From Bisimilarity to ε-Bisimilarity

Ana Sokolova

PARIS LODRON UNIVERSITY SALZBURG

Why formal methods?

Therac-25

Therac-25 is a computer-controlled radiation therapy machine produced by Atomic Energy of Canada Limited (AECL) in 2 after the Therac-6 and Therac-20 units (the earlier units had been produced in partnership with Compagnie générale de radiologie (CGR) of France).[1]

The Therac-25 was involved in at least six accidents between 1985 and 1987, in which some patients were given massive overdoses of radiation.[2]:425 Because of concurrent programming errors (also known as race conditions), it sometimes gave its patients radiation doses that were hundreds of times greater than normal, resulting in death or serious injury.[3] These accidents highlighted the dangers of software control of safety-critical systems.

The Therac-25 has become a standard case study in health informatics, software engineering, and computer ethics. It highlights the dangers of engineer overconfidence[2]:428 after the engineers dismissed end-user reports, leading to severe consequences.

History [edit]

The French company CGR manufactured the *Neptune* and *Sagittaire* linear accelerators.

In the early 1970s, CGR and the Canadian public company *Atomic Energy of Canada Limited* (AECL) collaborated on the construction of linear accelerators controlled by a DEC

Ariane-5

"Cluster (spacecraft)" redirects here. This article is about the failed Cluster launch. For the successful space mission, see Cluster II (spacecraft).

Ariane flight V88[1] was the failed maiden flight of the Arianespace Ariane 5 rocket, vehicle no. 501, on 4 June 1996. It carried the **Cluster** spacecraft, a constellation of four European Space Agency research satellites.

The launch ended in failure due to multiple errors in the software design: dead code, intended only for Ariane 4, with inadequate protection against integer overflow led to an exception handled inappropriately, halting the whole otherwise unaffected inertial navigation system. This caused the rocket to veer off its flight path 37 seconds after launch, beginning to disintegrate under high aerodynamic forces, and finally self-destructing via its automated flight termination system. The failure has become known as one of the most infamous and expensive software bugs in history.[2] The failure resulted in a loss of more than US$370 million.[3]

Launch failure [edit]

Cluster

Mission type	Magnetospheric
Operator	ESA
Spacecraft properties	
Launch mass	1,200 kilograms (2,600 lb)
Start of mission	
Launch date	12:34:06, 4 June 1996 (UTC)
Rocket	Ariane 5G
Launch site	Kourou ELA-3
End of mission	
Disposal	launch failure
Destroyed	4 June 1996

esa

Intel Pentium

"Pentium bug" redirects here. For the 1997 bug affecting Pentium processors, see Pentium F00F bug.

The **Pentium FDIV bug** is a hardware bug affecting the floating-point unit (FPU) of the early Intel Pentium processors. Because of the bug, the processor would return incorrect binary floating point results when dividing certain pairs of high-precision numbers. The bug was discovered in 1994 by Thomas R. Nicely, a professor of mathematics at Lynchburg College.[1] Missing values in a lookup table used by the FPU's floating-point division algorithm led to calculations acquiring small errors. While these errors would in most use-cases only occur rarely and result in small deviations from the correct output values, in certain circumstances the errors can occur frequently and lead to more significant deviations.[2]

The severity of the FDIV bug is debated. Though rarely encountered by most users (*Byte*

66 MHz Intel Pentium

Why formal methods?

Therac-25

Therac-25 is a computer-controlled radiation therapy machine produced by Atomic Energy of Canada Limited (AECL) in 2 after the Therac-6 and Therac-20 units (the earlier units had been produced in partnership with Compagnie générale de radiologie (CGR) of France).[1]

The Therac-25 was involved in at least six accidents between 1985 and 1987, in which some patients were given massive overdoses of radiation.[2]:425 Because of concurrent programming errors (also known as race conditions), it sometimes gave its patients radiation doses that were hundreds of times greater than normal, resulting in death or serious injury.[3] These accidents highlighted the dangers of software control of safety-critical systems.

, and computer ethics. It highlights
s, leading to severe

Patients were seriously injured and died

Ariane-5

"Cluster (spacecraft)" redirects here. This article is about the failed Cluster launch. For the successful space mission, see Cluster II (spacecraft).

Ariane flight V88[1] was the failed maiden flight of the Arianespace Ariane 5 rocket, vehicle no. 501, on 4 June 1996. It carried the **Cluster** spacecraft, a constellation of four European Space Agency research satellites.

The launch ended in failure due to multiple errors in the software design: dead code, intended only for Ariane 4, with inadequate protection against integer overflow led to an exception handled inappropriately, halting the whole otherwise unaffected inertial navigation system. This caused the rocket to veer off its flight path 37 seconds after launch, beginning to disintegrate under high aerodynamic forces, and finally self-destructing via its automated flight termination system. The failure has become known as one of the most infamous and expensive software bugs in history.[2] The failure resulted in a loss of more than US$370 million.[3]

Launch failure [edit]

Cluster

Mission type	Magnetospheric
Operator	ESA
Spacecraft properties	
Launch mass	1,200 kilograms (2,600 lb)
Start of mission	
Launch date	12:34:06, 4 June 1996 (UTC)
Rocket	Ariane 5G
Launch site	Kourou ELA-3
End of mission	
Disposal	launch failure
Destroyed	4 June 1996

esa

Intel Pentium

"Pentium bug" redirects here. For the 1997 bug affecting Pentium processors, see Pentium F00F bug.

The **Pentium FDIV bug** is a hardware bug affecting the floating-point unit (FPU) of the early Intel Pentium processors. Because of the bug, the processor would return incorrect binary floating point results when dividing certain pairs of high-precision numbers. The bug was discovered in 1994 by Thomas R. Nicely, a professor of mathematics at Lynchburg College.[1] Missing values in a lookup table used by the FPU's floating-point division algorithm led to calculations acquiring small errors. While these errors would in most use-cases only occur rarely and result in small deviations from the correct output values, in certain circumstances the errors can occur frequently and lead to more significant deviations.[2]

The severity of the FDIV bug is debated. Though rarely encountered by most users (*Byte*

66 MHz Intel Pentium

Why formal methods?

Therac-25

Therac-25 is a computer-controlled radiation therapy machine produced by Atomic Energy of Canada Limited (AECL) in 2 after the Therac-6 and Therac-20 units (the earlier units had been produced in partnership with Compagnie générale de radiologie (CGR) of France).[1]

The Therac-25 was involved in at least six accidents between 1985 and 1987, in which some patients were given massive overdoses of radiation.[2]:425 Because of concurrent programming errors (also known as race conditions), it sometimes gave its patients radiation doses that were hundreds of times greater than normal, resulting in death or serious injury.[3] These accidents highlighted the dangers of software control of safety-critical systems.

, and computer ethics. It highlights
s, leading to severe

Patients were seriously injured and died

Ariane-5

launch. For the successful space mission, see

Cluster

Mission type	Magnetospheric
Operator	ESA
Spacecraft properties	
Launch mass	1,200 kilograms (2,600 lb)
Start of mission	
Launch date	12:34:06, 4 June 1996 (UTC)
Rocket	Ariane 5G
Launch site	Kourou ELA-3
End of mission	
Disposal	launch failure
Destroyed	4 June 1996

esa

Intel Pentium

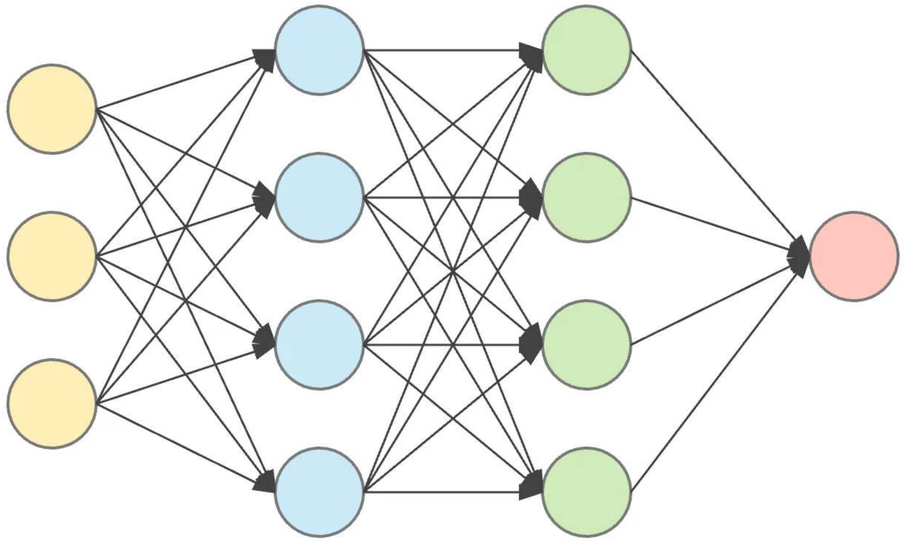
"Pentium bug" redirects here. For the 1997 bug affecting Pentium processors, see Pentium F00F bug.

The **Pentium FDIV bug** is a hardware bug affecting the floating-point unit (FPU) of the early Intel Pentium processors. Because of the bug, the processor would return incorrect binary floating point results when dividing certain pairs of high-precision numbers. The bug was discovered in 1994 by Thomas R. Nicely, a professor of mathematics at Lynchburg College.[1] Missing values in a lookup table used by the FPU's floating-point division algorithm led to calculations acquiring small errors. While these errors would in most use-cases only occur rarely and result in small deviations from the correct output values, in certain circumstances the errors can occur frequently and lead to more significant deviations.[2]

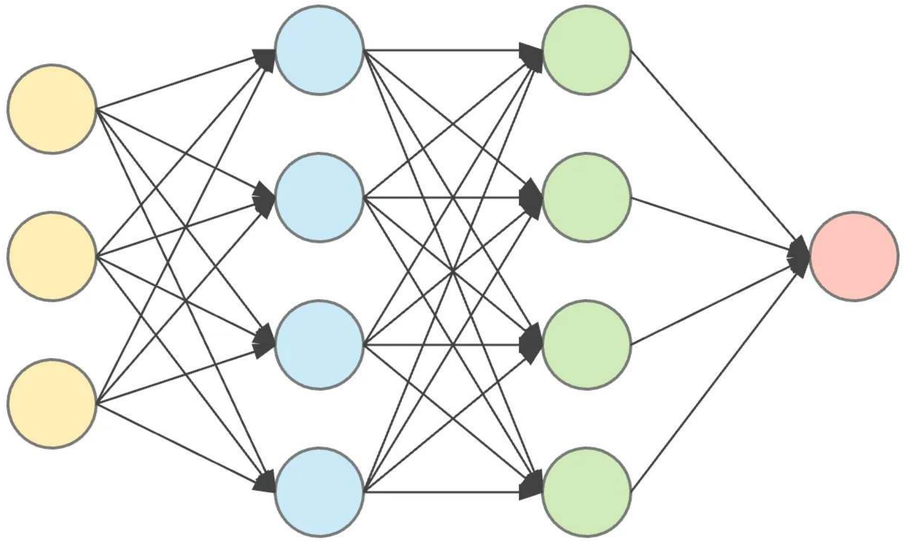
The severity of the FDIV bug is debated. Though rarely encountered by most users (*Byte*

66 MHz Intel Pentium

Why formal methods?



Why formal methods?



Do they do what we expect them to do ?

How do formal methods help?

How do formal methods help?

In my work: with behavioural equivalences, like bisimilarity or trace semantics

How do formal methods help?

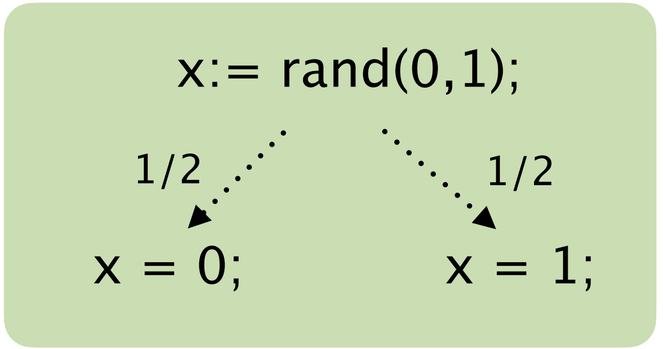
In my work: with behavioural equivalences, like bisimilarity or trace semantics

probabilistic programs

How do formal methods help?

In my work: with behavioural equivalences, like bisimilarity or trace semantics

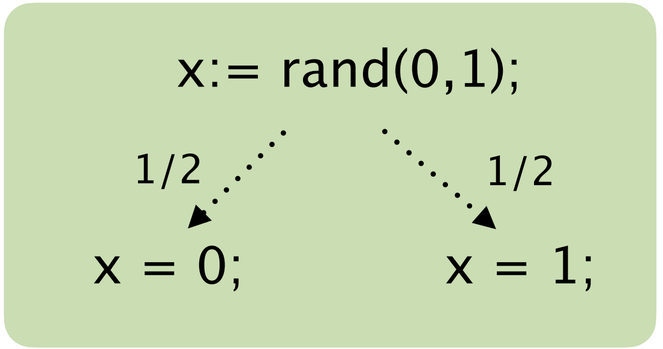
probabilistic programs



How do formal methods help?

In my work: with behavioural equivalences, like bisimilarity or trace semantics

probabilistic programs

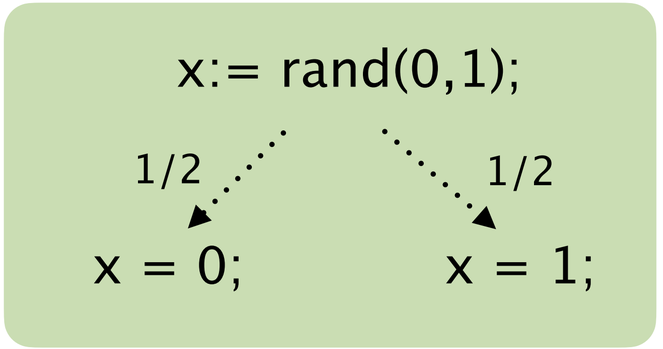


verifying **equivalence** of programs

How do formal methods help?

In my work: with behavioural equivalences, like bisimilarity or trace semantics

probabilistic programs



verifying **equivalence** of programs

P ≡ Optimised(P)

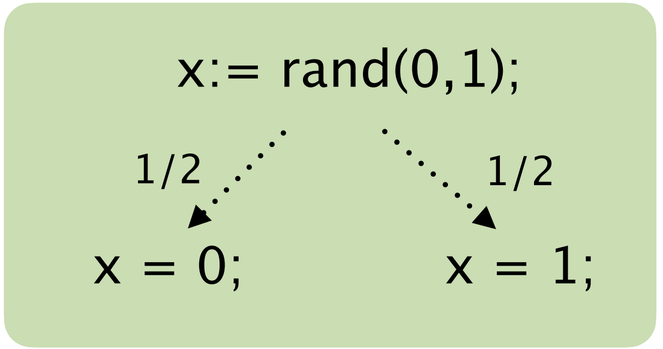
Source ≡ Compiled

Implementation ≡ Specification

How do formal methods help?

In my work: with behavioural equivalences, like bisimilarity or trace semantics

probabilistic programs



verifying **equivalence** of programs

P ≡ Optimised(P)

Source ≡ Compiled

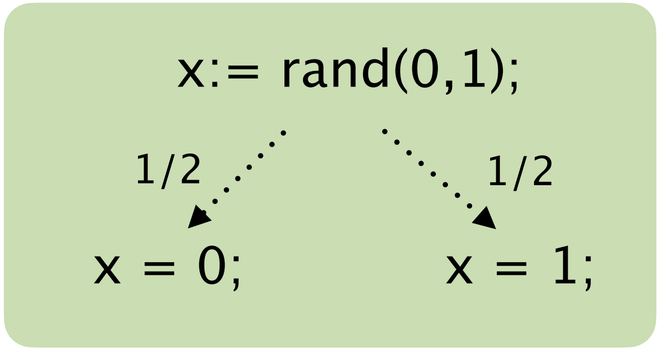
Implementation ≡ Specification

developing **proof techniques**

How do formal methods help?

In my work: with behavioural equivalences, like bisimilarity or trace semantics

probabilistic programs



verifying **equivalence** of programs

P ≡ Optimised(P)

Source ≡ Compiled

Implementation ≡ Specification

developing **proof techniques**

e.g. by axiomatisations / up-to techniques

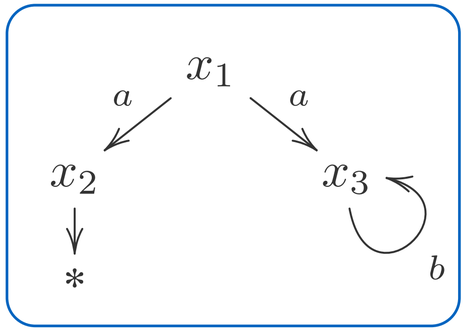
Coalgebras and Bisimilarity

Some coalgebras

Some coalgebras

NFA

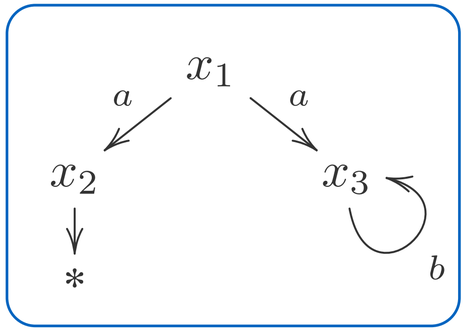
$X \to 2 \times (\mathcal{P}X)^A$



Some coalgebras

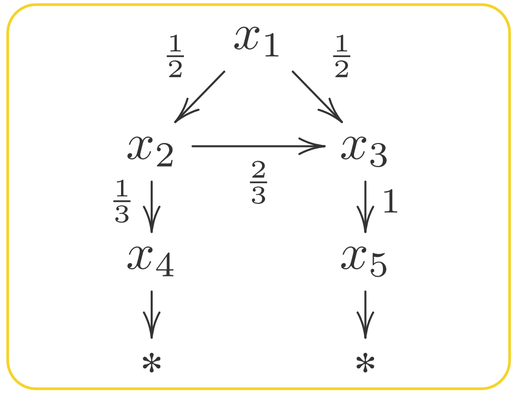
NFA

$X \to 2 \times (\mathcal{P}X)^A$



MC

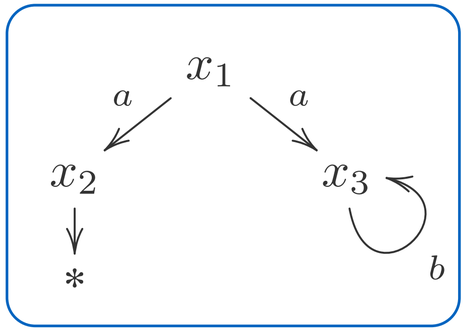
$X \to \mathcal{D}X + 1$



Some coalgebras

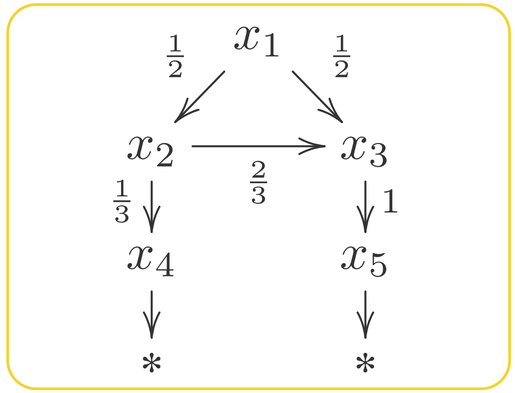
NFA

$X \to 2 \times (\mathcal{P}X)^A$



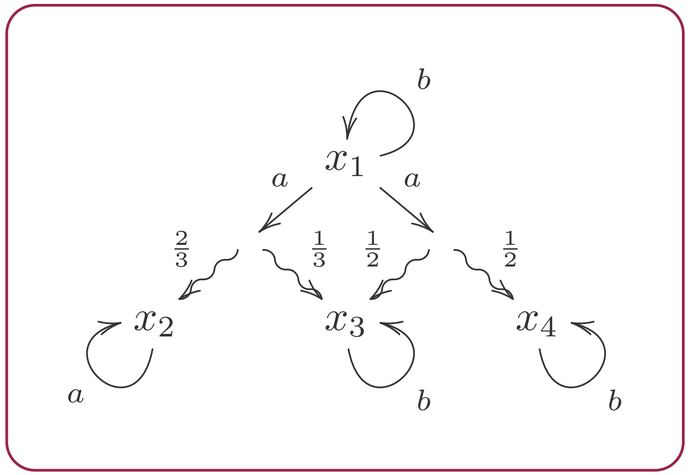
MC

$X \to \mathcal{D}X + 1$



PA

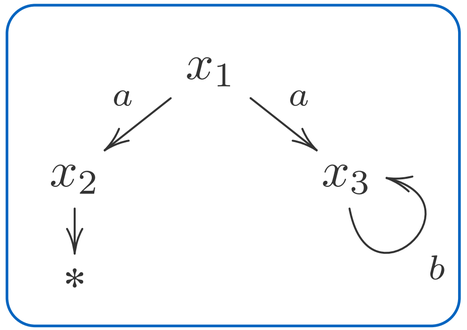
$X \to (\mathcal{P}\mathcal{D}X)^A$



Some coalgebras

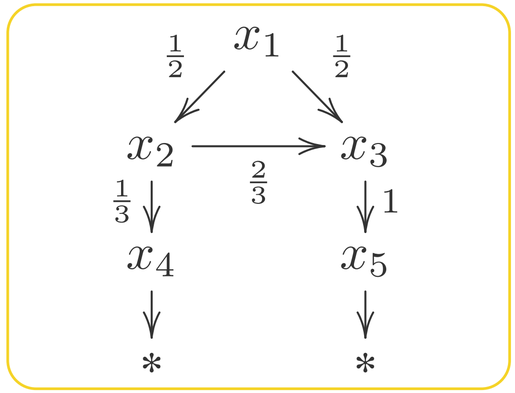
NFA

$X \to 2 \times (\mathcal{P}X)^A$



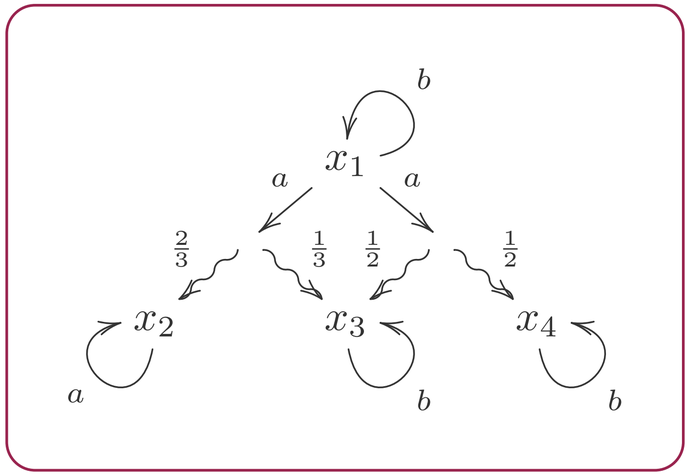
MC

$X \to \mathcal{D}X + 1$



PA

$X \to (\mathcal{P}\mathcal{D}X)^A$



Various transitions systems / automata are coalgebras

Coalgebra

Uniform framework for dynamic transition systems, based on category theory.

A coalgebra is a generic transition system:

Coalgebra

Uniform framework for dynamic transition systems, based on category theory.

