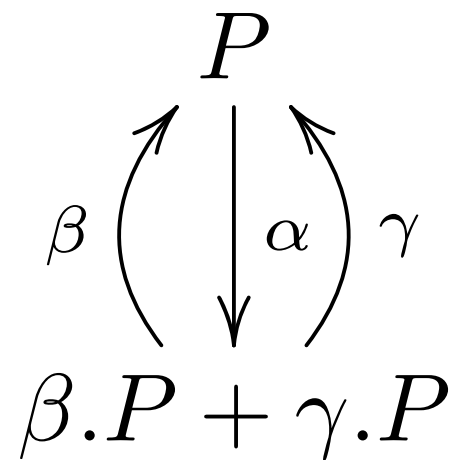
$$\mathbf{R}_2 = \{ \{P\} , \{Q\} , \{\beta.P + \gamma.P\} , \{\beta.Q\} , \{\gamma.Q\} \}$$



Example

Guarded!

$$P \triangleq \alpha.(\beta.P + \gamma.P)$$

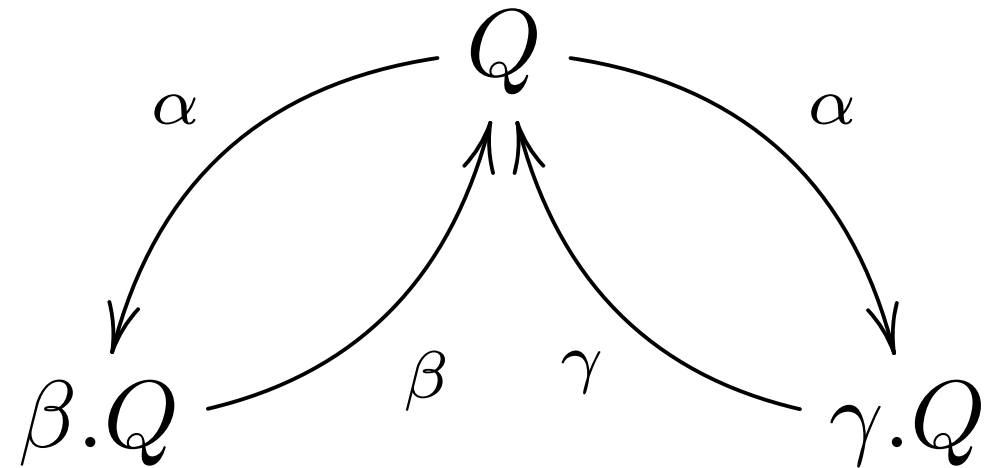


$$P \stackrel{?}{\simeq} Q$$



Guarded!

$$Q \triangleq \alpha.\beta.Q + \alpha.\gamma.Q$$



$$\mathbf{R}_0 = \{ \{P, Q, \beta.P + \gamma.P, \beta.Q, \gamma.Q\} \}$$

$$\mathbf{R}_1 = \{ \{P, Q\}, \{\beta.P + \gamma.P\}, \{\beta.Q\}, \{\gamma.Q\} \}$$