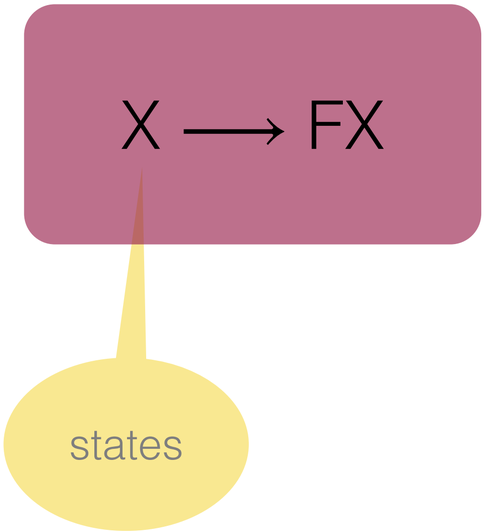
A coalgebra is a generic transition system:

$$X \longrightarrow FX$$

Coalgebra

Uniform framework for dynamic transition systems, based on category theory.

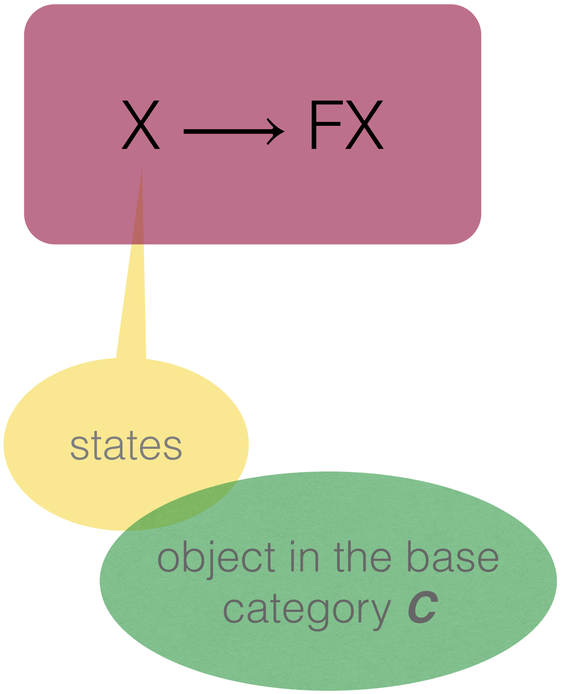
A coalgebra is a generic transition system:



Coalgebra

Uniform framework for dynamic transition systems, based on category theory.

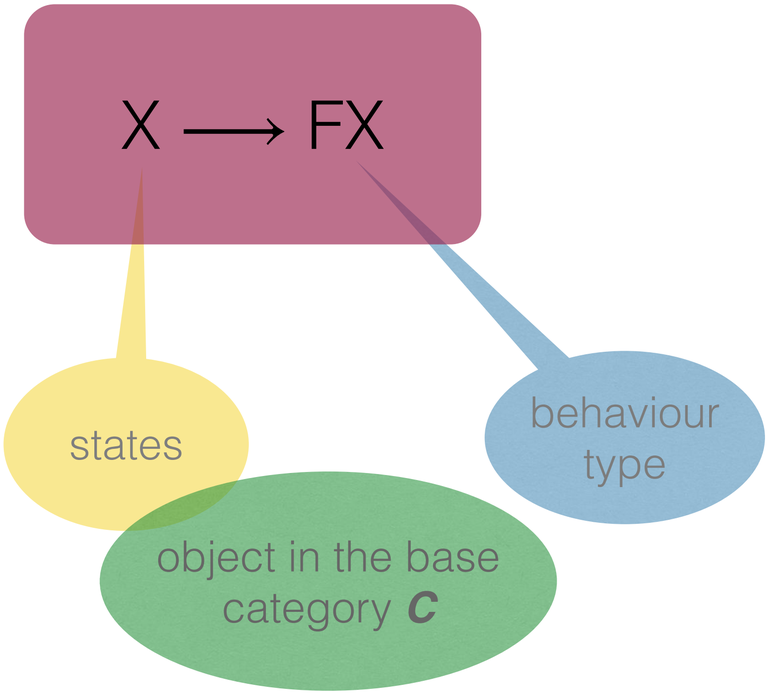
A coalgebra is a generic transition system:



Coalgebra

Uniform framework for dynamic transition systems, based on category theory.

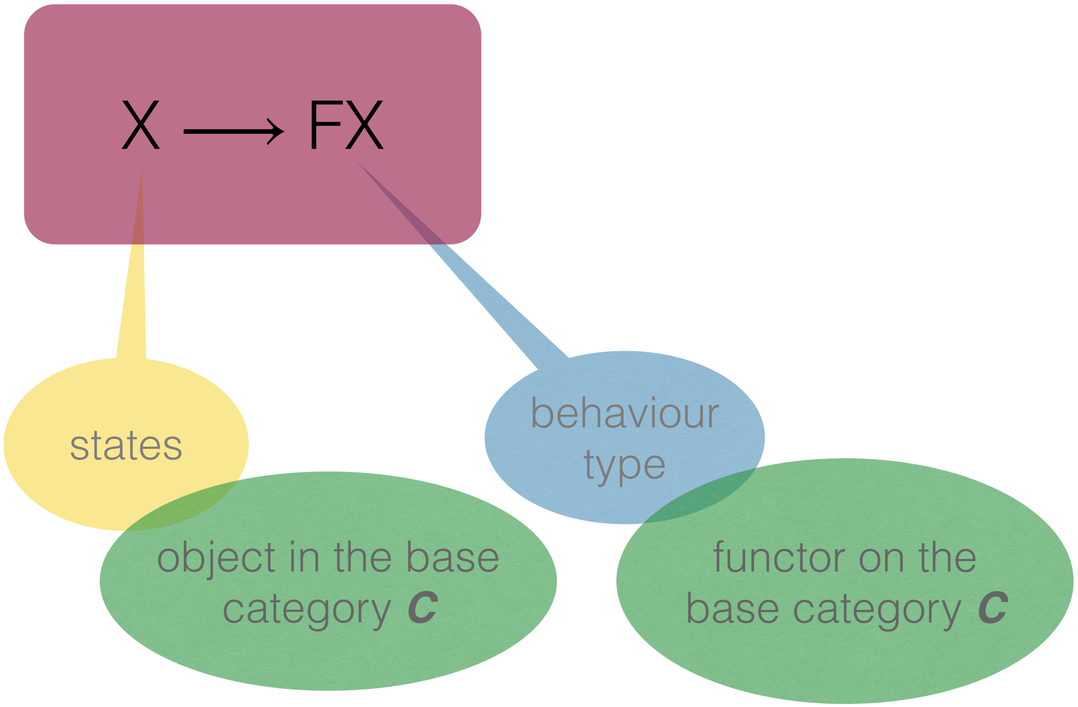
A coalgebra is a generic transition system:



Coalgebra

Uniform framework for dynamic transition systems, based on category theory.

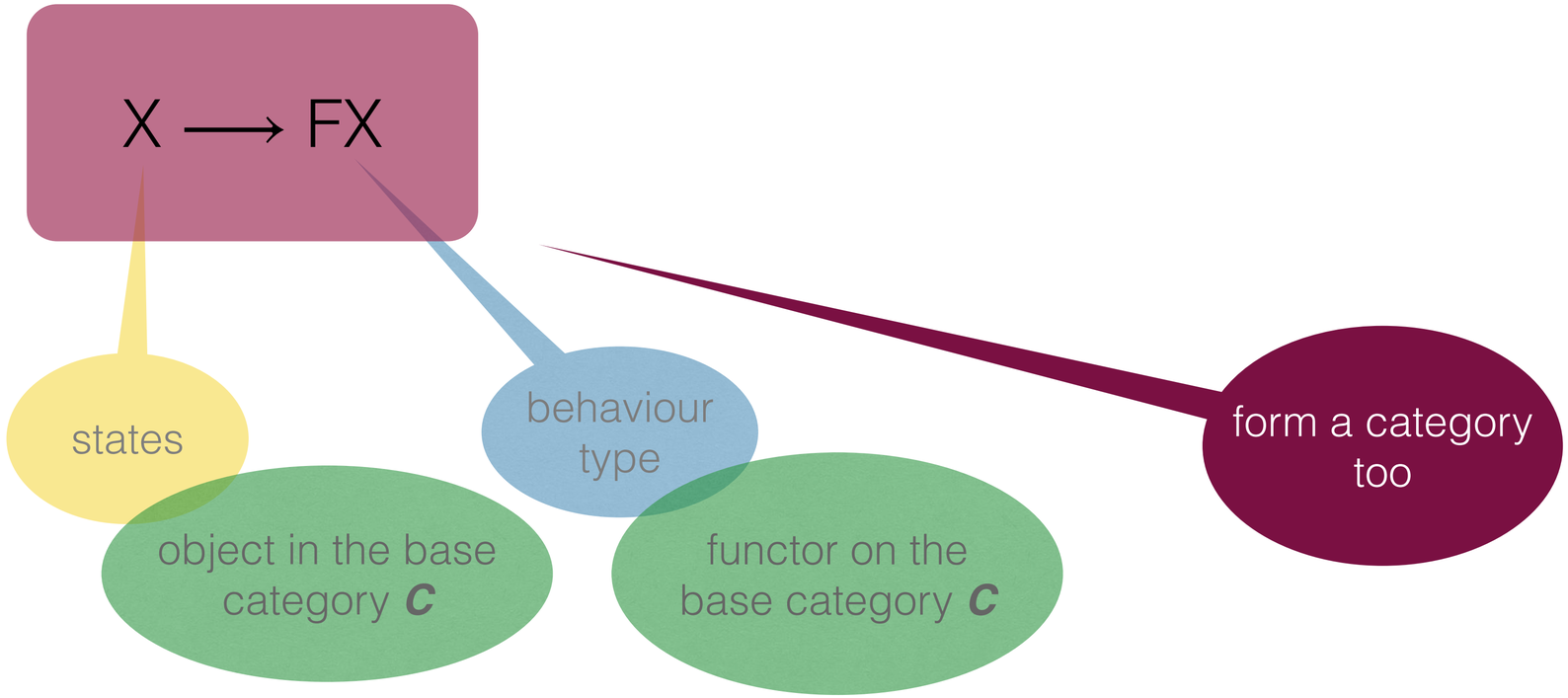
A coalgebra is a generic transition system:



Coalgebra

Uniform framework for dynamic transition systems, based on category theory.

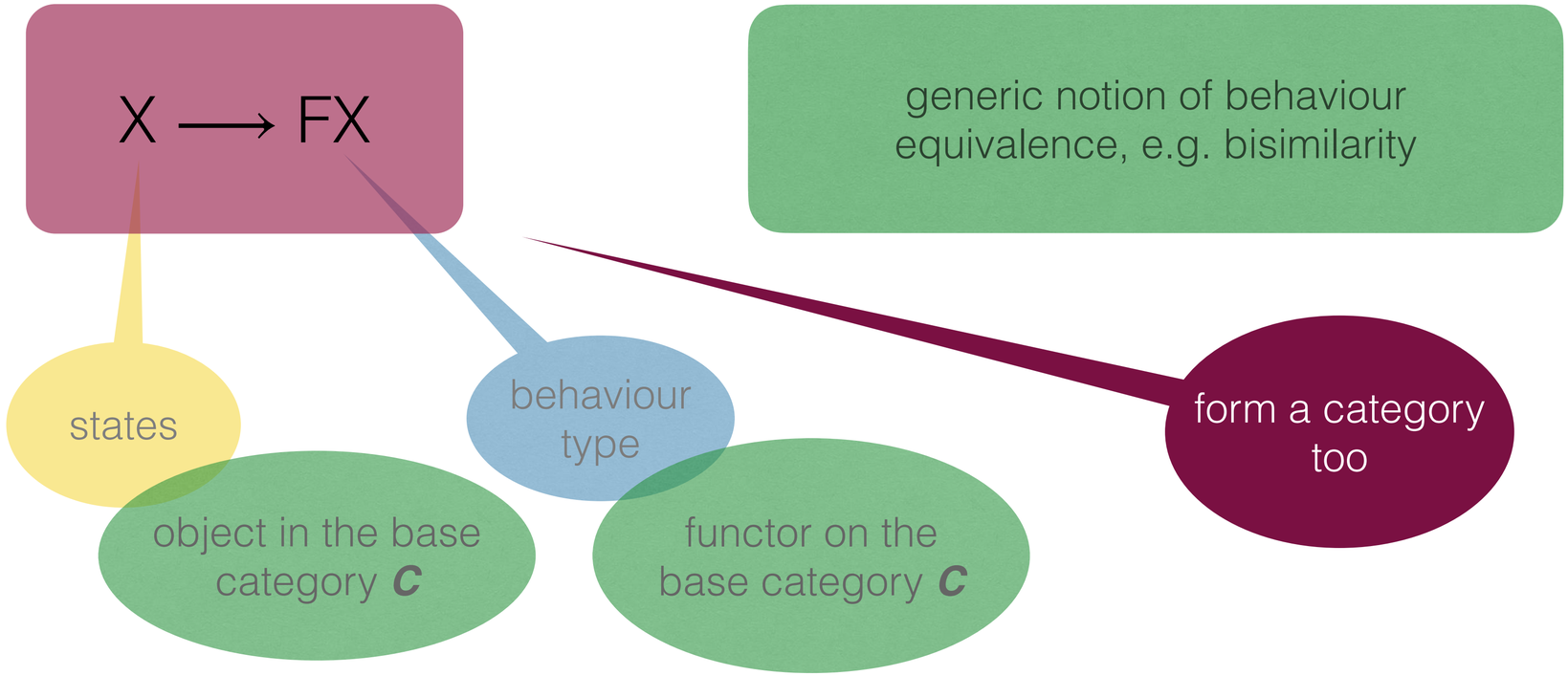
A coalgebra is a generic transition system:



Coalgebra

Uniform framework for dynamic transition systems, based on category theory.

A coalgebra is a generic transition system:



Bisimilarity of Labelled Transition Systems

$S \to (\mathcal{P}\, S)^A$

Bisimilarity of Labelled Transition Systems

An LTS is a map $S \to (\mathcal{P}\, S)^A$

with a set of states S and a set of actions A.

A binary relation R between the states of two LTS is a bisimulation

iff whenever two states are related, each transition from one of them

has a matching transition from the other and the target states are related again.

Two states are bisimilar iff they are related by a bisimulation.

Bisimilarity of Labelled Transition Systems

An LTS is a map $S \to (\mathcal{P}\, S)^A$ a coalgebra

with a set of states S and a set of actions A.

A binary relation R between the states of two LTS is a bisimulation

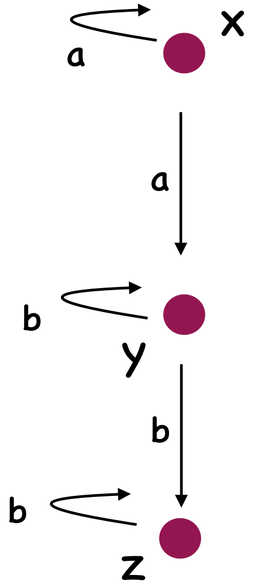
iff whenever two states are related, each transition from one of them

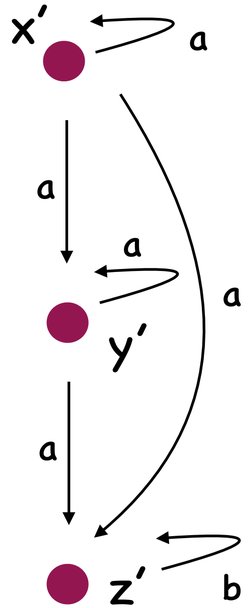
has a matching transition from the other and the target states are related again.

Two states are bisimilar iff they are related by a bisimulation.

Bisimilarity of Labelled Transition Systems

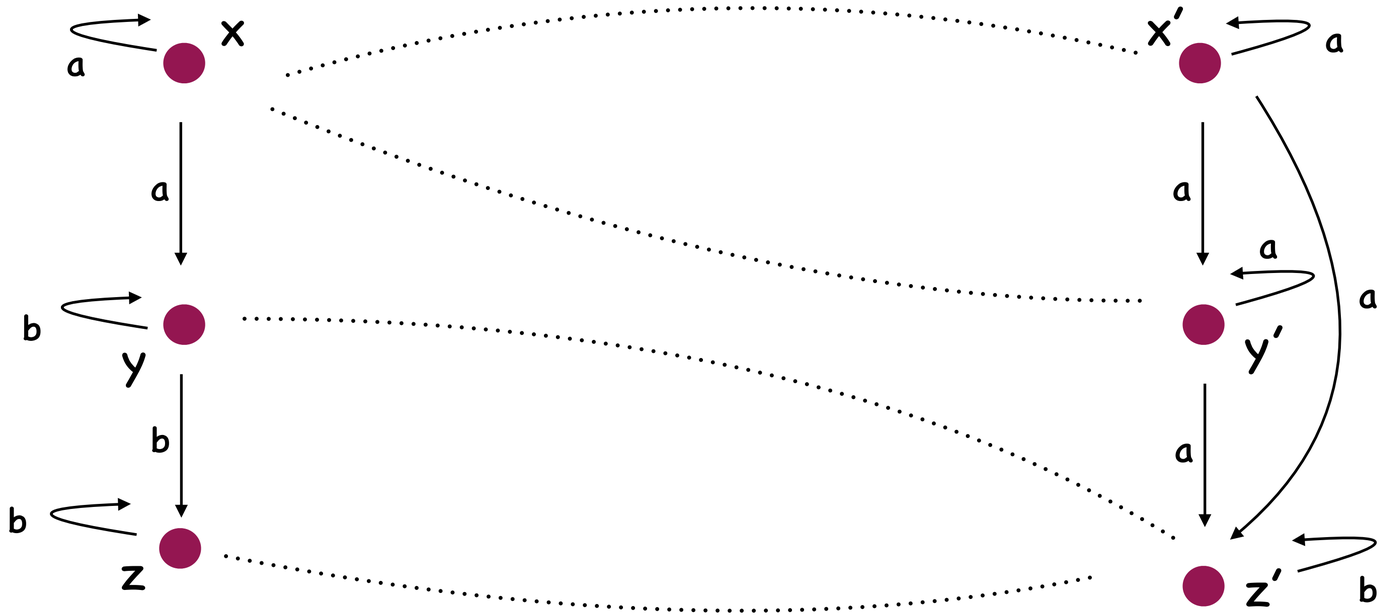
$S \to (\mathcal{P}\, S)^A$





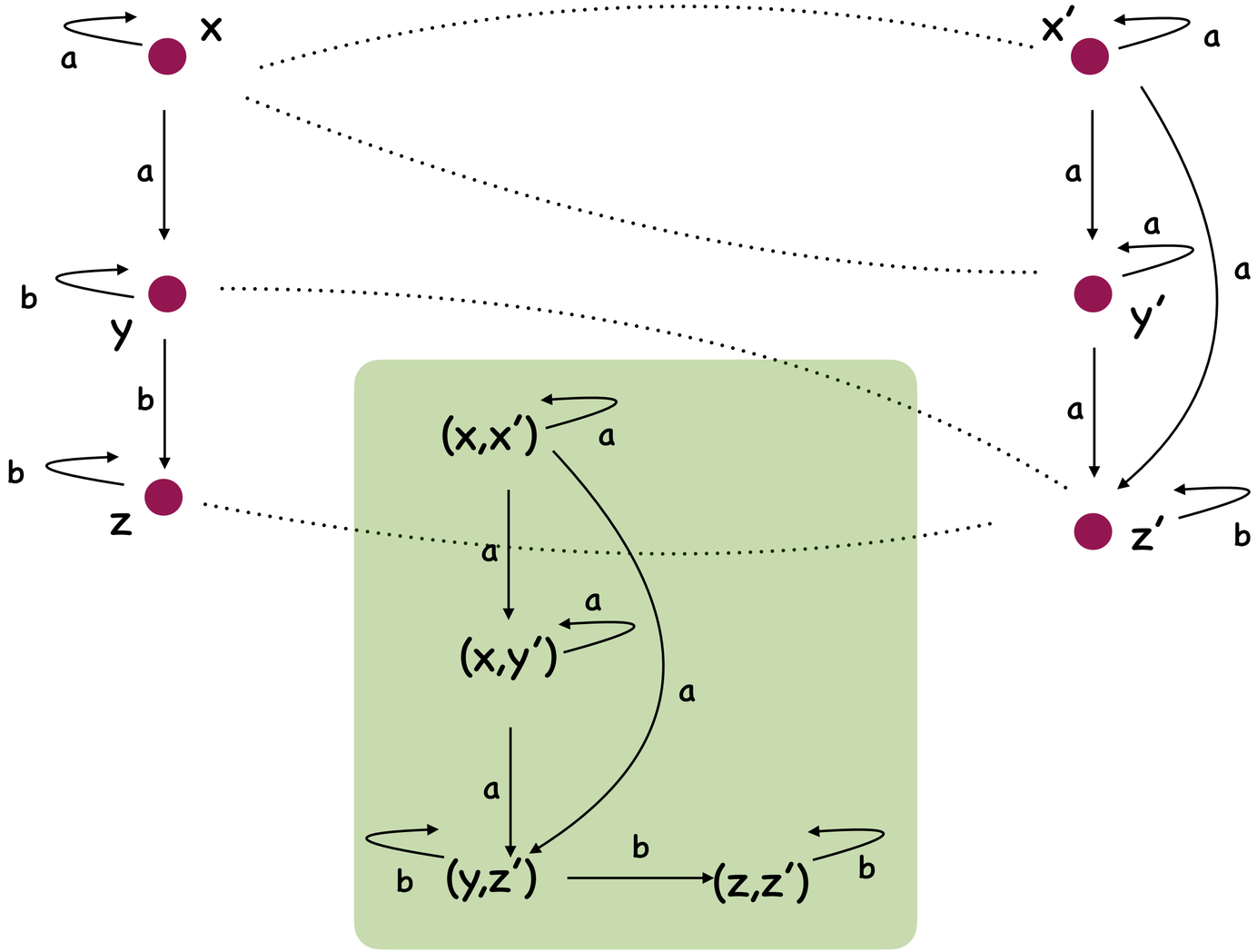
Bisimilarity of Labelled Transition Systems

$S \to (\mathcal{P}\, S)^A$



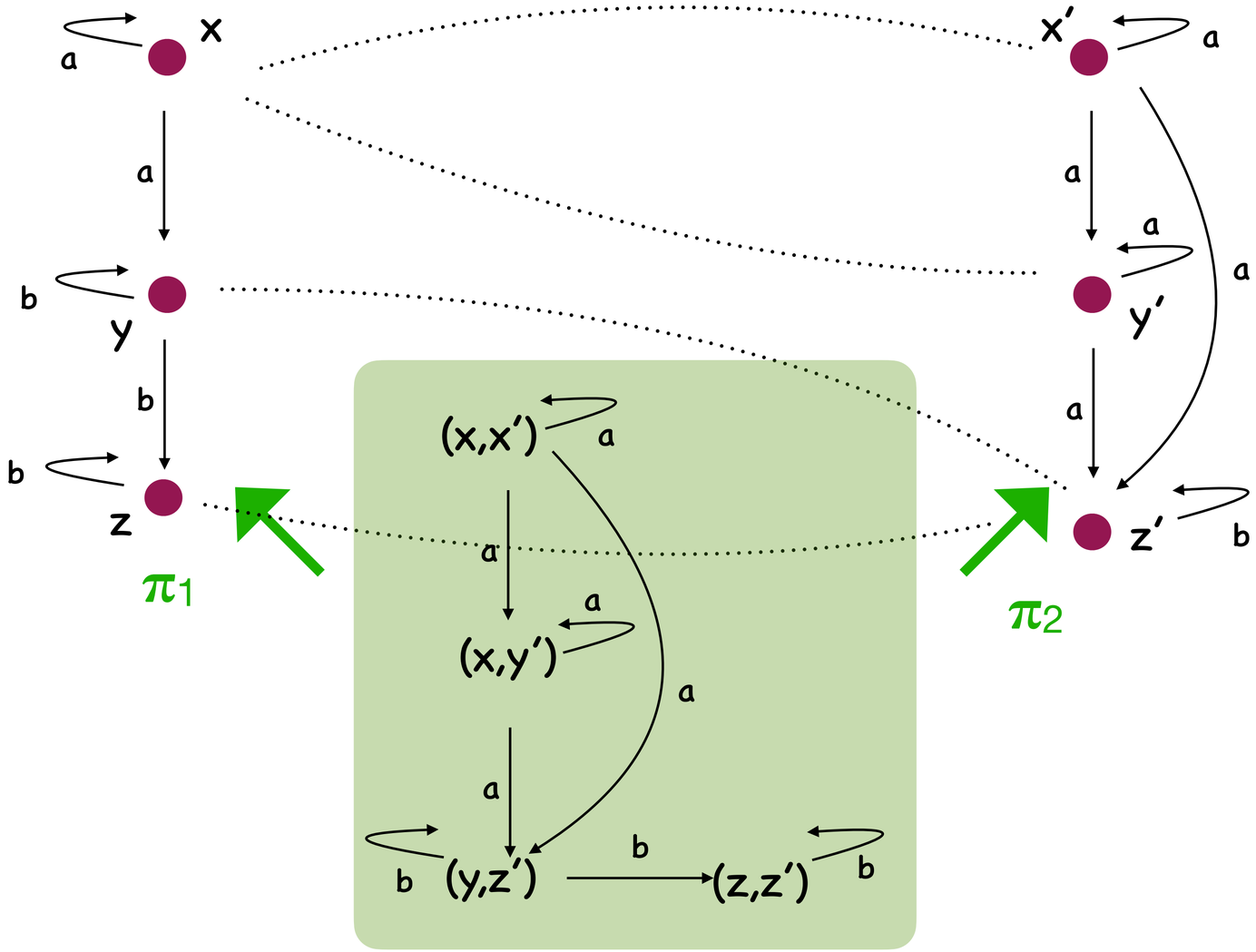
Bisimilarity of Labelled Transition Systems

$S \to (\mathcal{P}\, S)^A$



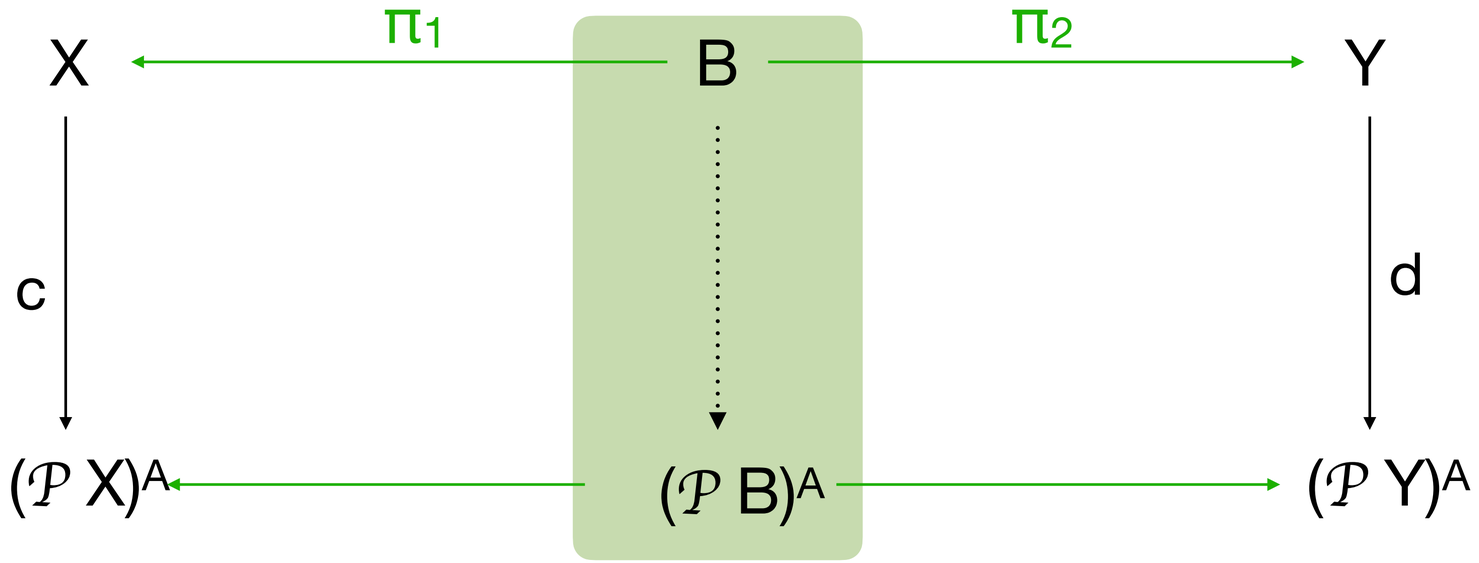
Bisimilarity of Labelled Transition Systems

$S \to (\mathcal{P}\, S)^A$



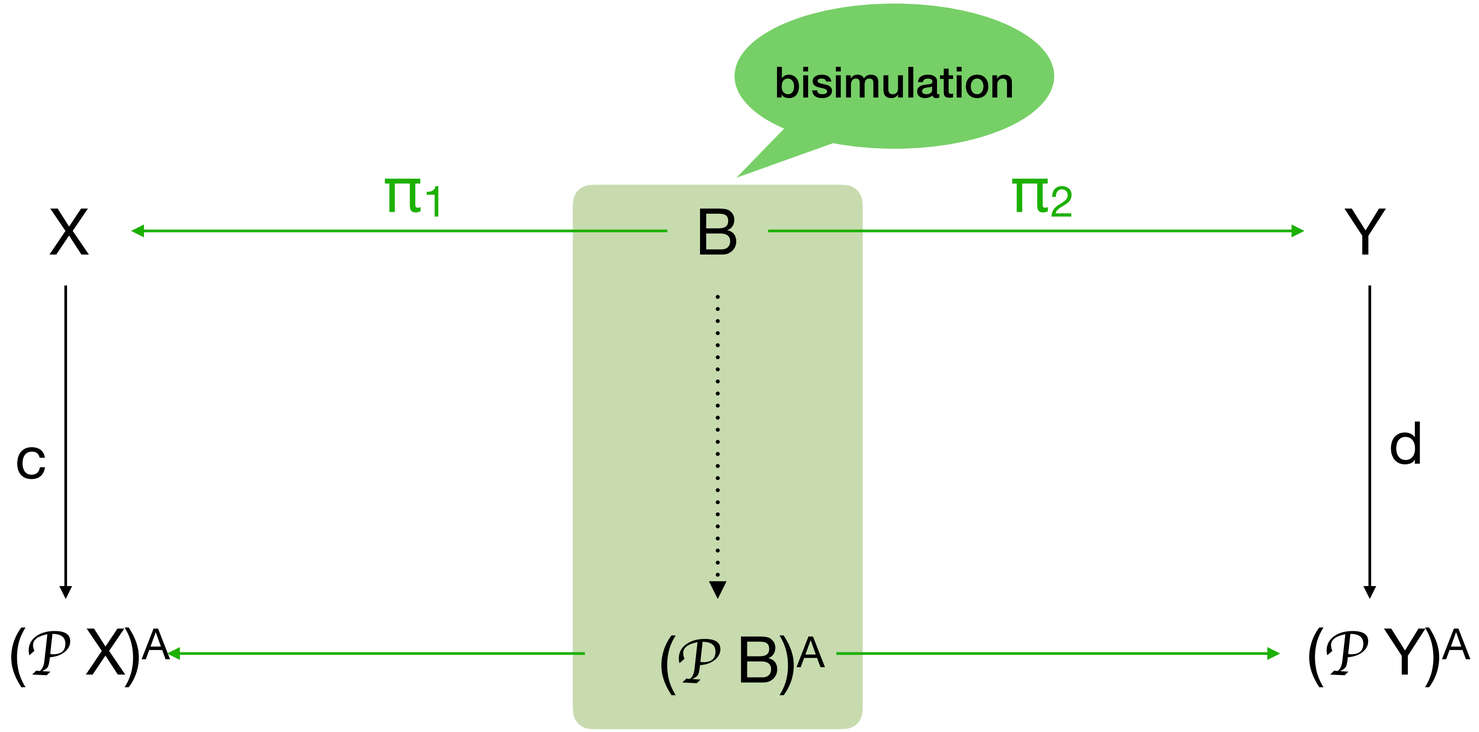
Bisimilarity of Labelled Transition Systems

$S \to (\mathcal{P}\, S)^A$



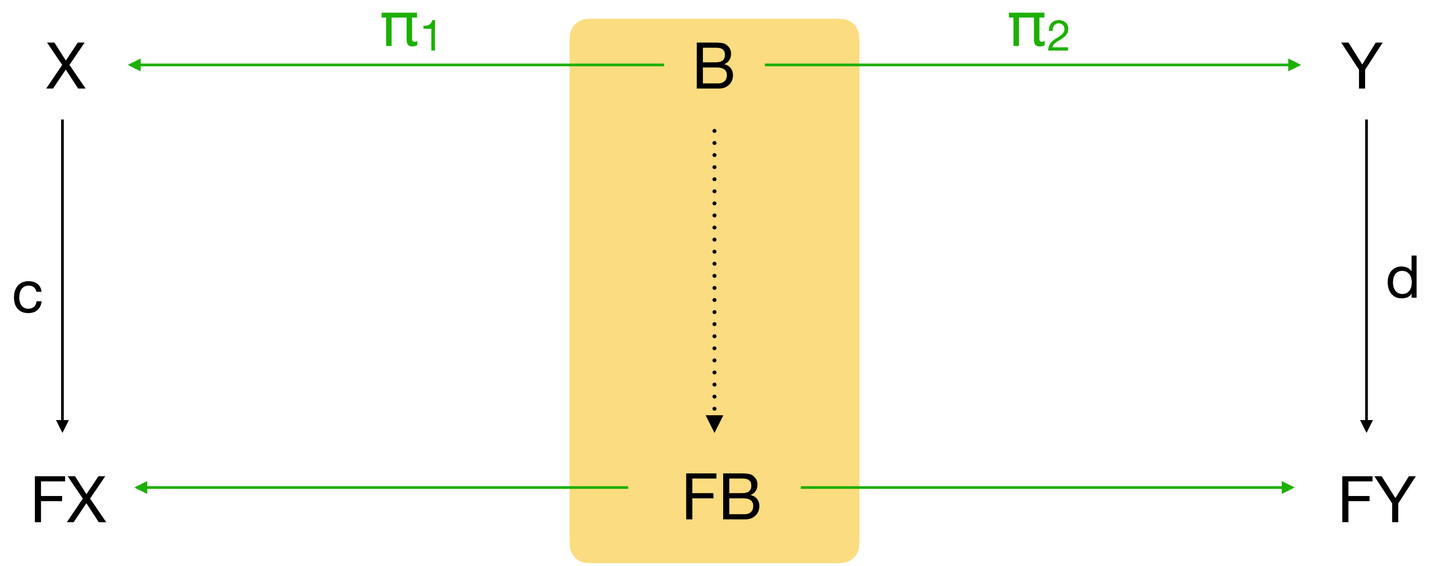
Bisimilarity of Labelled Transition Systems