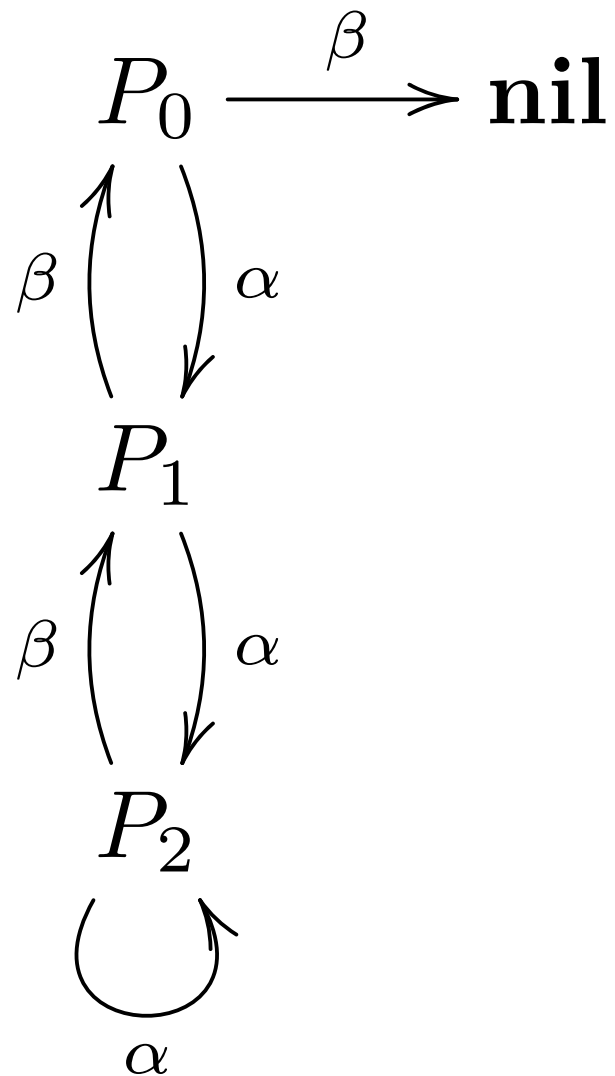
$$\mathbf{R}_2 = \{ \{P\}, \{Q\}, \{\beta.P + \gamma.P\}, \{\beta.Q\}, \{\gamma.Q\} \}$$

Only singletons partitions, we can stop $P \not\approx Q$



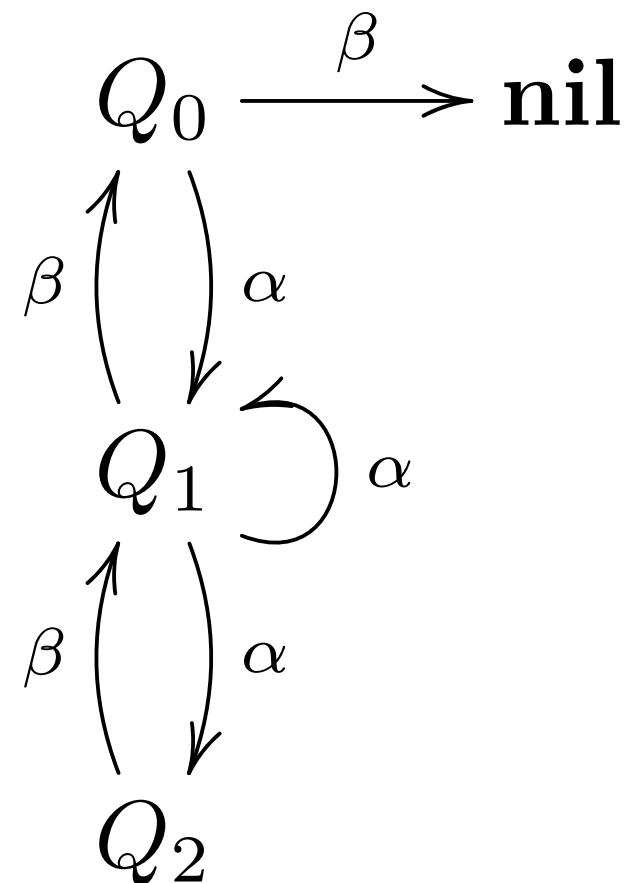
Exercise

finitely branching!



$$P_0 \stackrel{?}{\simeq} Q_0$$

finitely branching!



nil $\nrightarrow$

$Q_2 \xrightarrow{\beta}$

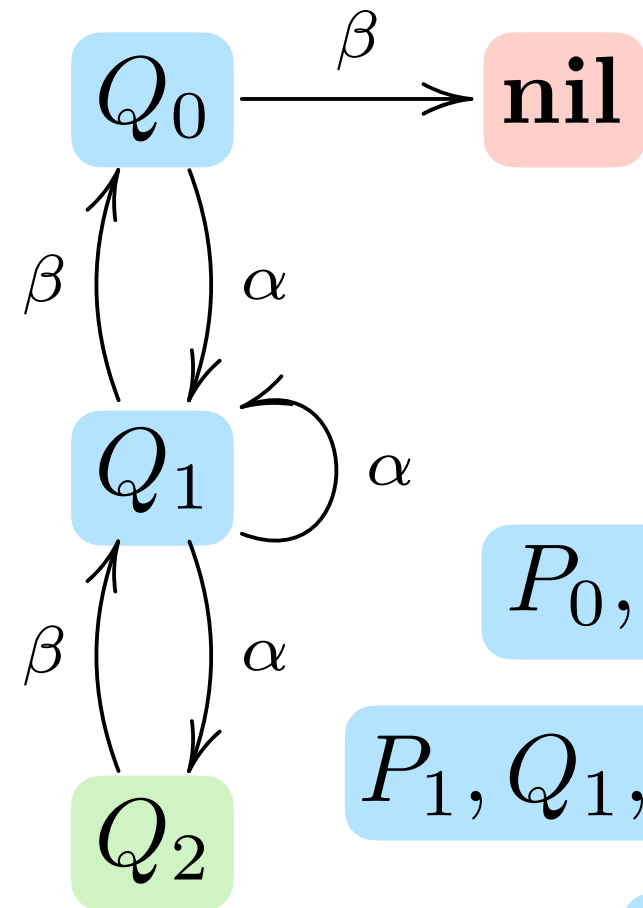
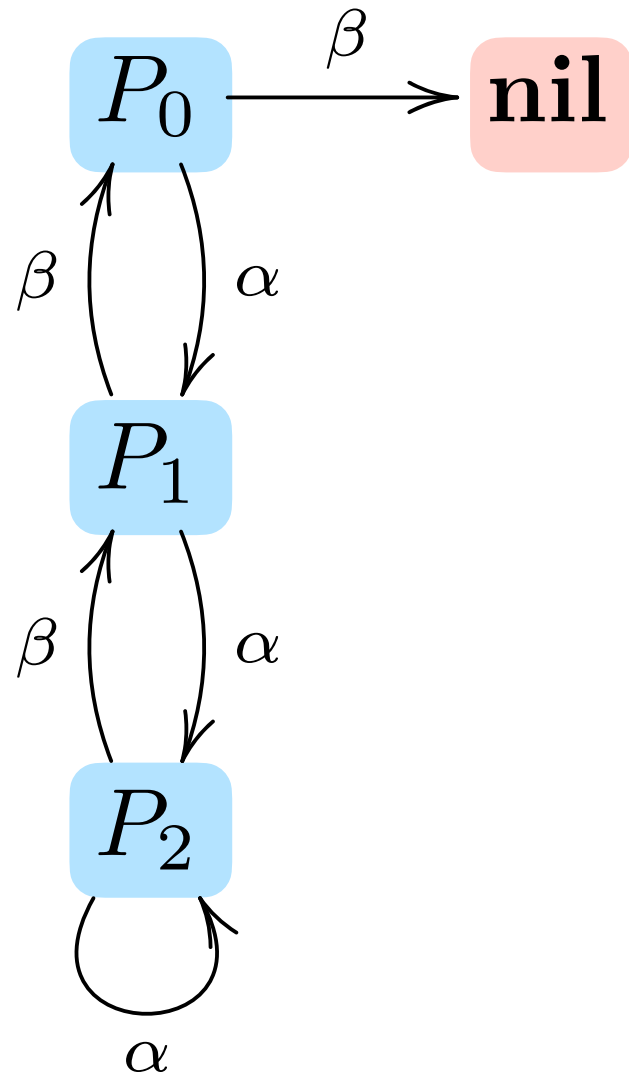
$P_0, Q_0, P_1, Q_1, P_2 \xrightarrow{\alpha, \beta}$