$S \to (\mathcal{P}\, S)^A$

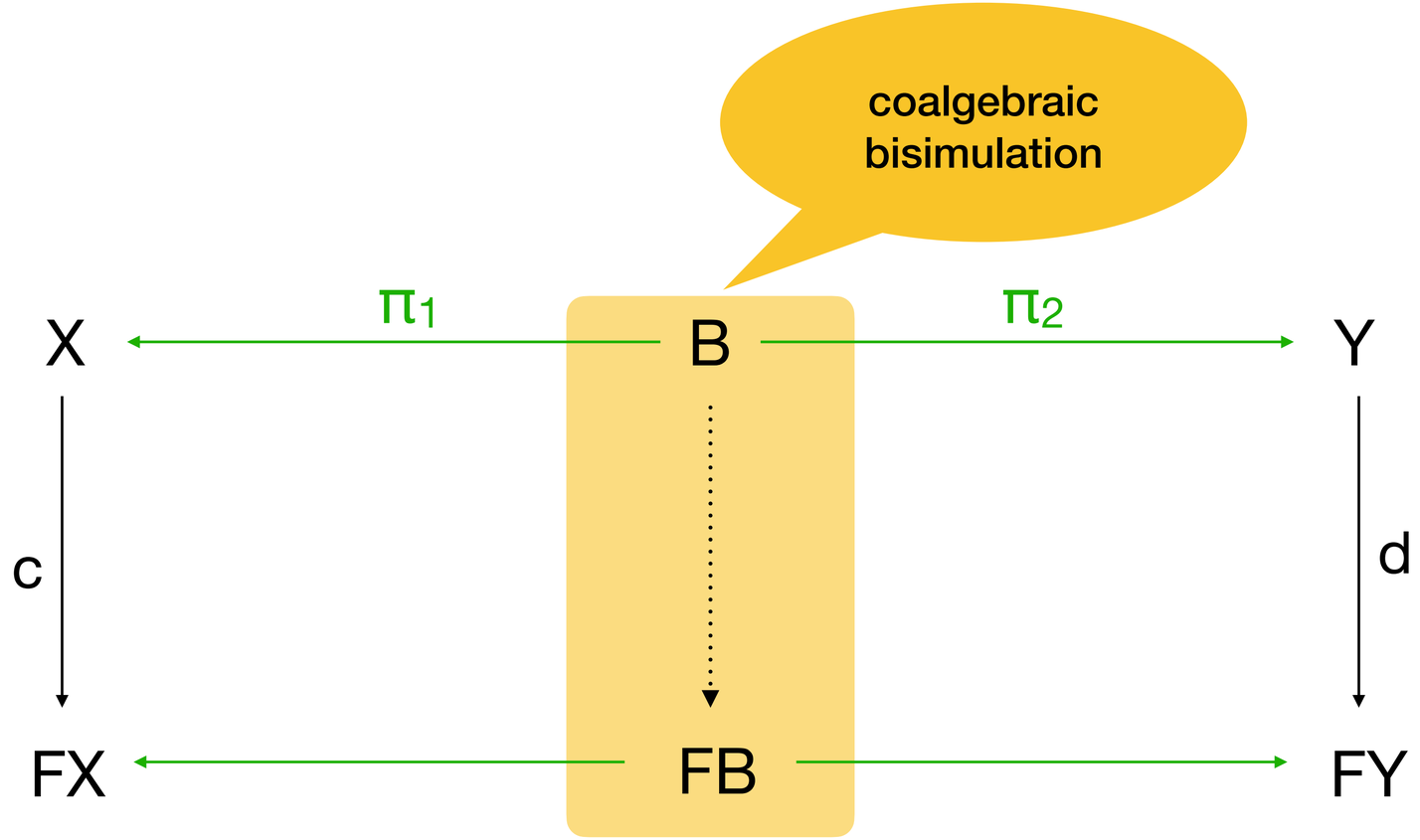


Bisimilarity of F-Coalgebras

S → FS

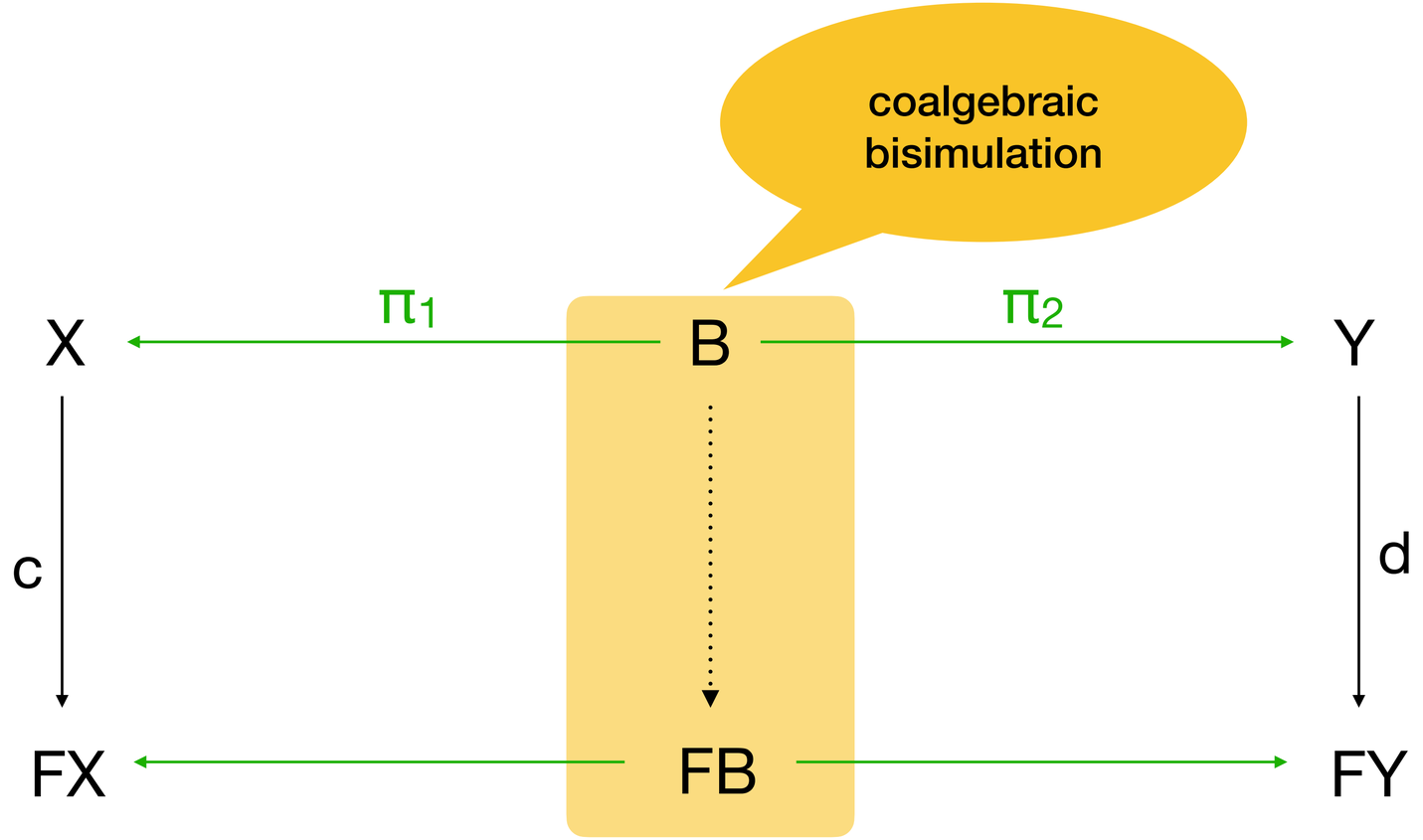


Bisimilarity of F-Coalgebras



S→ FS

Bisimilarity of F-Coalgebras



S→ FS

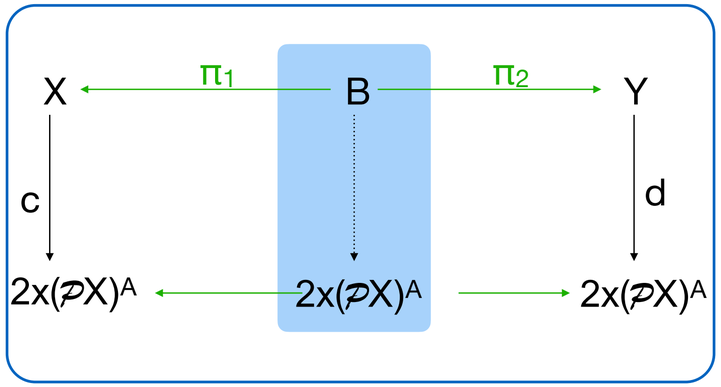
Aczel & Mendler 1989

Coalgebraic bisimilarity is bisimilarity

Coalgebraic bisimilarity is bisimilarity

NFA

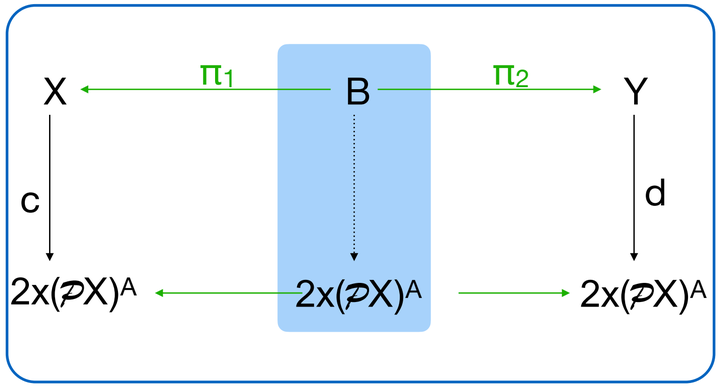
$X \to 2 \times (\mathcal{P}X)^A$



Coalgebraic bisimilarity is bisimilarity

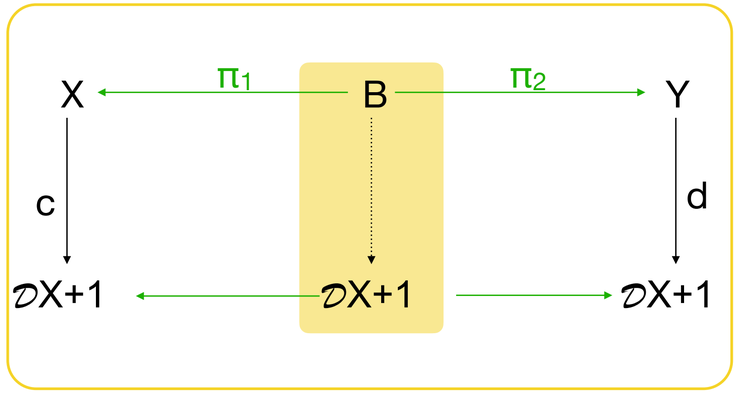
NFA

$X \to 2 \times (\mathcal{P}X)^A$



MC

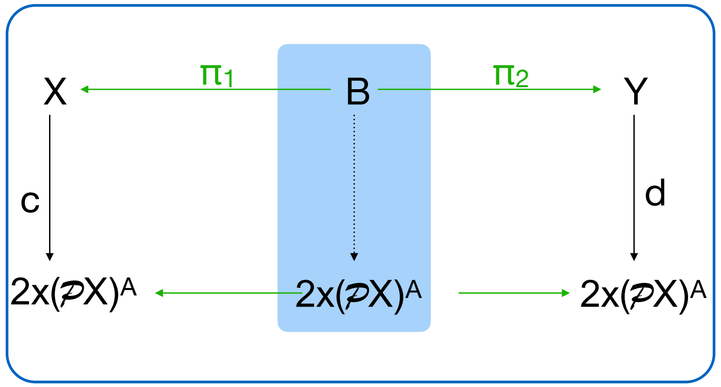
$X \to \mathcal{D}X + 1$



Coalgebraic bisimilarity is bisimilarity

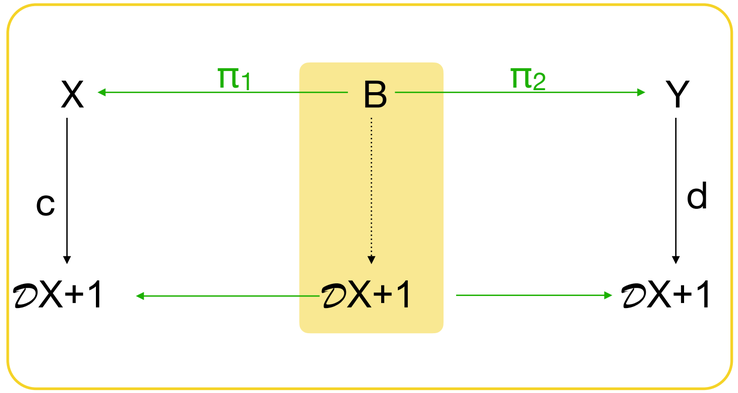
NFA

$X \to 2 \times (\mathcal{P}X)^A$



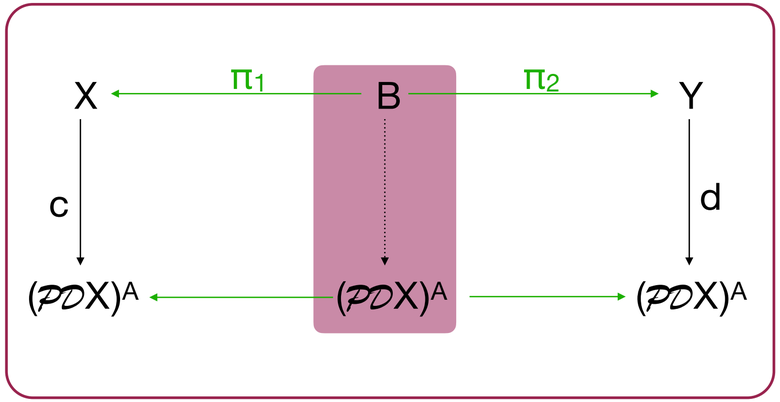
MC

$X \to \mathcal{D}X + 1$



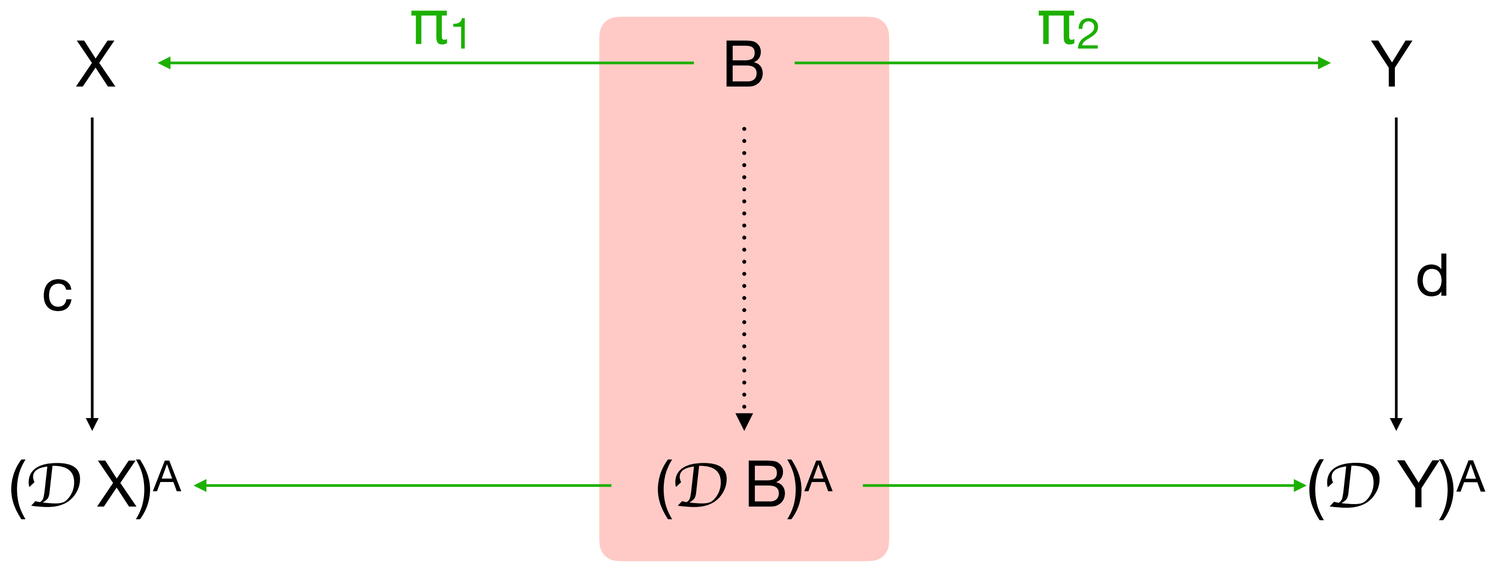
PA

$X \to (\mathcal{P}\mathcal{D}X)^A$



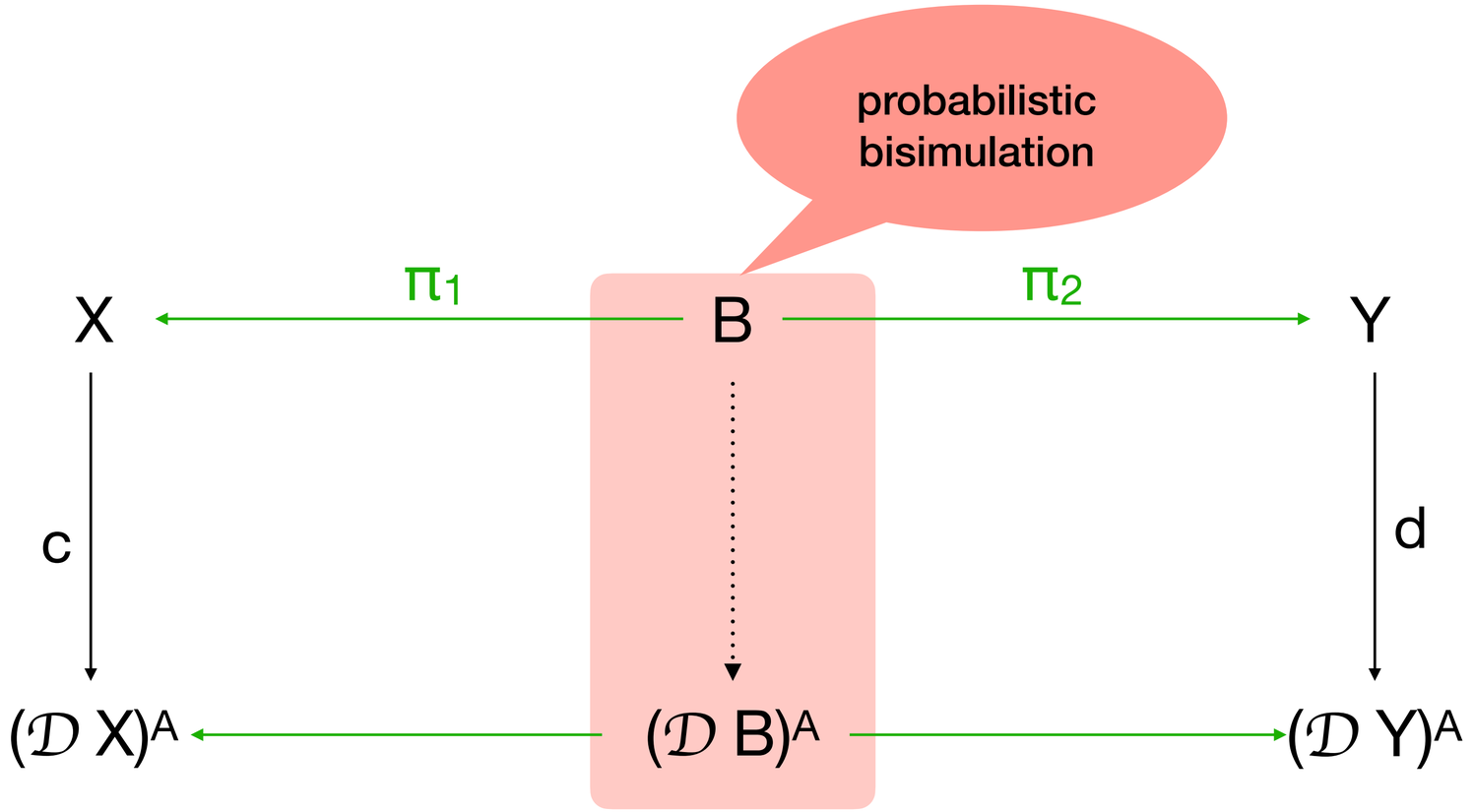
Bisimilarity of Probabilistic Transition Systems

$S \to (\mathcal{D}\, S)^A$

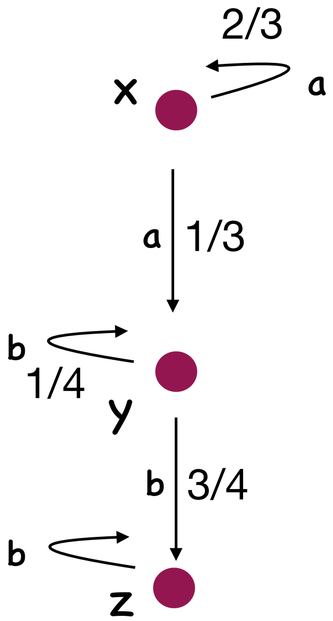


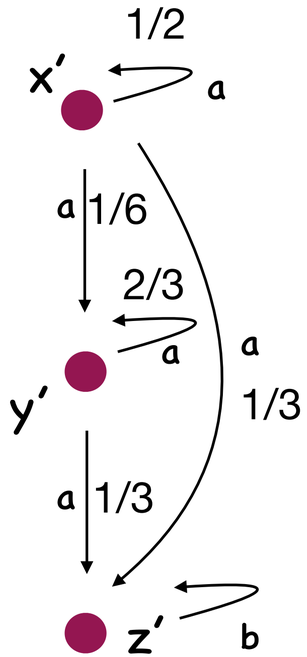
Bisimilarity of Probabilistic Transition Systems

$S \to (\mathcal{D}\, S)^A$



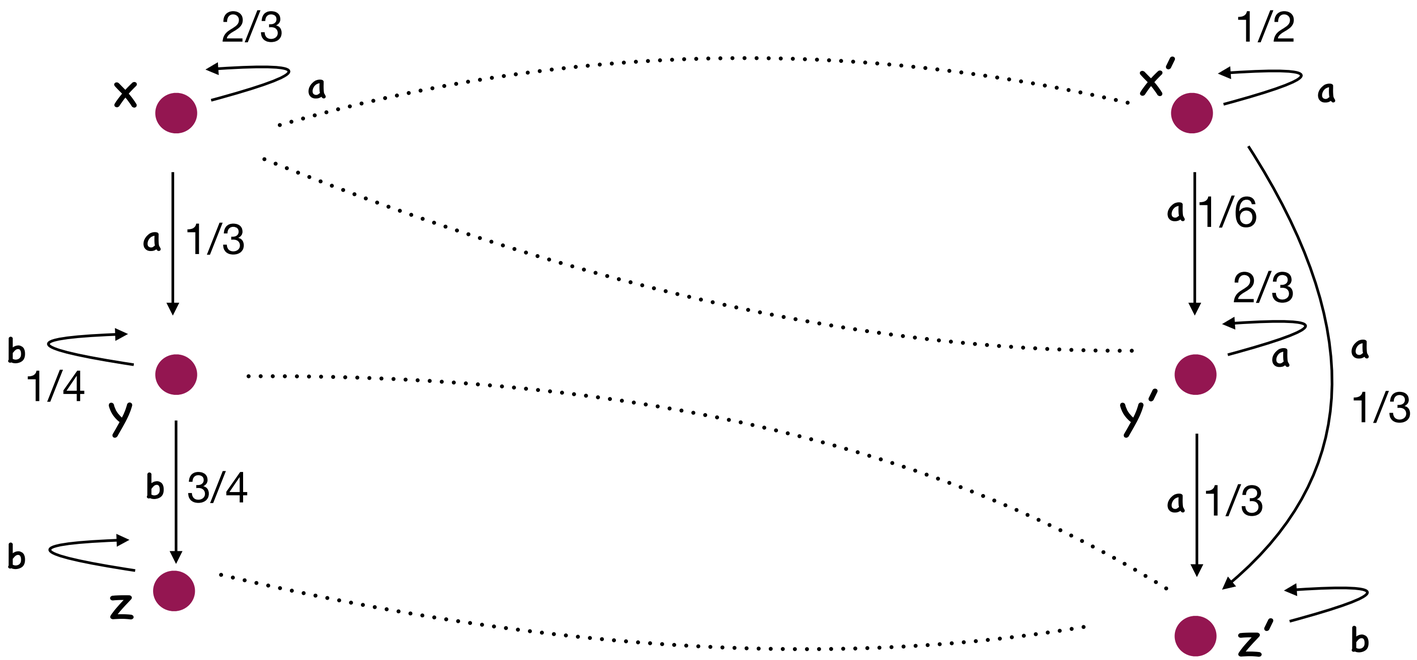
Bisimilarity of Probabilistic Transition Systems





$S \to (\mathcal{D}\, S)^A$

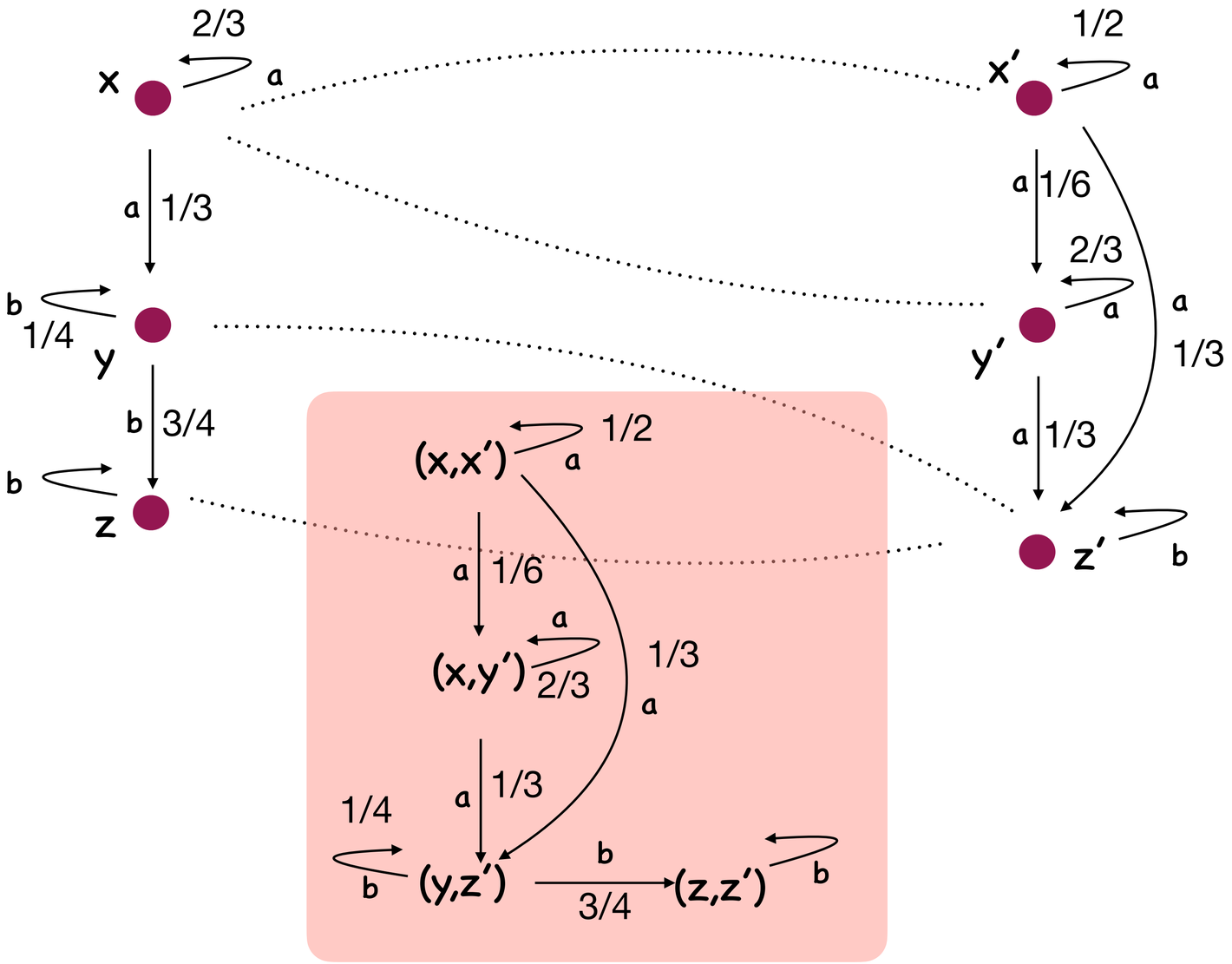
Bisimilarity of Probabilistic Transition Systems



$S \to (\mathcal{D}\, S)^A$

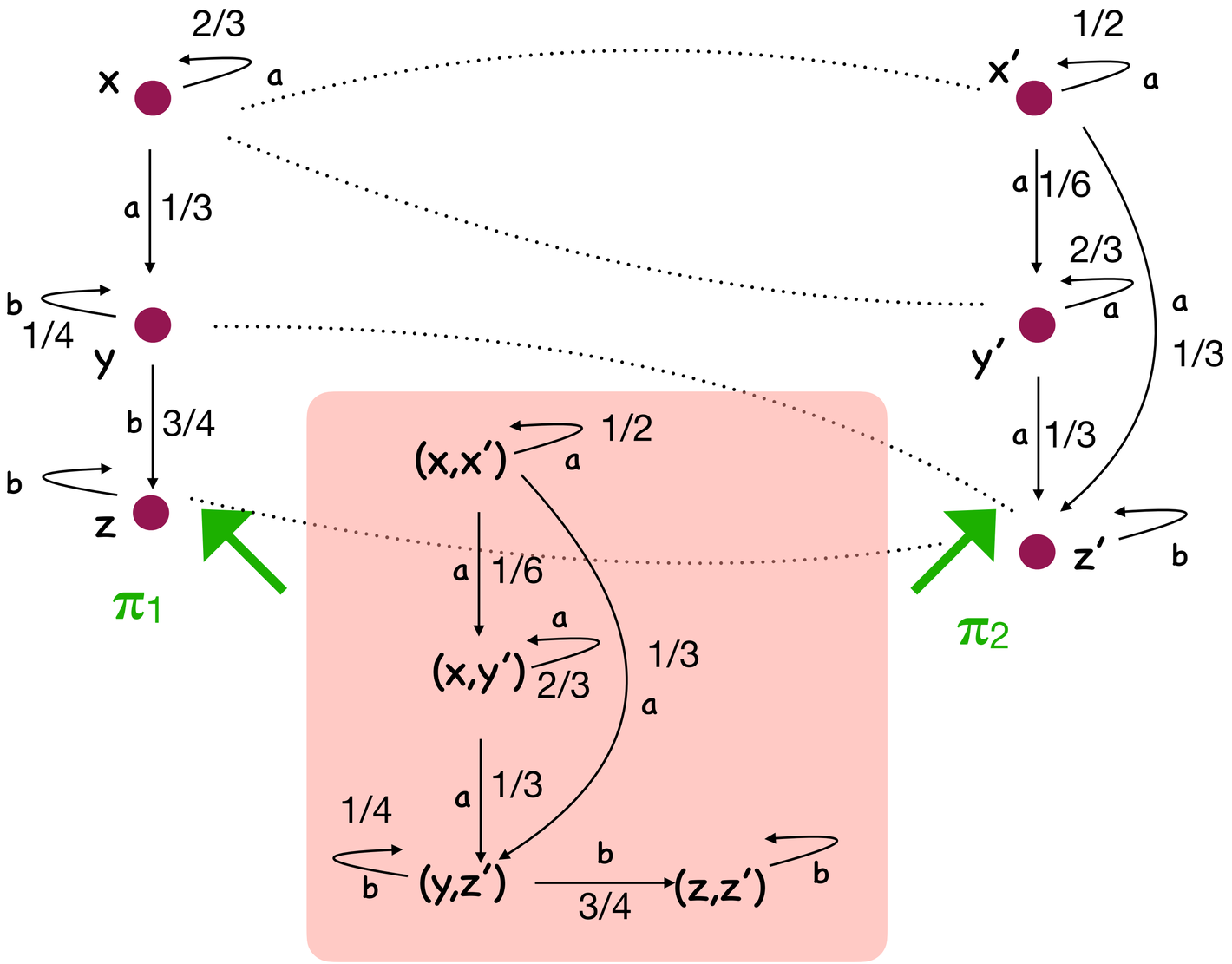
Bisimilarity of Probabilistic Transition Systems

$S \to (\mathcal{D}\, S)^A$

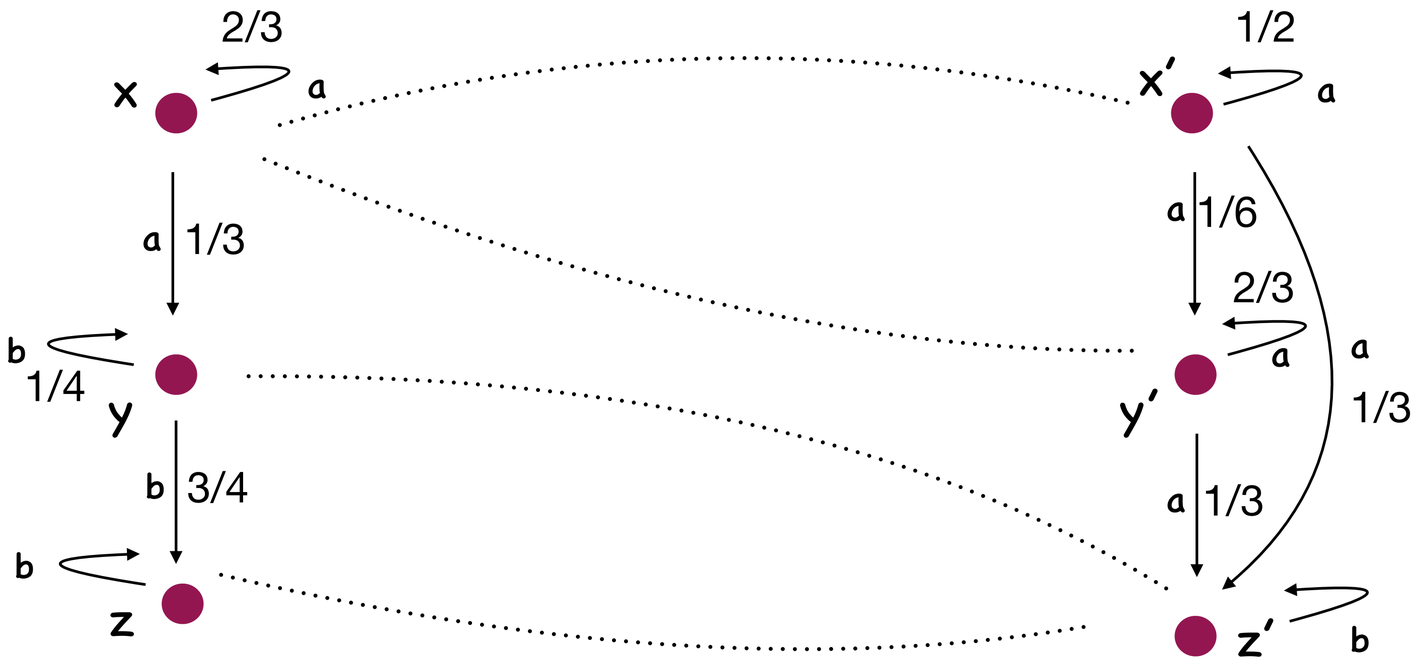


Bisimilarity of Probabilistic Transition Systems

$S \to (\mathcal{D}\, S)^A$



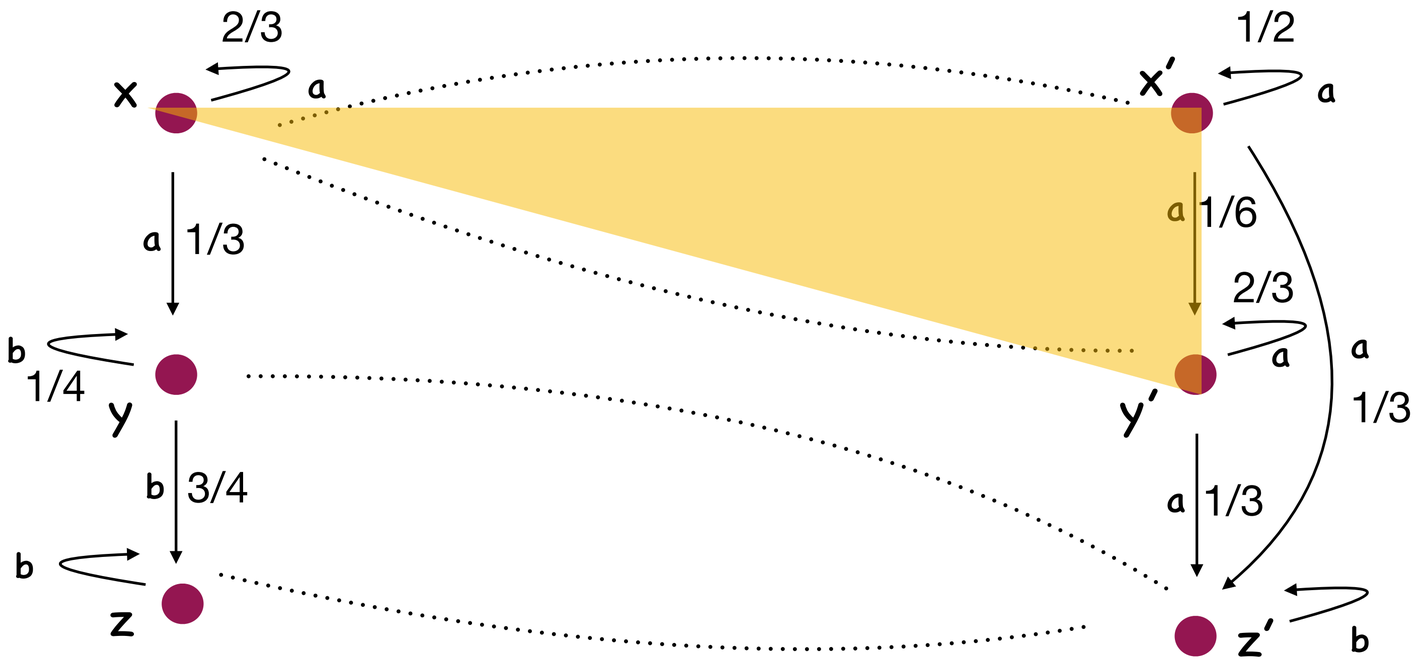
Bisimilarity of Probabilistic Transition Systems



$S \to (\mathcal{D}\, S)^A$

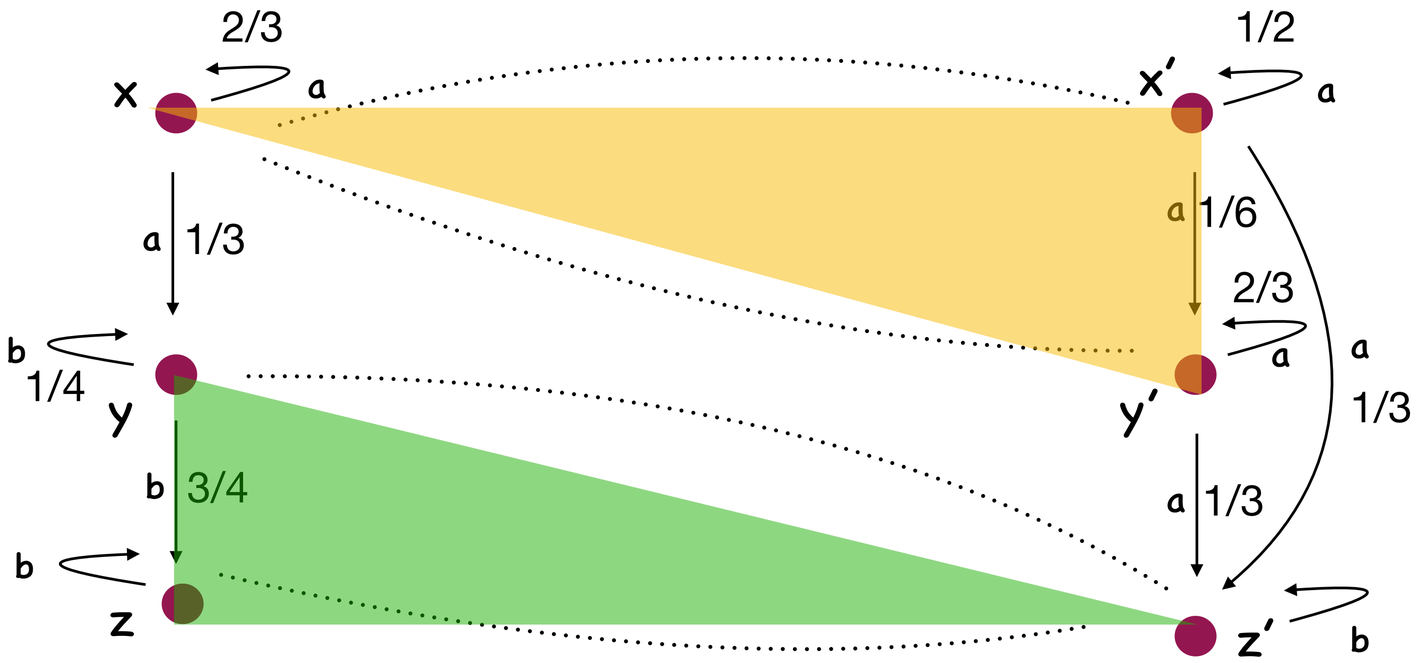
Bisimilarity of Probabilistic Transition Systems

$S \to (\mathcal{D}\, S)^A$

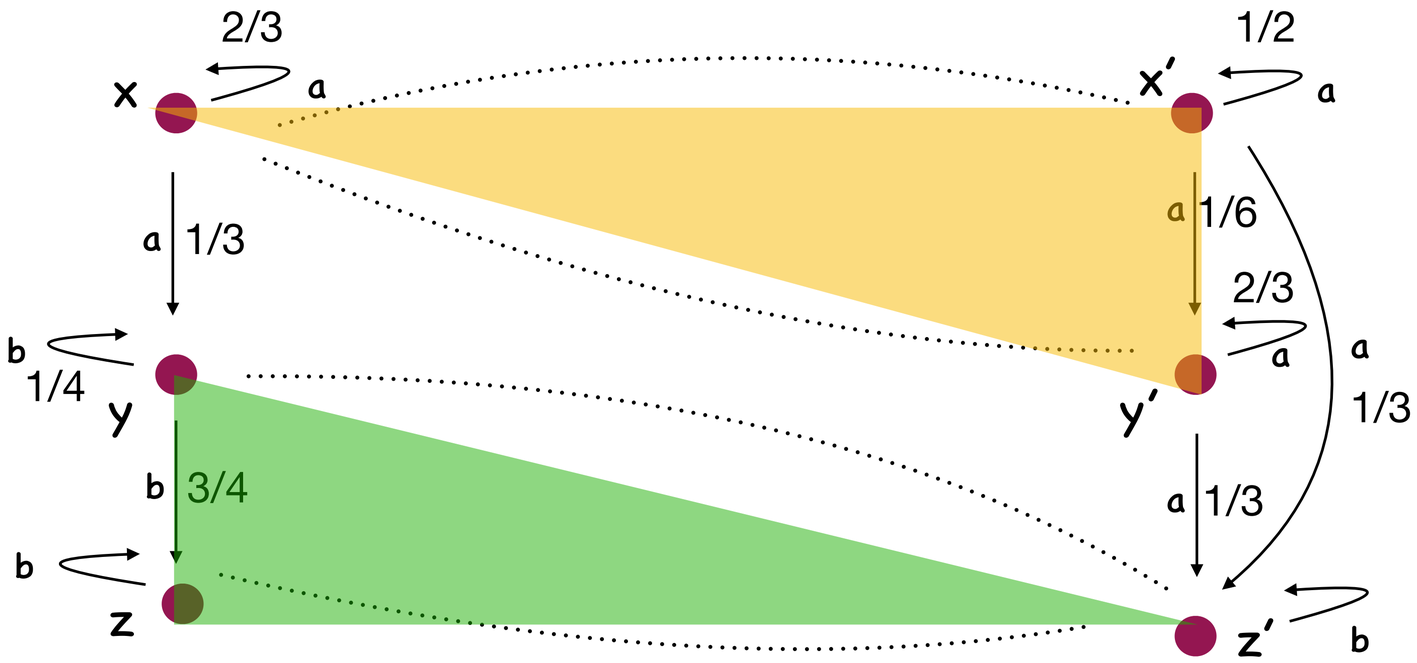


Bisimilarity of Probabilistic Transition Systems

$S \to (\mathcal{D}\, S)^A$



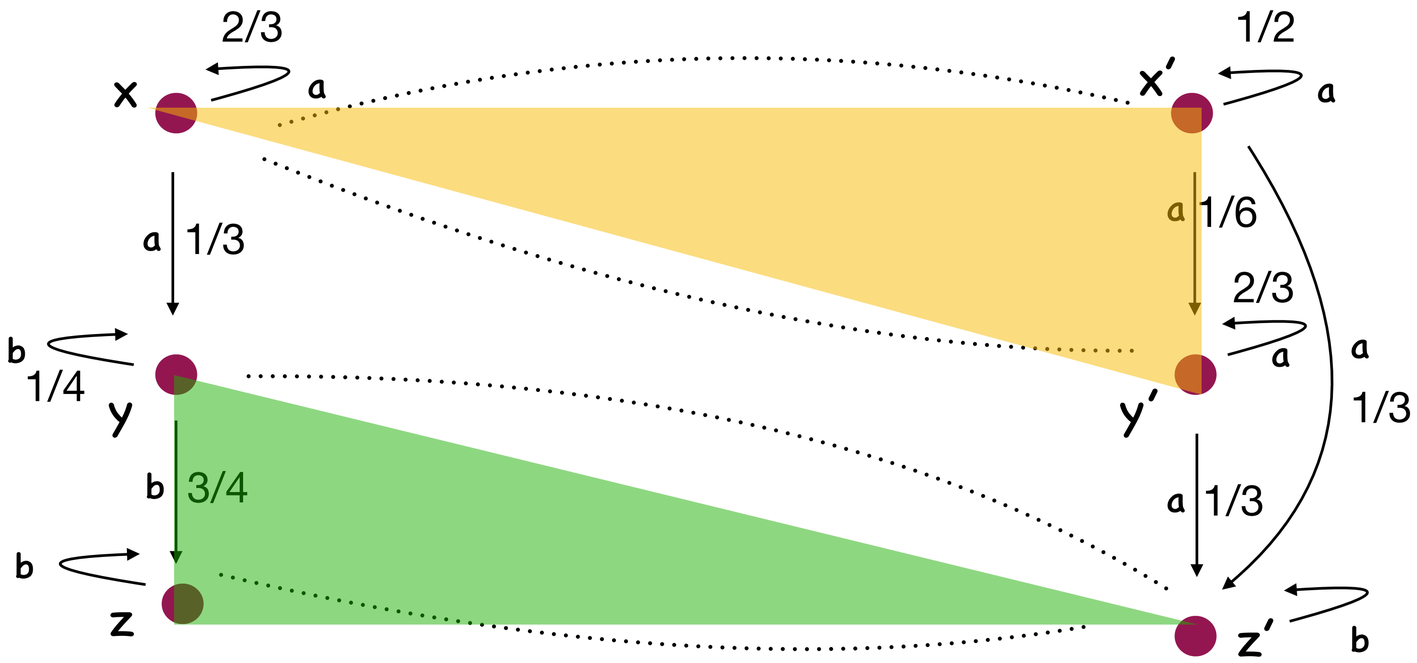
Bisimilarity of Probabilistic Transition Systems



$S \to (\mathcal{D}\, S)^A$

A (probabilistic) bisimulation is an equivalence relation on states preserving probabilities towards equivalence classes.

Bisimilarity of Probabilistic Transition Systems

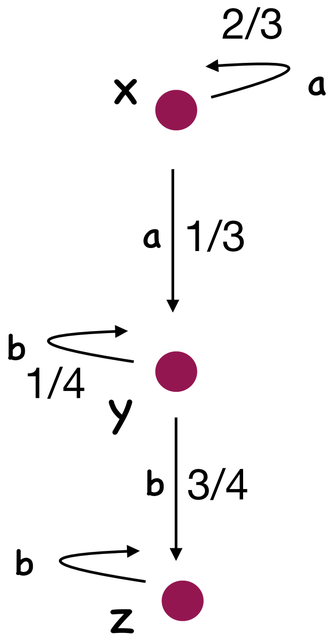


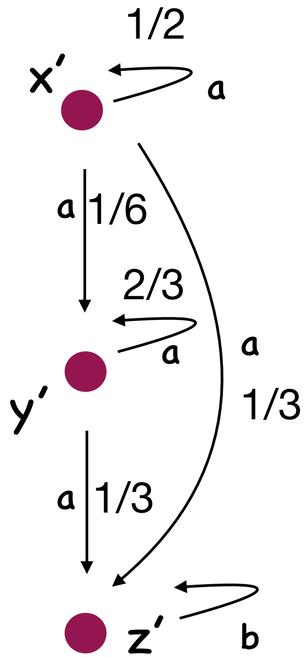
$S \to (\mathcal{D}\, S)^A$

A (probabilistic) bisimulation is an equivalence relation on states preserving probabilities towards equivalence classes.

Larsen & Skou 1991

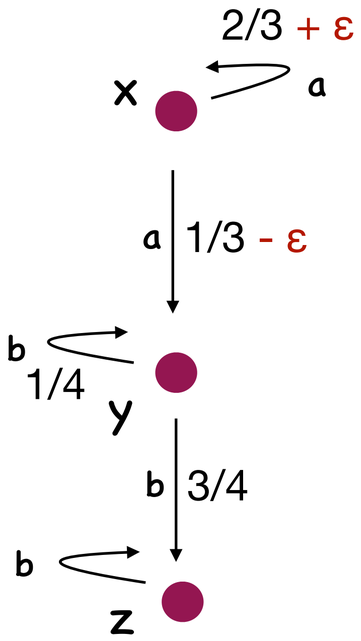
Sometimes bisimilarity is too strong / not robust

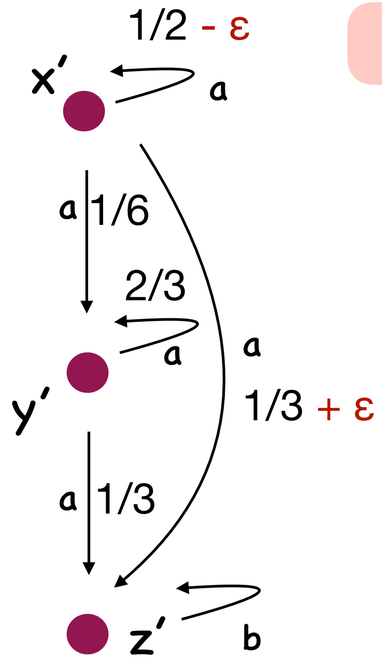




$S \to (\mathcal{D}\, S)^A$

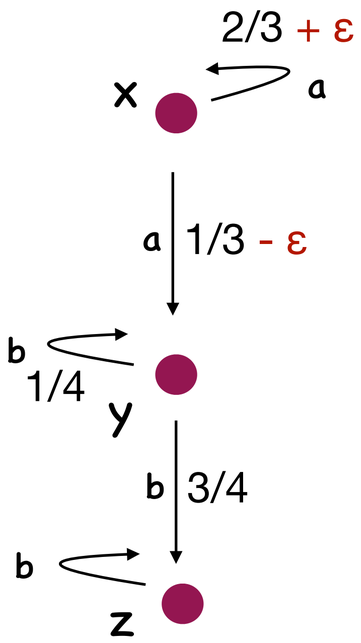
Sometimes bisimilarity is too strong / not robust

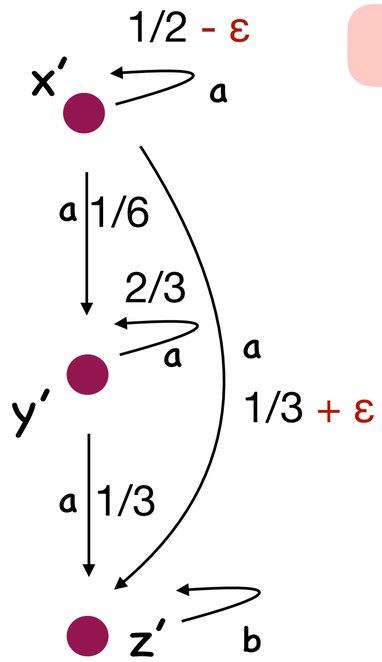




$S \to (\mathcal{D}\, S)^A$

Sometimes bisimilarity is too strong / not robust





$S \to (\mathcal{D}\, S)^A$

Realistic situation, as probabilities are often approximations of real behaviour.

Two possible solutions:

1. Behavioural pseudo metrics /distances on states, the well known d_K

2. Approximate, ε-bisimulations and ε-bisimilarity

Two possible solutions:

Desharnais & Panangaden
van Breugel & Worrell
...

1. Behavioural pseudo metrics /distances on states, the well known d_K

2. Approximate, ε-bisimulations and ε-bisimilarity

Two possible solutions:

Desharnais & Panangaden
van Breugel & Worrell
...

1. Behavioural pseudo metrics /distances on states, the well known d_K

2. Approximate, ε-bisimulations and ε-bisimilarity

...
Desharnais & Laviolette & Tracol
...

Two possible solutions:

$R_\varepsilon = \{ (x,y) \mid d_K(x,y) < \varepsilon \}$

Desharnais & Panangaden
van Breugel & Worrell
...

1. Behavioural pseudo metrics /distances on states, the well known d_K

2. Approximate, ε-bisimulations and ε-bisimilarity

...
Desharnais & Laviolette & Tracol
...

Two possible solutions:

$R_\varepsilon = \{ (x,y) \mid d_K(x,y) < \varepsilon \}$

Desharnais & Panangaden
van Breugel & Worrell
...

1. Behavioural pseudo metrics /distances on states, the well known d_K

2. Approximate, ε-bisimulations and ε-bisimilarity

$d_\varepsilon(x,y) = \inf\{ \varepsilon \mid (x,y) \text{ is in some } \varepsilon\text{-bisimulation} \}$

...
Desharnais & Laviolette & Tracol
...

The well known d_K

Kantorovich distance on distributions

1. Is the gfp of a functional $\Delta(d)(x,y) = \sup_a \delta_K (\tau_a(x), \tau_a(y))$

2. Provides a functor, even monad on .. pseudo metric spaces, has a final coalgebra.

3. Has a generic categorical lifting.

4. The monad is axiomatizable by a quantitative theory.

The well known d_K

Kantorovich distance on distributions

1. Is the gfp of a functional $\Delta(d)(x,y) = \sup_a \delta_K (\tau_a(x), \tau_a(y))$

van Breugel & Worrell

2. Provides a functor, even monad on .. pseudo metric spaces, has a final coalgebra.

3. Has a generic categorical lifting.

4. The monad is axiomatizable by a quantitative theory.

The well known d_K

Kantorovich distance on distributions

1. Is the gfp of a functional $\Delta(d)(x,y) = \sup_a \delta_K (\tau_a(x), \tau_a(y))$

van Breugel & Worrell

2. Provides a functor, even monad on .. pseudo metric spaces, has a final coalgebra.

3. Has a generic categorical lifting.

Baldan, Bonchi, König

4. The monad is axiomatizable by a quantitative theory.

The well known d_K

Kantorovich distance on distributions

1. Is the gfp of a functional $\Delta(d)(x,y) = \sup_a \delta_K (\tau_a(x), \tau_a(y))$

   van Breugel & Worrell

2. Provides a functor, even monad on .. pseudo metric spaces, has a final coalgebra.

3. Has a generic categorical lifting.

   Baldan, Bonchi, König

4. The monad is axiomatizable by a quantitative theory.

   Mio, Sarkis, Vignudelli

The well known d_K

Kantorovich distance on distributions

1. Is the gfp of a functional $\Delta(d)(x,y) = \sup_a \delta_K(\tau_a(x), \tau_a(y))$

van Breugel & Worrell

2. Provides a functor, even monad on .. pseudo metric spaces, has a final coalgebra.

3. Has a generic categorical lifting.

Baldan, Bonchi, König

4. The monad is axiomatizable by a quantitative theory.

Mio, Sarkis, Vignudelli

Has no intuitive approximate bisimulation.

ε-Bisimulation, ε-distance (one label)

...
Desharnais & Laviolette & Tracol
...

ε-Bisimulation, ε-distance (one label)

...
Desharnais & Laviolette & Tracol
...

1. A relation R is an ε-simulation iff $x\ R\ y \Rightarrow \tau(x)(X) \leq \tau(y)(R(X)) + \varepsilon$ for any subset of states X.

2. It is an ε-bisimulation if it is a symmetric ε-simulation.

3. $s \sim_\varepsilon t$ if they are related by some ε-bisimulation.

ε-Bisimulation, ε-distance (one label)

...
Desharnais & Laviolette & Tracol
...

1. A relation R is an ε-simulation iff $x\ R\ y \Rightarrow \tau(x)(X) \leq \tau(y)(R(X)) + \varepsilon$ for any subset of states X.

2. It is an ε-bisimulation if it is a symmetric ε-simulation.

3. $s \sim_\varepsilon t$ if they are related by some ε-bisimulation.

ε-bisimilarity, is a uniformity

ε-Bisimulation, ε-distance (one label)

…
Desharnais & Laviolette & Tracol
…

1. A relation R is an ε-simulation iff $x\ R\ y \Rightarrow \tau(x)(X) \leq \tau(y)(R(X)) + \varepsilon$ for any subset of states X.

2. It is an ε-bisimulation if it is a symmetric ε-simulation.

3. $s \sim_\varepsilon t$ if they are related by some ε-bisimulation.