$$\mathbf{R}_0 = \{ \{P_0, Q_0, P_1, Q_1, P_2, Q_2, \mathbf{nil}\} \}$$



Exercise

$$P_0 \stackrel{?}{\simeq} Q_0$$



$$P_0, Q_0 \xrightarrow{\beta} [\text{nil}]$$

$$P_1, Q_1, P_2 \not\xrightarrow{\beta} [\text{nil}]$$

$$Q_1 \xrightarrow{\alpha} [Q_2]$$

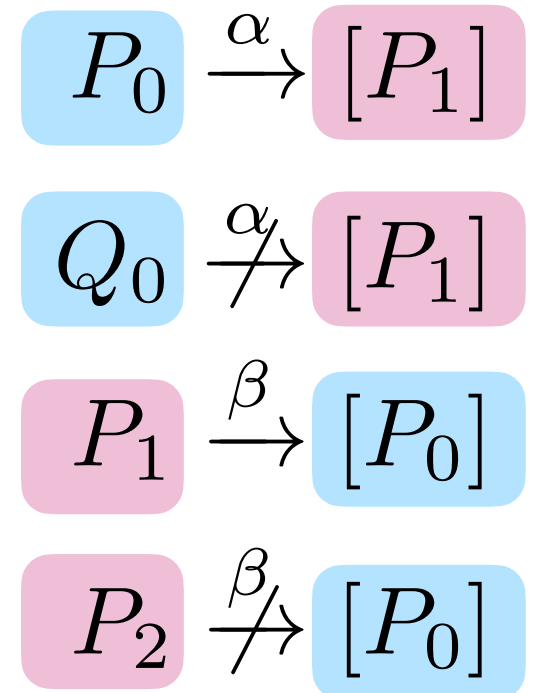
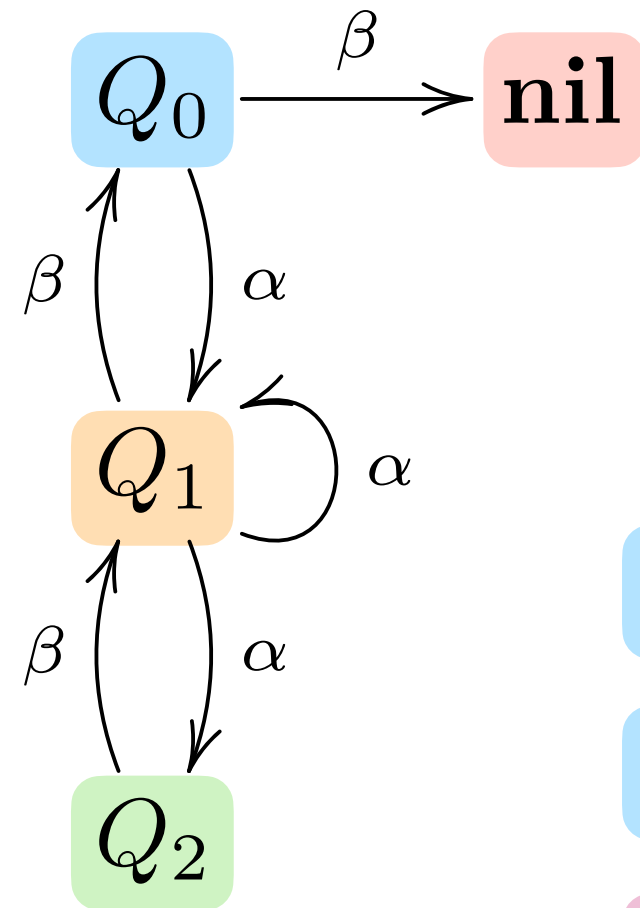
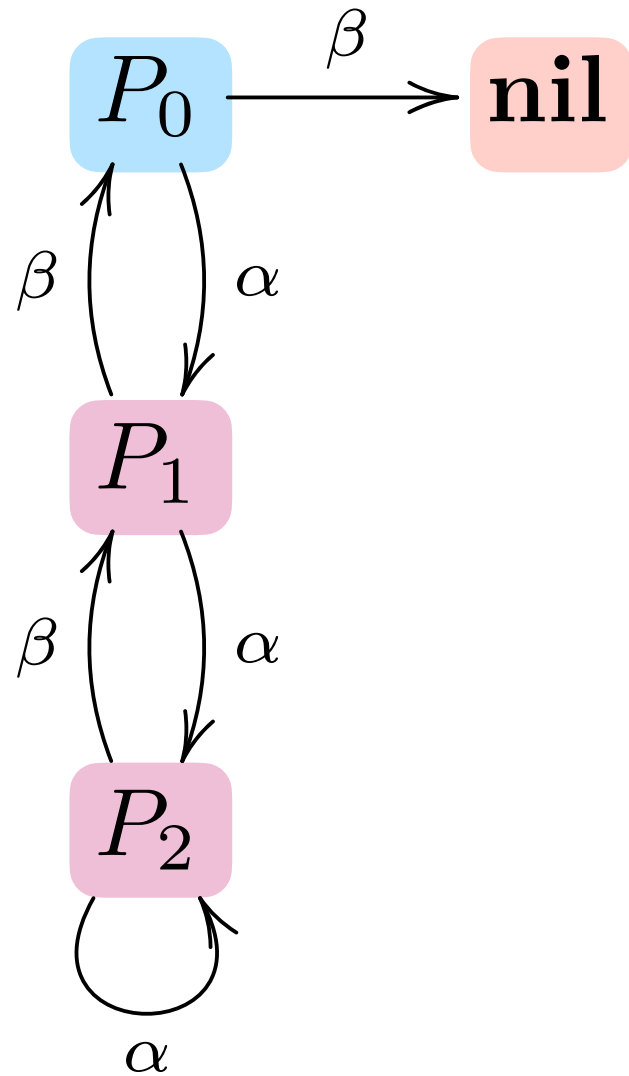
$$P_1, P_2 \not\xrightarrow{\alpha} [Q_2]$$