ε-bisimilarity, is a uniformity

The ε-distance is defined by $d_\varepsilon(s,t) = \inf \{ \varepsilon \mid s \sim_\varepsilon t \}$

ε-Bisimulation, ε-distance (one label)

…
Desharnais & Laviolette & Tracol
…

1. A relation R is an ε-simulation iff $x\ R\ y \Rightarrow \tau(x)(X) \leq \tau(y)(R(X)) + \varepsilon$ for any subset of states X.

2. It is an ε-bisimulation if it is a symmetric ε-simulation.

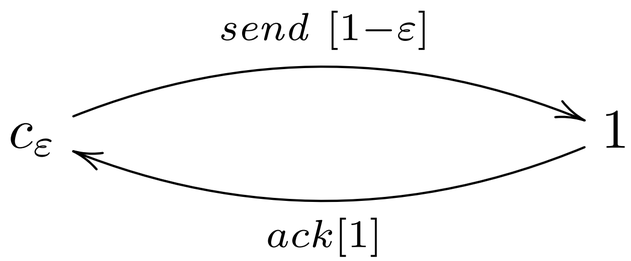
3. $s \sim_\varepsilon t$ if they are related by some ε-bisimulation.

ε-bisimilarity, is a uniformity

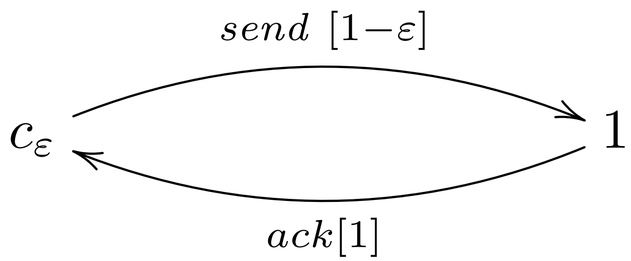
The ε-distance is defined by $d_\varepsilon(s,t) = \inf \{ \varepsilon \mid s \sim_\varepsilon t \}$

Has no fixpoint characterisation, or other general properties ?

ε-simulation, ε-bisimulation example

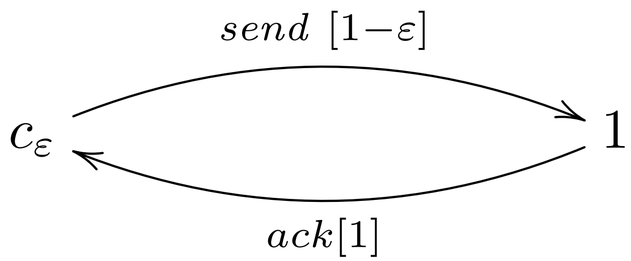


ε-simulation, ε-bisimulation example



The ε-distance between correct and approximate channel is indeed ε

ε-simulation, ε-bisimulation example

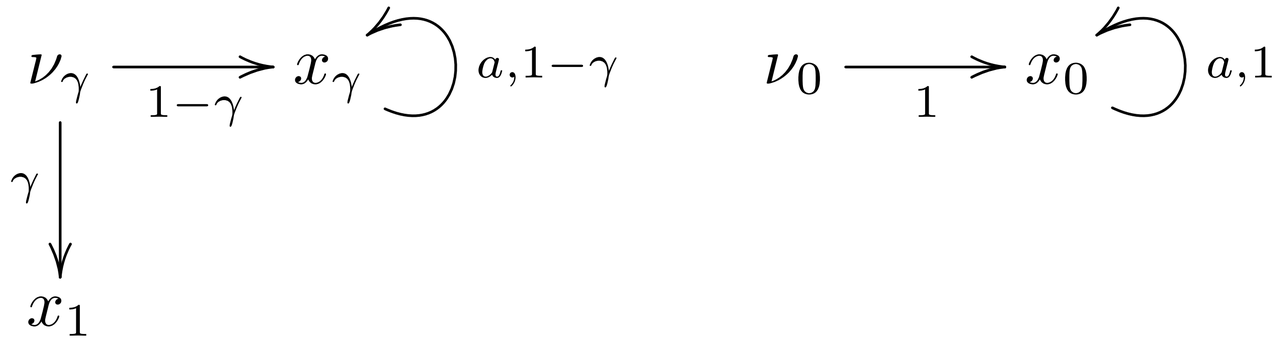


The ε-distance between correct and approximate channel is indeed ε

The K-distance between correct and approximate channel is 1

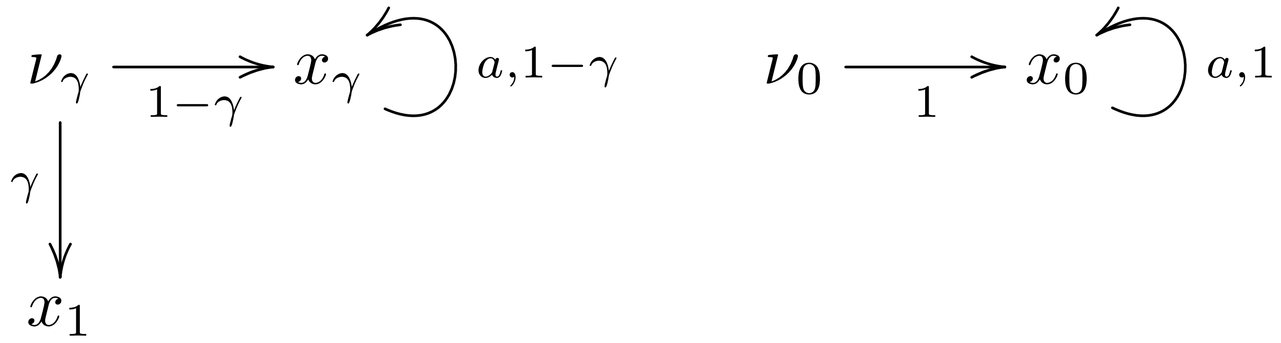
ε-simulation, ε-bisimulation example

$\gamma \in [0,1/2)$



ε-simulation, ε-bisimulation example

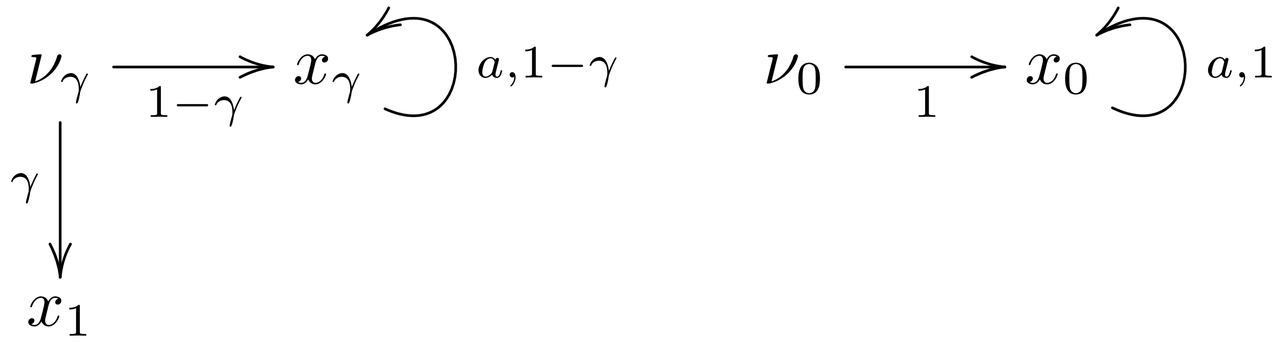
$\gamma \in [0,1/2)$



The ε-distance between ν_γ and ν_0 is γ

ε-simulation, ε-bisimulation example

$\gamma \in [0,1/2)$



The ε-distance between ν_γ and ν_0 is γ

The ε-distance between ν_γ and ν_ξ is $|\gamma-\xi|$

Our new results on ε-distance:

Josée Desharnais

Desharnais & S. 2025

Our new results on ε-distance:

Josée Desharnais

Desharnais & S. 2025

1. It is the gfp of a functional $\Delta(d)(x,y) = \sup_a \delta_{LP,d}(\tau_a(x), \tau_a(y))$
2. Provides a functor, not a monad on pseudo metric spaces.
3. ε-bisimulations and ε-bisimilarity have a coalgebraic characterisation.

Our new results on ε-distance:

Josée Desharnais

Desharnais & S. 2025

Lévy-Prokhorov distance on distributions

1. It is the gfp of a functional $\Delta(d)(x,y) = \sup_a \delta_{LP,d}(\tau_a(x), \tau_a(y))$
2. Provides a functor, not a monad on pseudo metric spaces.
3. ε-bisimulations and ε-bisimilarity have a coalgebraic characterisation.

Our new results on ε-distance:

Josée Desharnais

Desharnais & S. 2025

Lévy-Prokhorov distance on distributions

1. It is the gfp of a functional $\Delta(d)(x,y) = \sup_a \delta_{LP,d}(\tau_a(x), \tau_a(y))$
2. Provides a functor, not a monad on pseudo metric spaces.
3. ε-bisimulations and ε-bisimilarity have a coalgebraic characterisation.

Lévy-Prokhorov lifting

Starting from a metric space (S,d), it gives a distance on $\mathcal{D}$S by

$$\delta^d_{LP}(\mu_0, \mu_1) = \inf\{\varepsilon \mid \forall X \subseteq S : \mu_i(X) \leq \mu_{1-i}(X^d_\varepsilon) + \varepsilon,\ \ for\ i = 0, 1\}$$

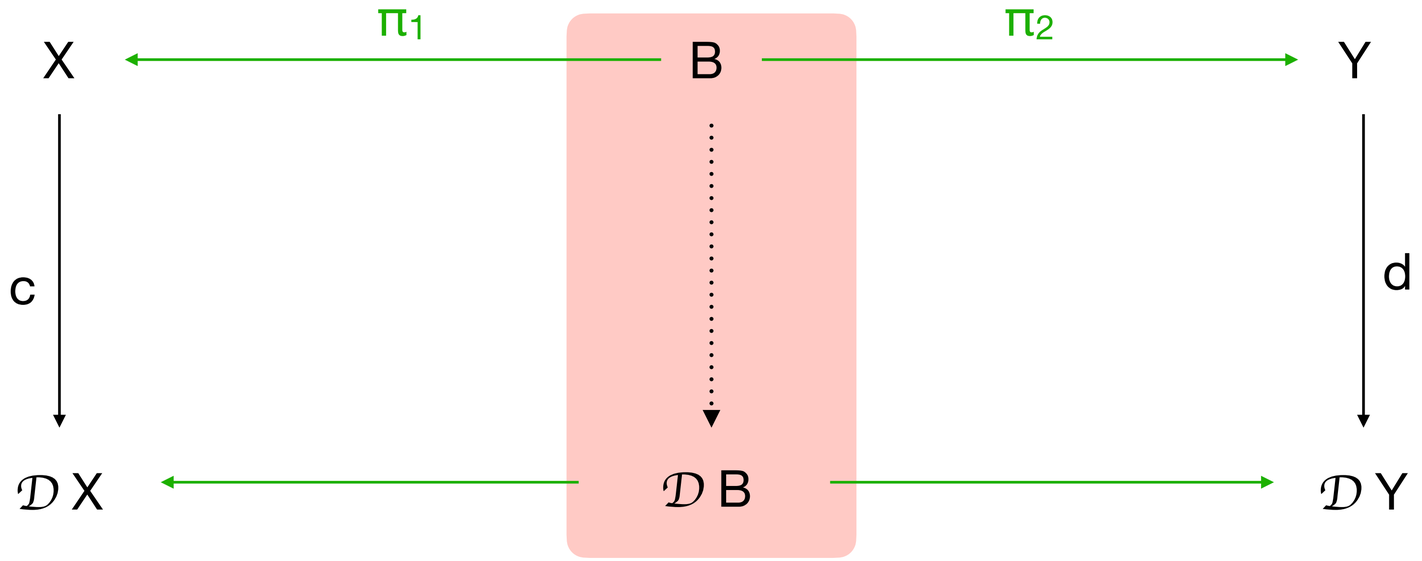
where

$$X^d_\varepsilon = \{y \mid \exists x \in X : d(x, y) < \varepsilon\}$$

... and further a behavioural distance on $S \to (\mathcal{D}\, S)^A$ **by fixpoints**

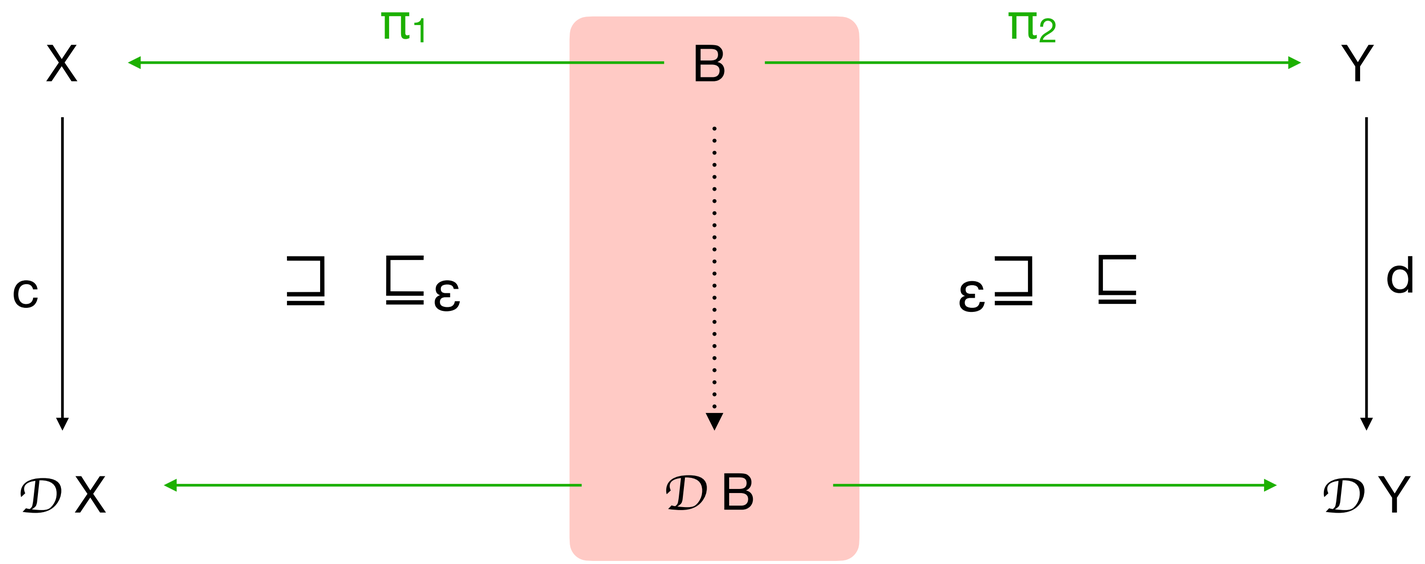
Recall bisimulation for PTS (one label)

$S \to \mathcal{D}\, S$



ε-Bisimulation in a diagram

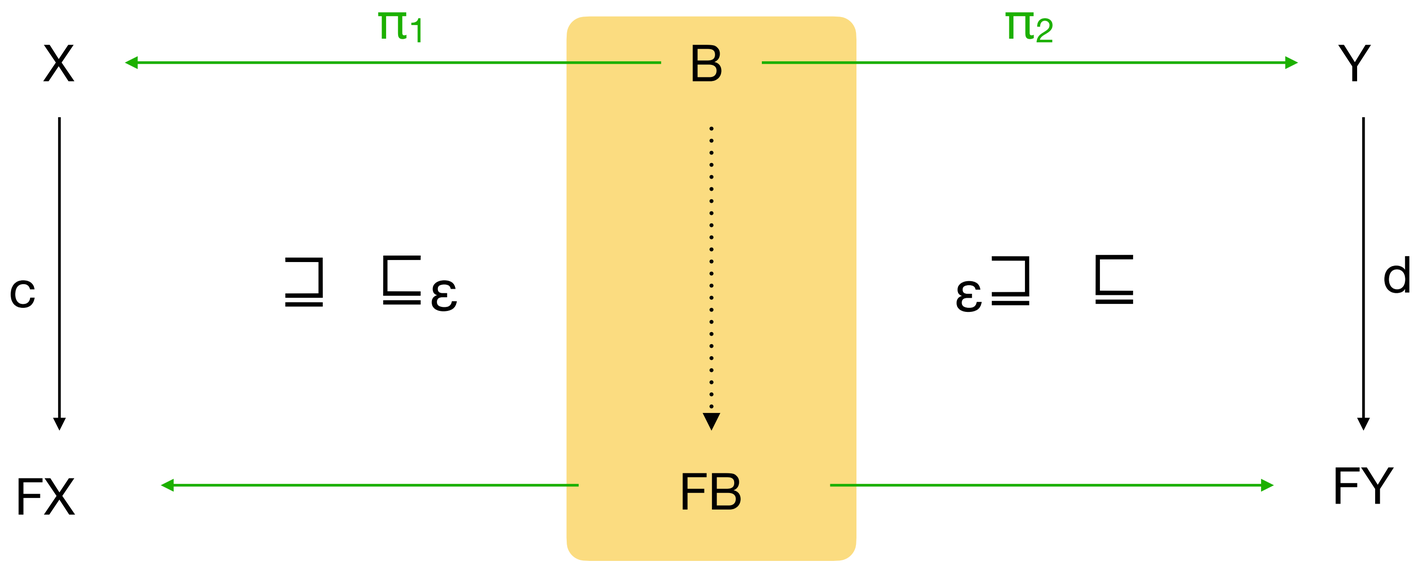
$S \rightarrow \mathcal{D}\,S$



with: $\varphi \sqsubseteq_\varepsilon \psi$ iff for all $A \subseteq B$, $\varphi(A) \leq \psi(A) + \varepsilon$

ε-Bisimulation for F-coalgebras

$S \to FS$

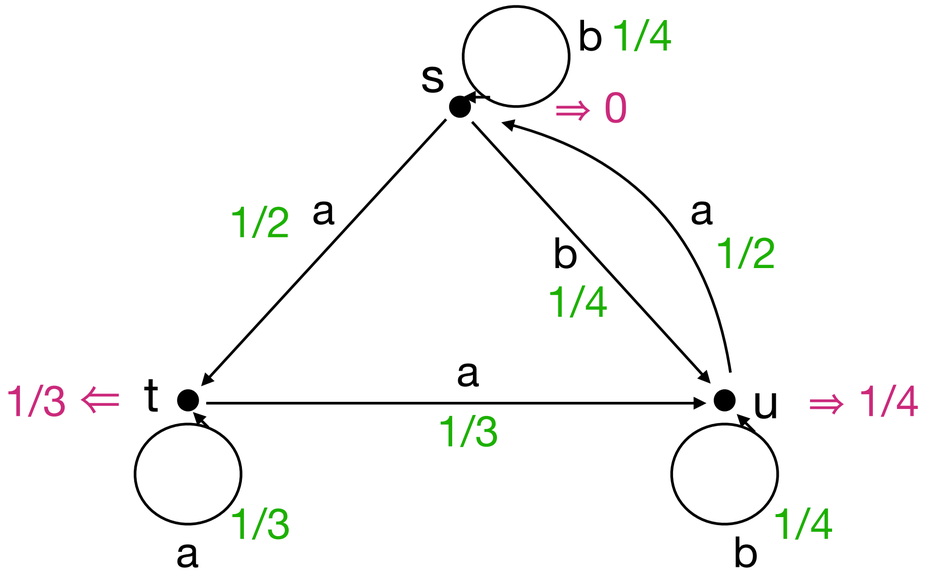


as long as F comes with a suitable order

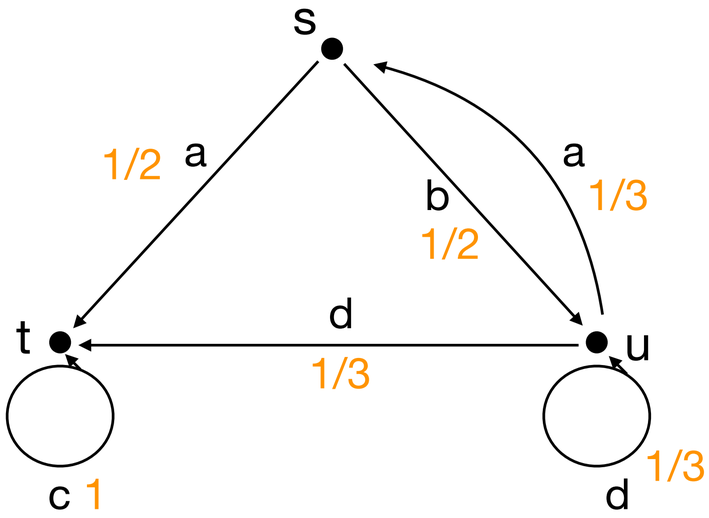
this enables new definition of approximate bisimulations for other types of systems

Trace Semantics = Bisimilarity Elsewhere

Finite Traces

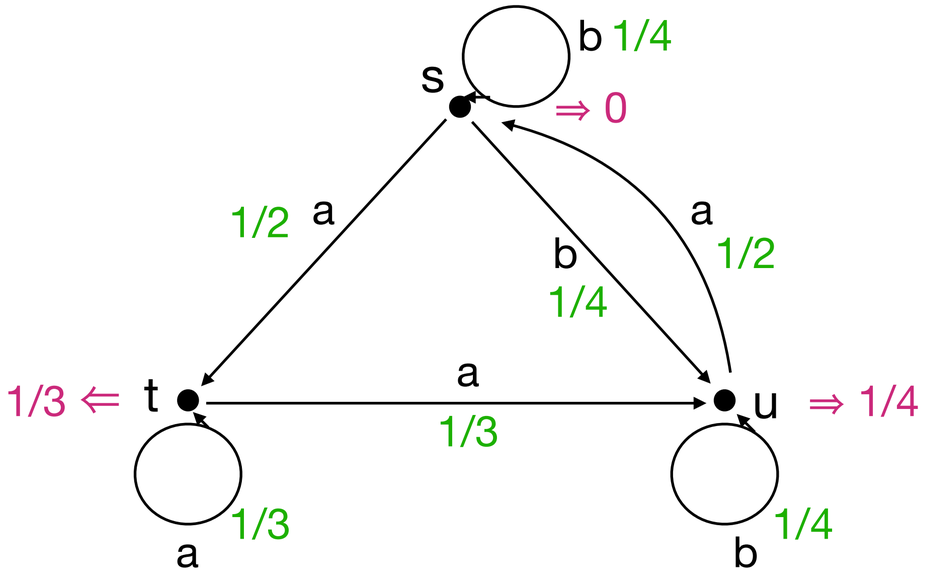


Infinite Traces

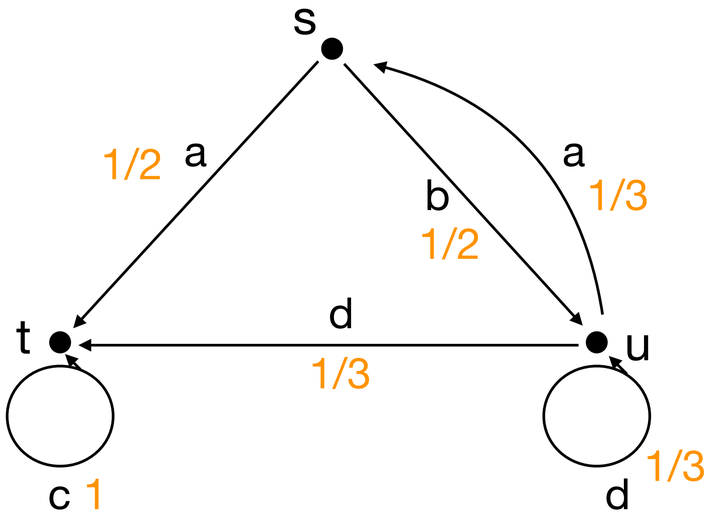


Finite Traces

c: S→ $\mathcal{D}_{\leq}$ (A x S)

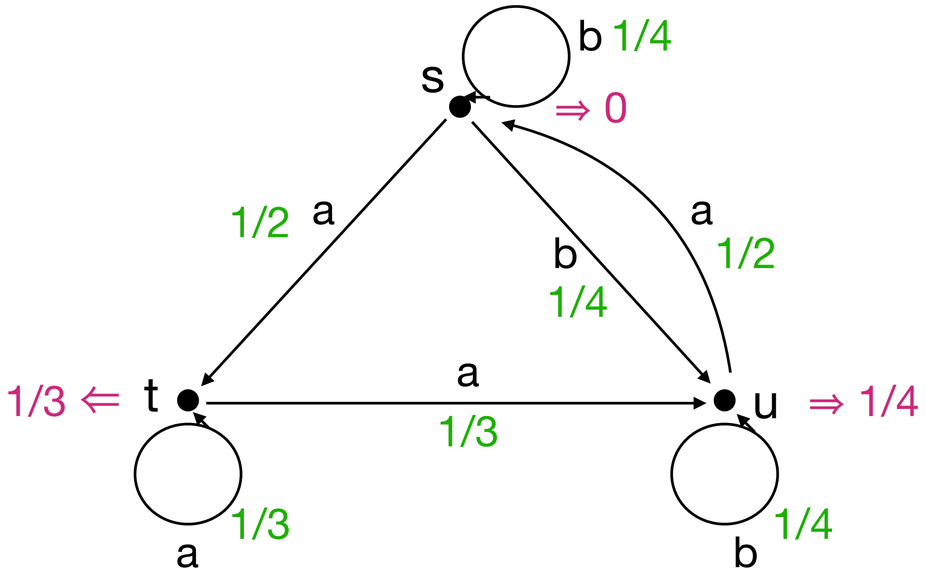


Infinite Traces



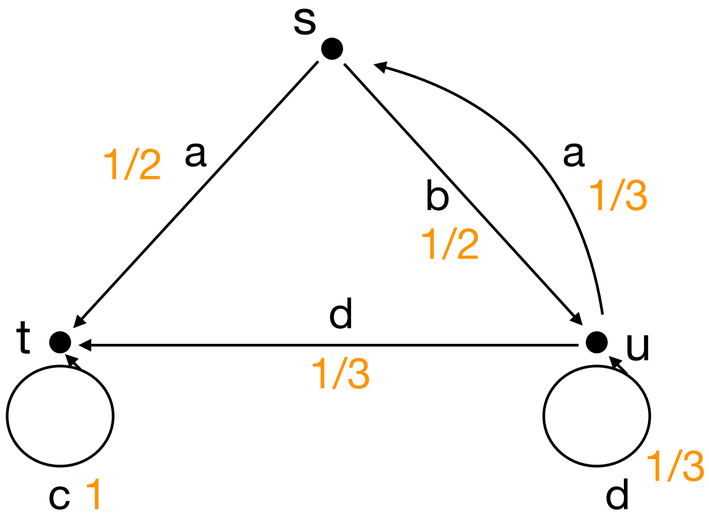
Finite Traces

c: S→ $\mathcal{D}_{\leq}$ (A x S)



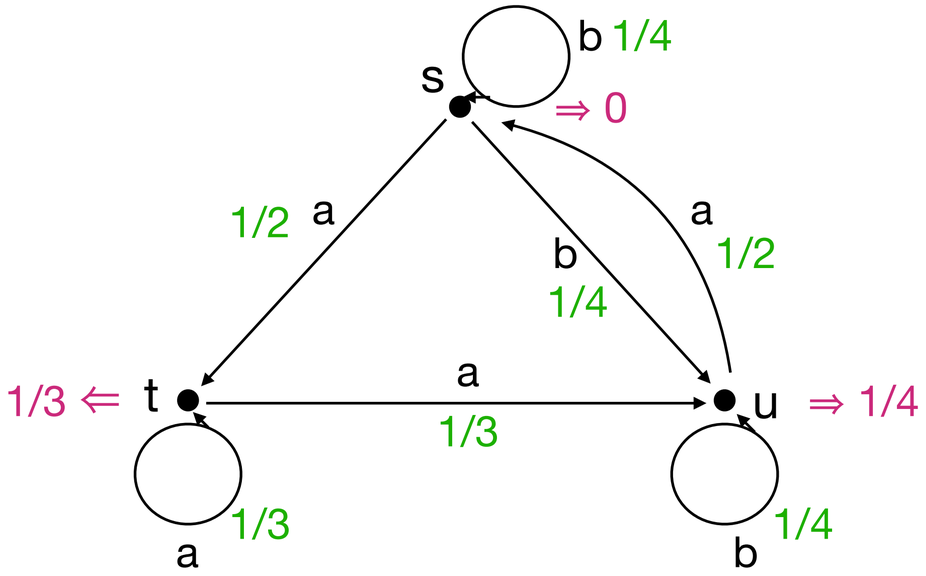
Infinite Traces

c: S→ $\mathcal{D}$ (A x S)



Finite Traces

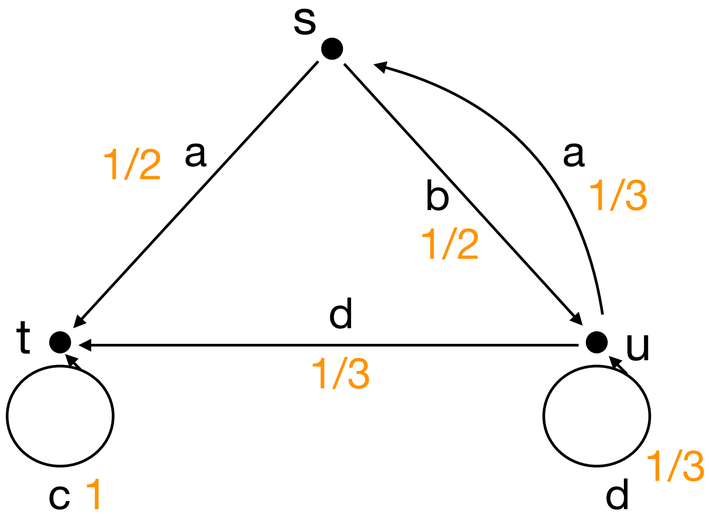
c: S→ $\mathcal{D}_{\leq}$ (A x S)



trace (s) (aaa) = Prob (sttt, sttu, stus)

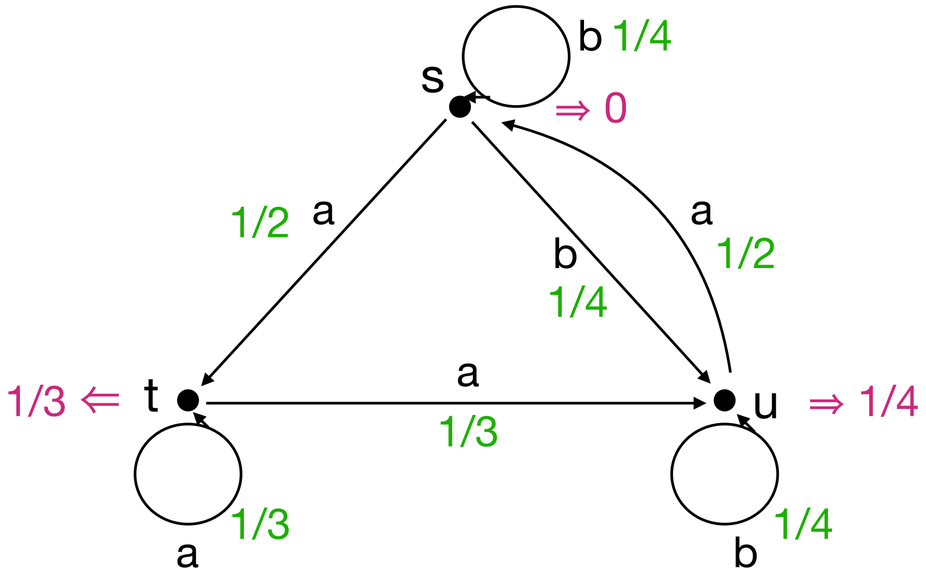
Infinite Traces

c: S→ $\mathcal{D}$ (A x S)



Finite Traces

c: S → $\mathcal{D}_{\leq}$ (A x S)

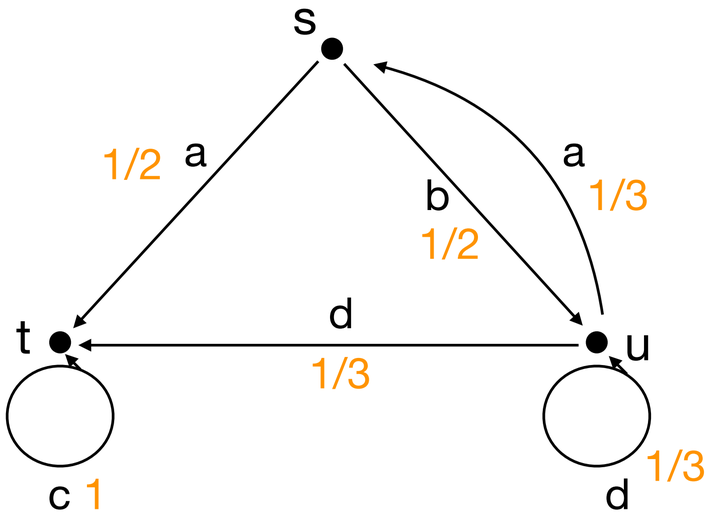


trace (s) (aaa) = Prob (sttt, sttu, stus)

= 1/2 · 1/3 · 1/3 · 1/3

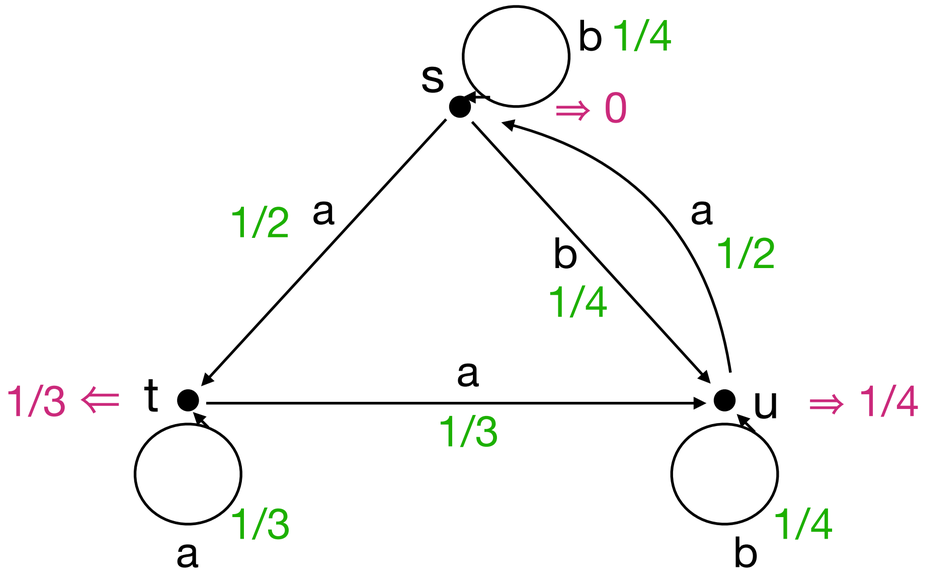
Infinite Traces

c: S → $\mathcal{D}$ (A x S)



Finite Traces

c: S → $\mathcal{D}_{\leq}$ (A x S)



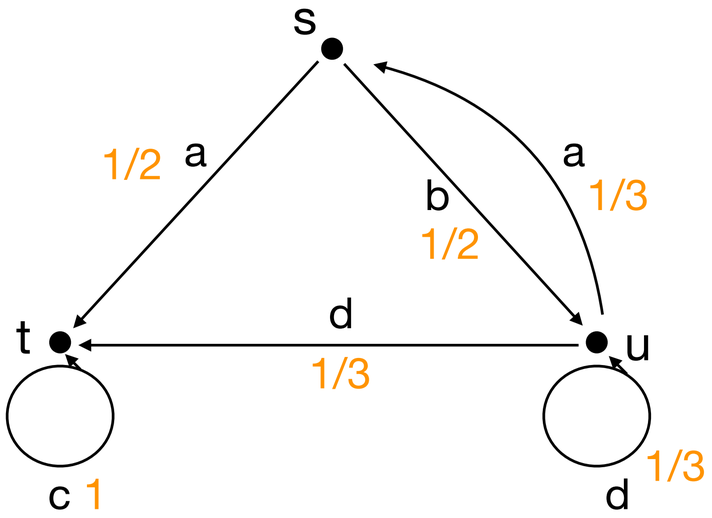
trace (s) (aaa) = Prob (sttt, sttu, stus)

= 1/2 · 1/3 · 1/3 · 1/3

+ 1/2 · 1/3 · 1/3 · 1/4

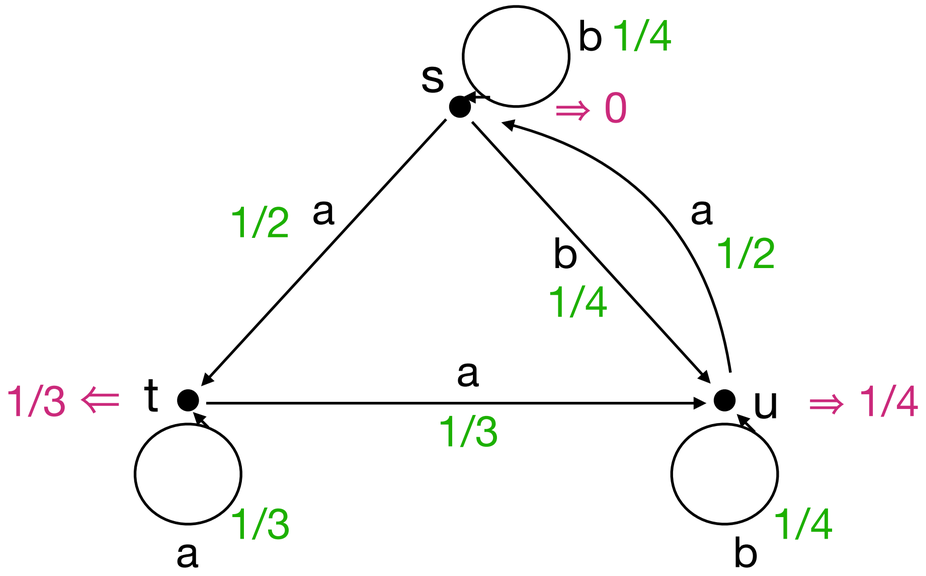
Infinite Traces

c: S → $\mathcal{D}$ (A x S)



Finite Traces

c: S → $\mathcal{D}_{\leq}$ (A x S)



trace (s) (aaa) = Prob (sttt, sttu, stus)

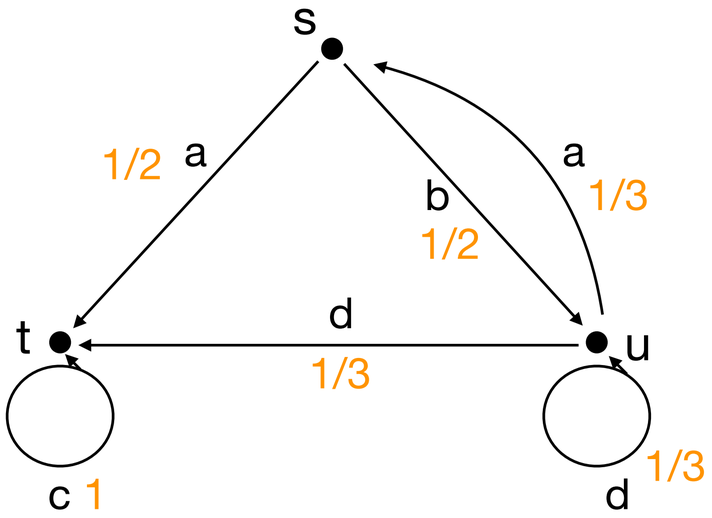
= 1/2 · 1/3 · 1/3 · 1/3

+ 1/2 · 1/3 · 1/3 · 1/4

+ 1/2 · 1/3 · 1/2 · 0

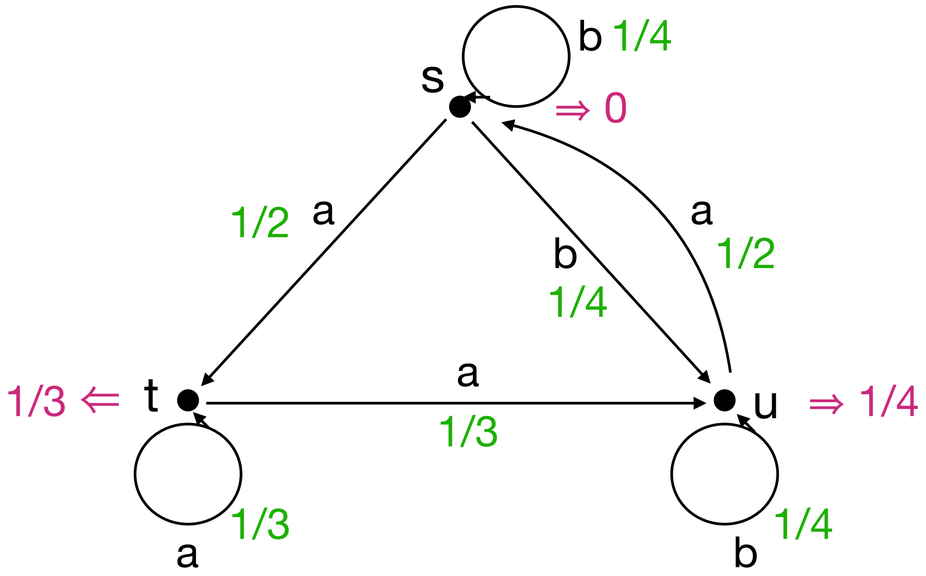
Infinite Traces

c: S → $\mathcal{D}$ (A x S)



Finite Traces

c: S → $\mathcal{D}_{\leq}$ (A x S)



trace (s) (aaa) = Prob (sttt, sttu, stus)

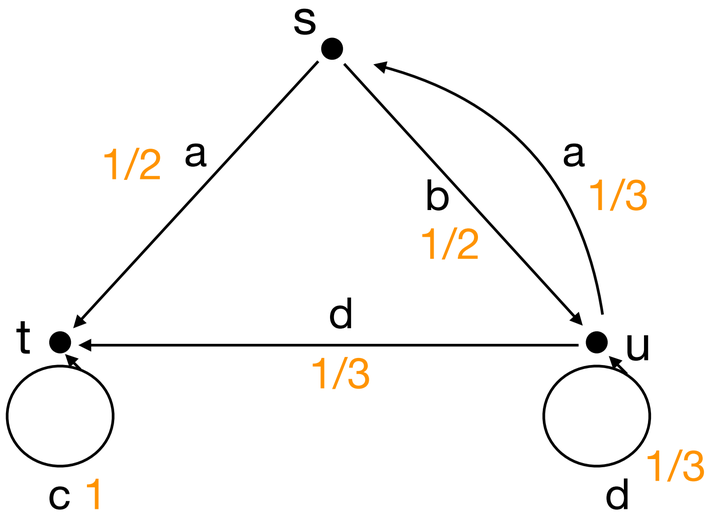
= 1/2 · 1/3 · 1/3 · 1/3

+ 1/2 · 1/3 · 1/3 · 1/4

+ 1/2 · 1/3 · 1/2 · 0

Infinite Traces

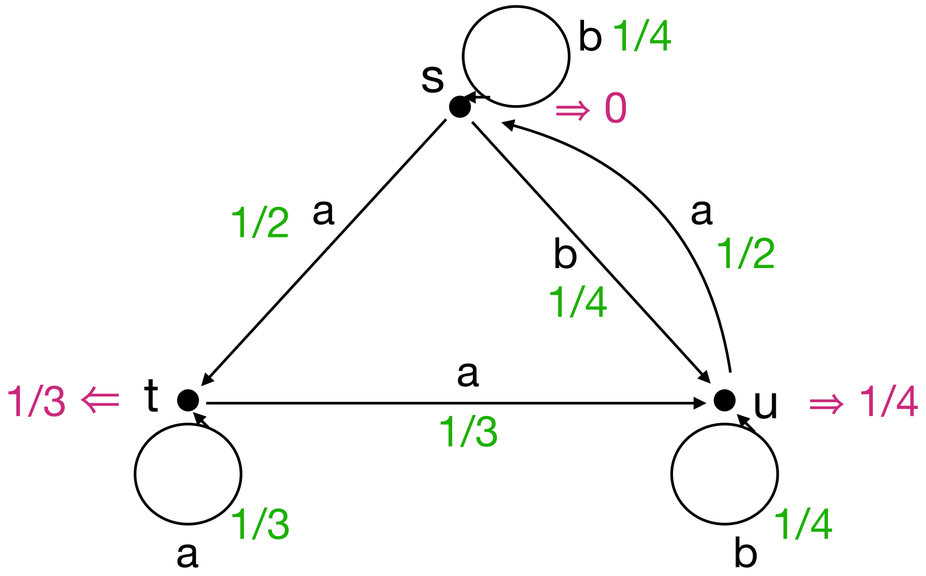
c: S → $\mathcal{D}$ (A x S)



trace (s) (ac^{ω}) = 1/2

Finite Traces

c: S → $\mathcal{D}_{\leq}$ (A x S)



trace (s) (aaa) = Prob (sttt, sttu, stus)

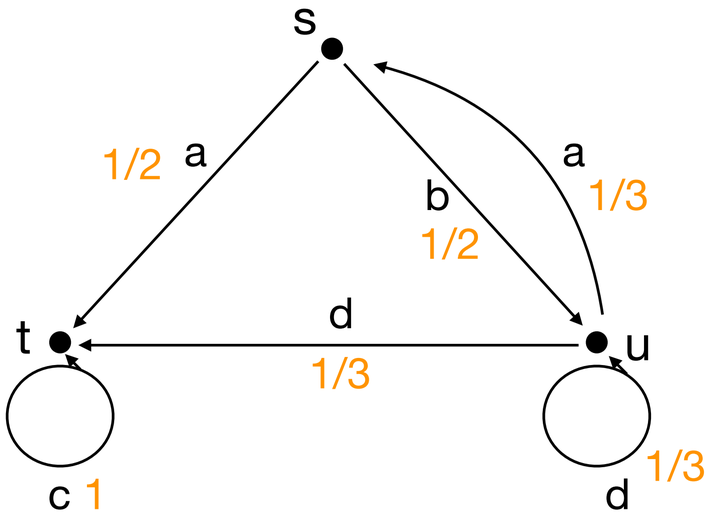
= 1/2 · 1/3 · 1/3 · 1/3

+ 1/2 · 1/3 · 1/3 · 1/4

+ 1/2 · 1/3 · 1/2 · 0

Infinite Traces

c: S → $\mathcal{D}$ (A x S)