$$\mathbf{R}_1 = \{ \{P_0, Q_0, P_1, Q_1, P_2\}, \{Q_2\}, \{\text{nil}\} \}$$



Exercise

$$P_0 \stackrel{?}{\simeq} Q_0$$

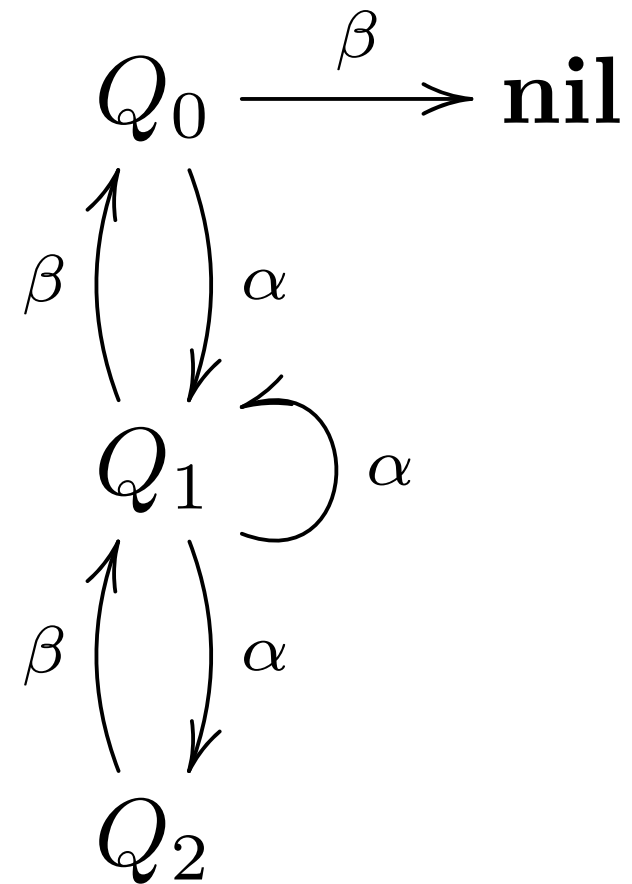
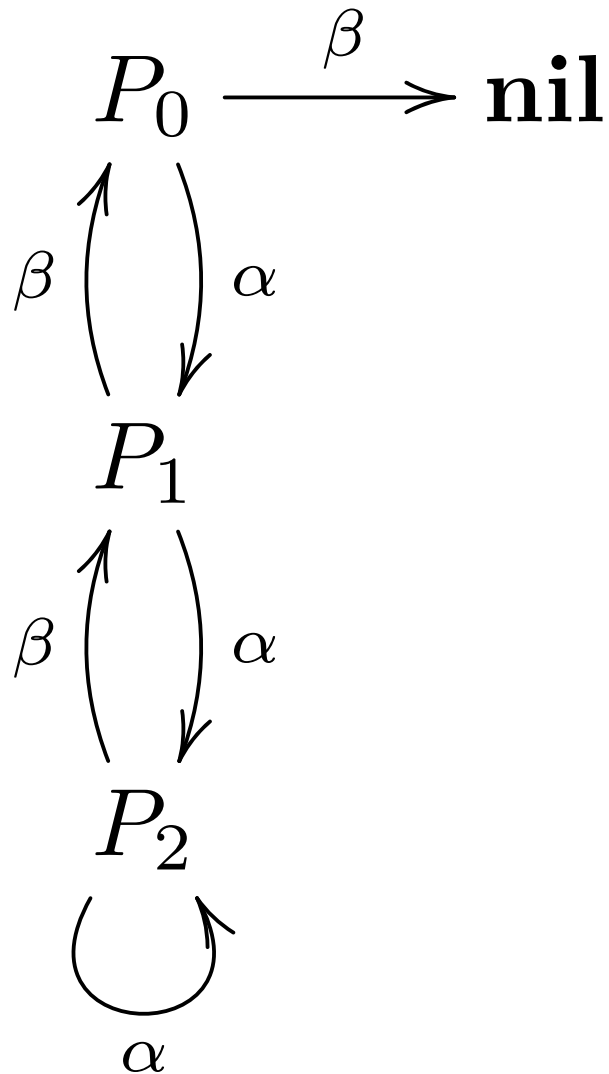


$$\mathbf{R}_2 = \{ \{P_0, Q_0\}, \{P_1, P_2\}, \{Q_1\}, \{Q_2\}, \{\text{nil}\} \}$$



Exercise

$$P_0 \stackrel{?}{\simeq} Q_0$$



$$P_0 \not\simeq Q_0$$

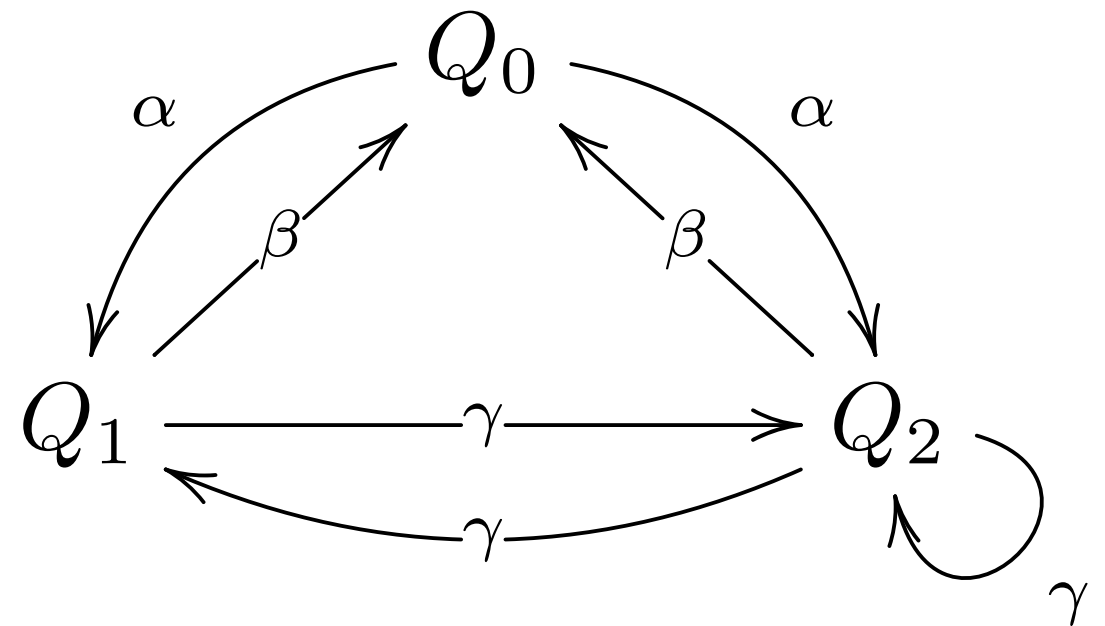
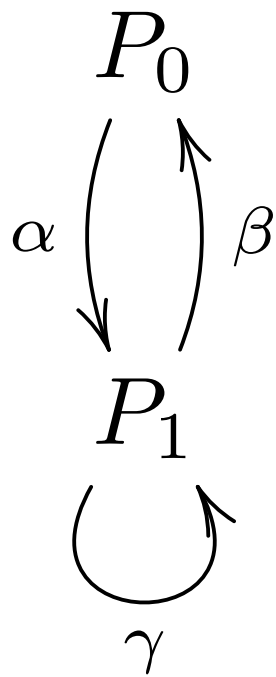
$$\mathbf{R}_3 = \{ \{P_0\}, \{Q_0\}, \{P_1\}, \{P_2\}, \{Q_1\}, \{Q_2\}, \{\mathbf{nil}\} \}$$

Exercise

finitely branching!

$$P_0 \stackrel{?}{\simeq} Q_0$$

finitely branching!



$$\mathbf{R}_0 = \{ \{P_0, Q_0, P_1, Q_1, Q_2\} \}$$

$$\begin{array}{l} P_0, Q_0 \xrightarrow{\alpha} \\ P_1, Q_1, Q_2 \xrightarrow{\beta, \gamma} \end{array}$$

$$\mathbf{R}_1 = \{ \{P_0, Q_0\} , \{P_1, Q_1, Q_2\} \}$$

No more reasons to discriminate!

Unguarded processes?

What about the general case? (unguarded processes)

any powerset ordered by inclusion defines a complete lattice

Complete lattice: $(D, \sqsubseteq)$ PO such that

any $X \subseteq D$ has a least upper bound $\bigsqcup X$

any $X \subseteq D$ has a greatest lower bound $\bigsqcap X$

it has bottom and top elements $\perp = \bigsqcap D$ $\top = \bigsqcup D$

TH. [Knaster-Tarski] $(D, \sqsubseteq)$ complete lattice

$f : D \rightarrow D$ monotone

has least and greatest fixpoint

$d_{\min} \triangleq \bigsqcap \{d \in D \mid f(d) \sqsubseteq d\}$ is the least fixpoint

glb pre-fixpoints

$d_{\max} \triangleq \bigsqcup \{d \in D \mid d \sqsubseteq f(d)\}$ is the greatest fixpoint

lub post-fixpoints

least and greatest fixpoint exist... but how to compute them?