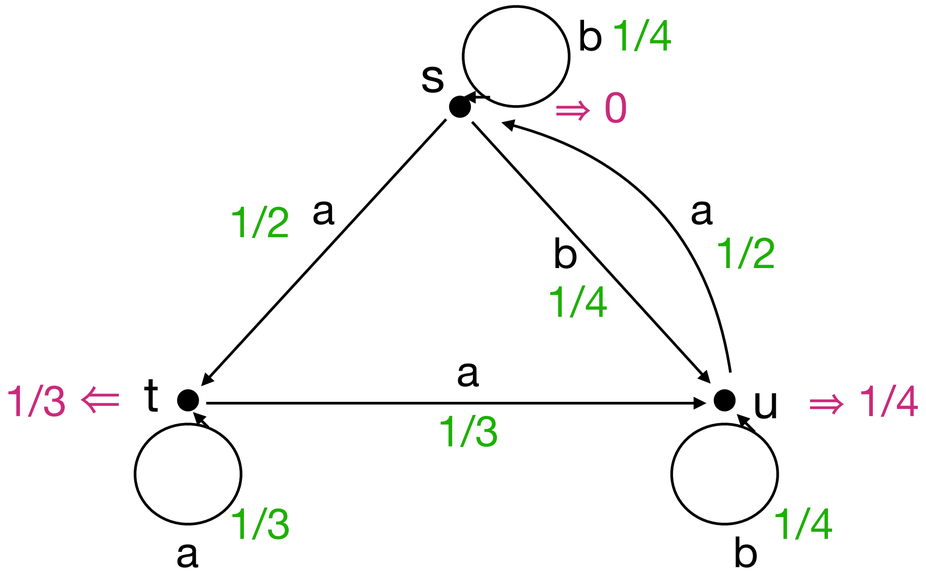


trace (s) (ac^{ω}) = 1/2

trace (s) (bd^{ω}) = $\lim_{n\to\infty}$ 1/2 · $(1/3)^n$ = 0

Finite Traces

c: S → $\mathcal{D}_{\leq}$ (A x S)



trace (s) (aaa) = Prob (sttt, sttu, stus)

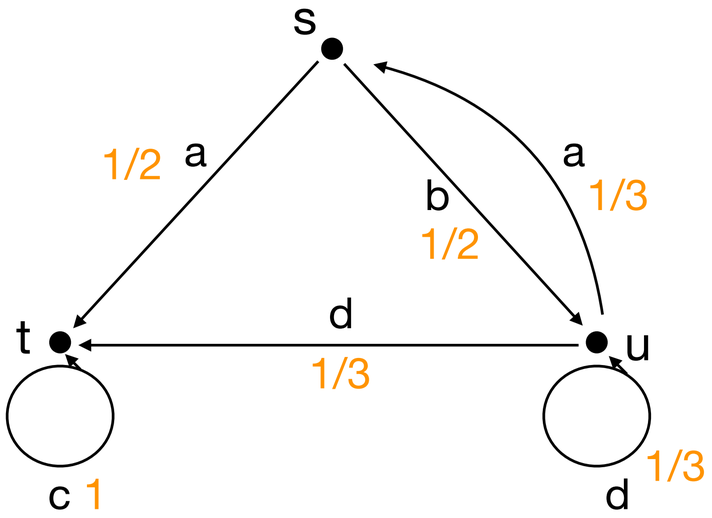
= 1/2 · 1/3 · 1/3 · 1/3

+ 1/2 · 1/3 · 1/3 · 1/4

+ 1/2 · 1/3 · 1/2 · 0

Infinite Traces

c: S → $\mathcal{D}$ (A x S)



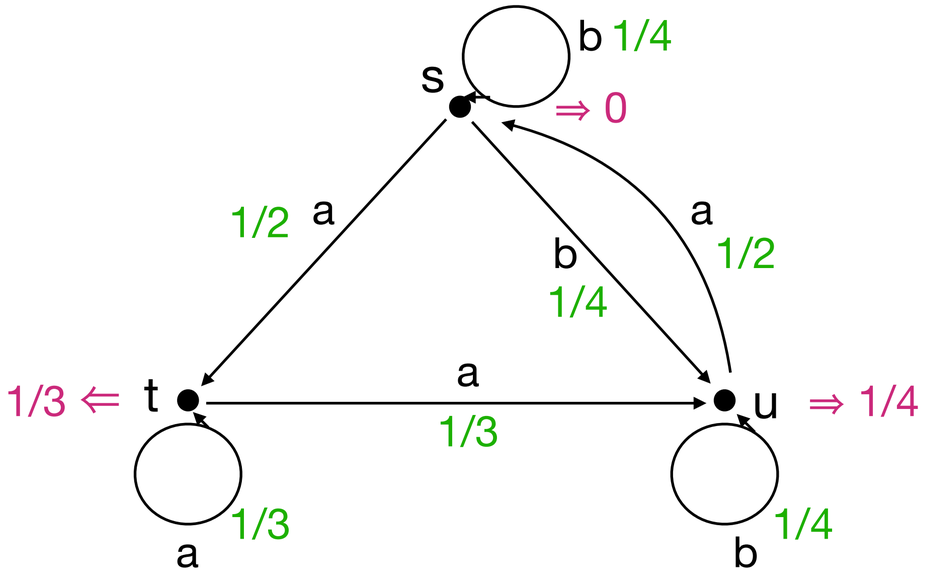
trace (s) (ac^{ω}) = 1/2

trace (s) (bd^{ω}) = $\lim_{n\to\infty}$ 1/2 · $(1/3)^n$ = 0

Therefore, measurable spaces !

Finite Traces

c: S → $\mathcal{D}_{\leq}$(A x S)



trace (s) (aaa) = Prob (sttt, sttu, stus)

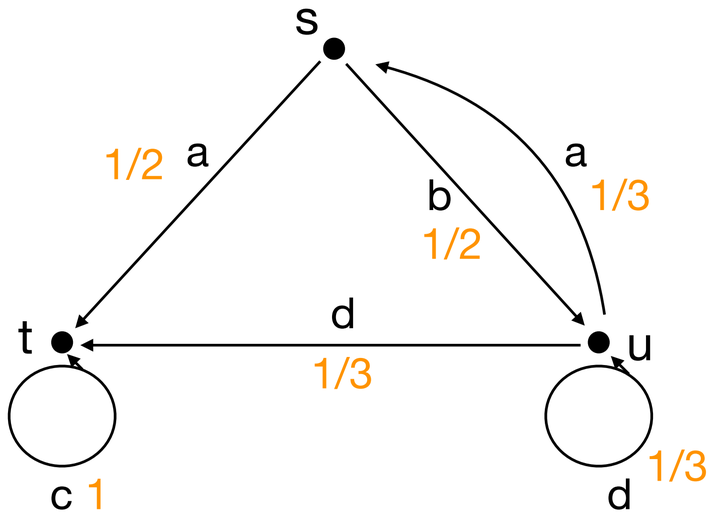
= 1/2 · 1/3 · 1/3 · 1/3

+ 1/2 · 1/3 · 1/3 · 1/4

+ 1/2 · 1/3 · 1/2 · 0

Infinite Traces

c: S → $\mathcal{D}$(A x S)



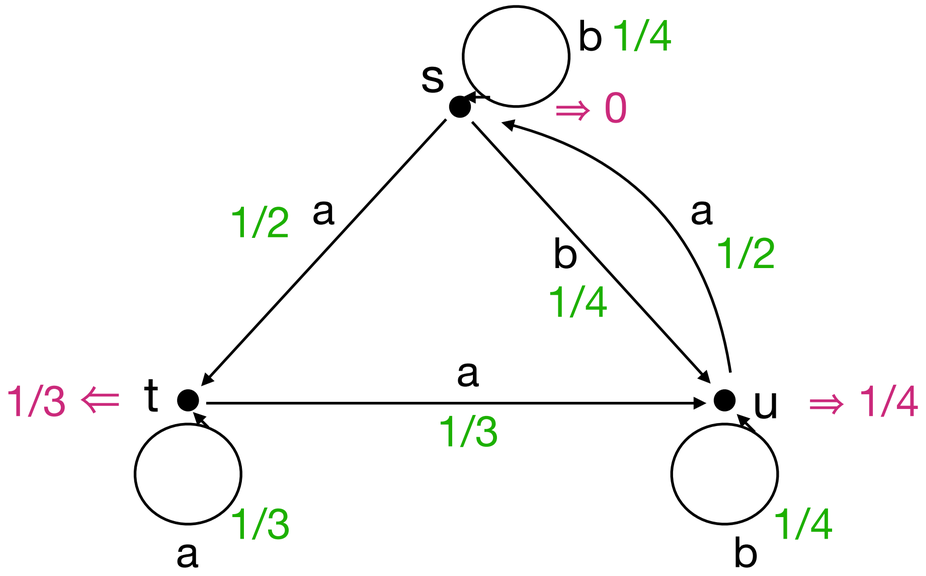
Each state defines a Borel probability measure on A^{ω}, trace (s) ∈ Prob(A^{ω})

with the σ-algebra generated by the cylinders

$B_w = \{ \alpha \in A^{\omega} \mid w \text{ is a prefix of } \alpha \}$

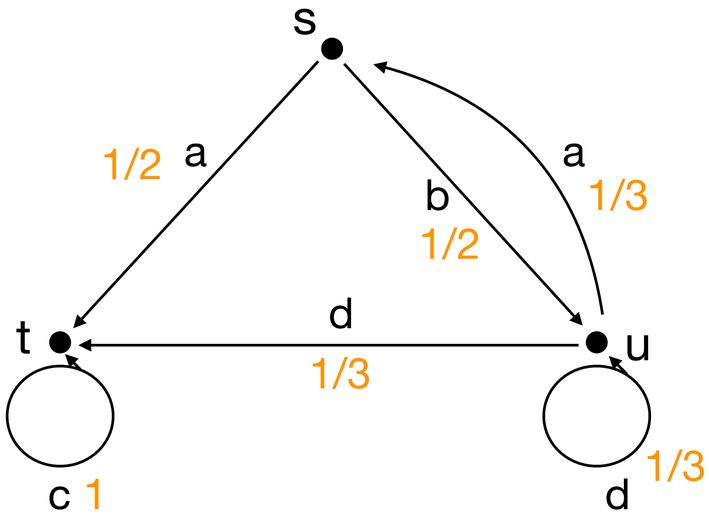
Finite Traces

c: S→ $\mathcal{D}_{\leq}$(A x S)



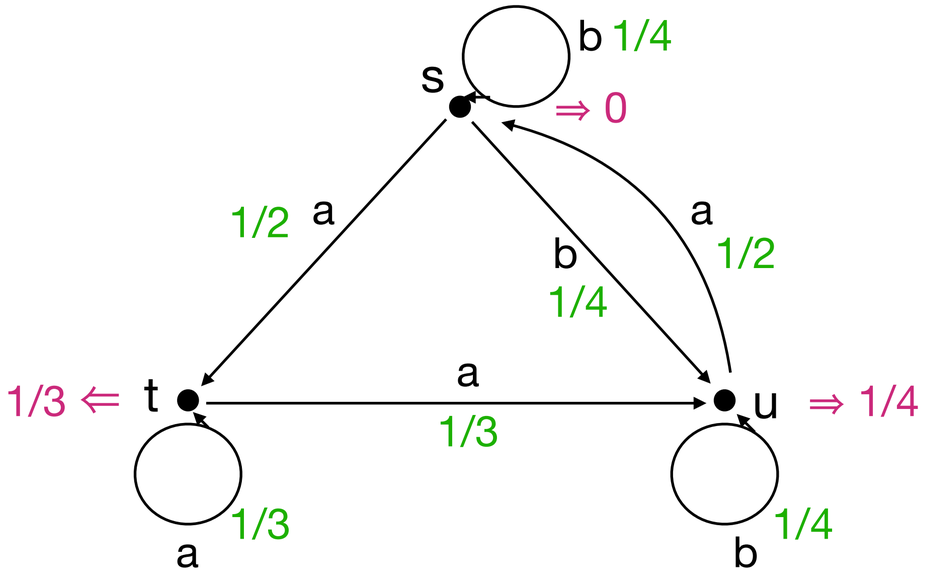
Infinite Traces

c: S→ $\mathcal{D}$(A x S)



Finite Traces

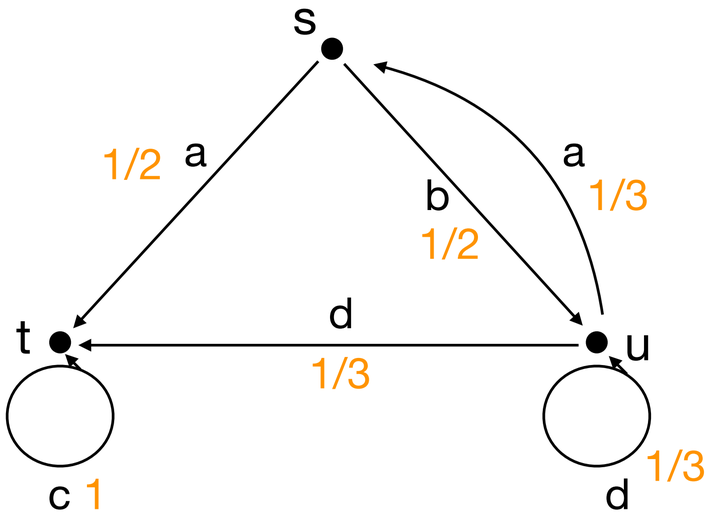
c: S→ $\mathcal{D}_{\leq}$(A x S)



trace (s) (ε) = c (s) (*) = output-at-s

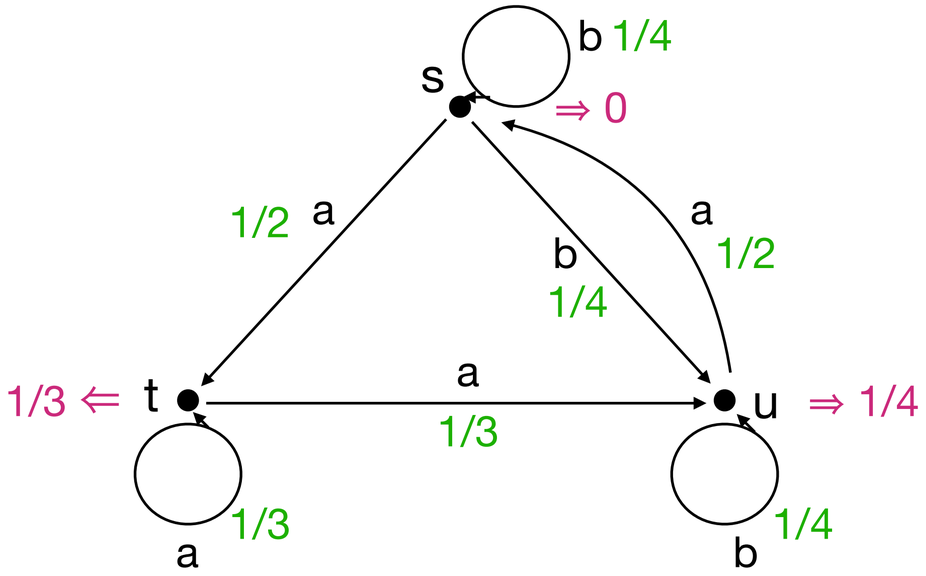
Infinite Traces

c: S→ $\mathcal{D}$(A x S)



Finite Traces

c: S → $\mathcal{D}_{\leq}$(A x S)

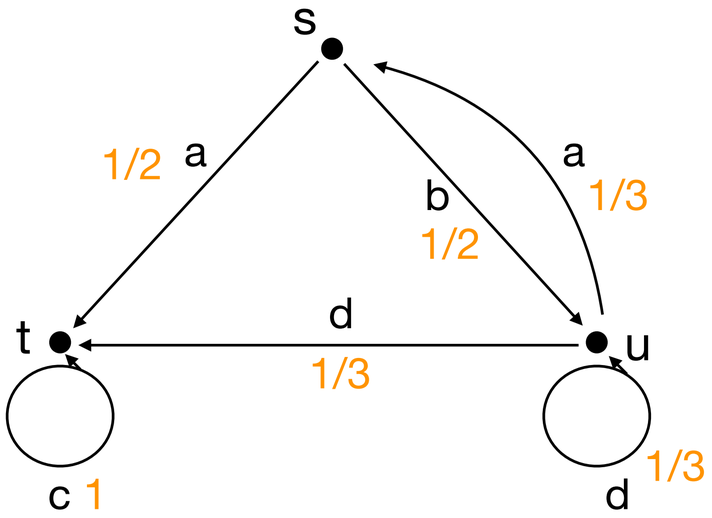


trace (s) (ε) = c (s) (*) = output-at-s

trace (s) (aw) = $\sum_{(a,t) \in \text{supp}(s)}$ c (s) (a,t) · trace (t) (w)

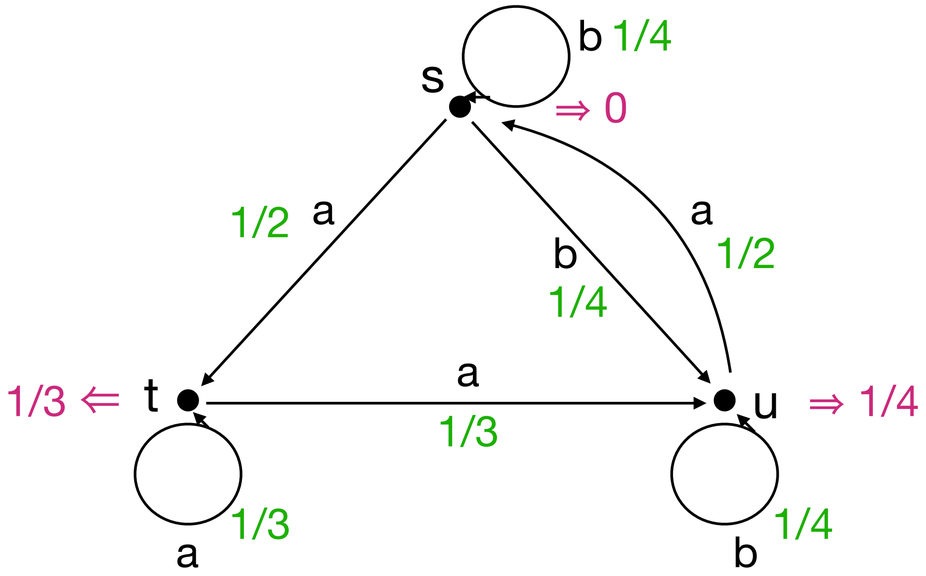
Infinite Traces

c: S → $\mathcal{D}$(A x S)



Finite Traces

c: S → $\mathcal{D}_{\leq}$(A x S)

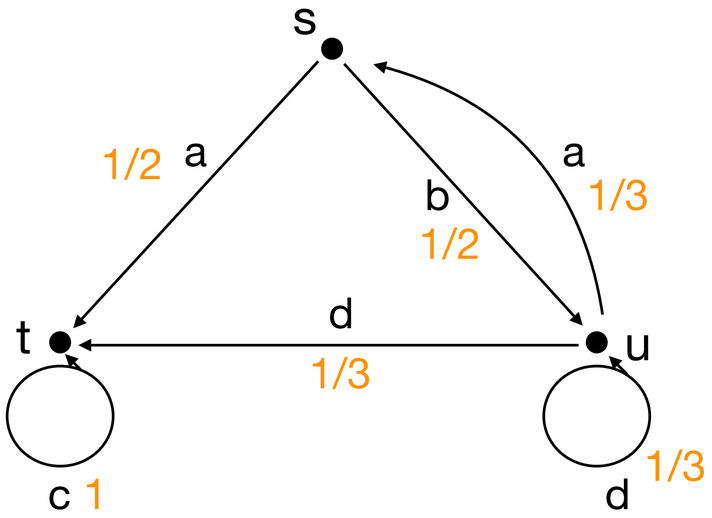


trace (s) (ε) = c (s) (*) = output-at-s

trace (s) (aw) = $\sum_{(a,t) \in \text{supp}(s)}$ c (s) (a,t) · trace (t) (w)

Infinite Traces

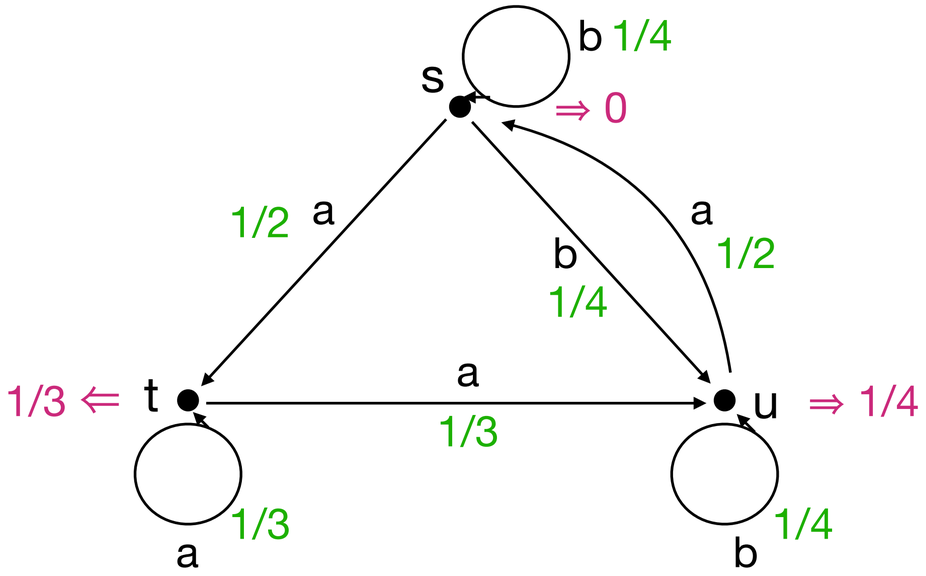
c: S → $\mathcal{D}$(A x S)



trace (s) (B_ε) = 1

Finite Traces

c: S → $\mathcal{D}_{\leq}$(A x S)

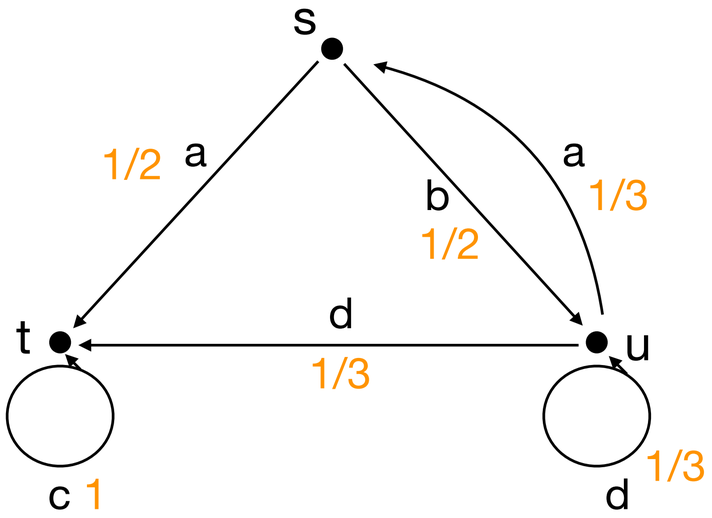


trace (s) (ε) = c (s) (*) = output-at-s

trace (s) (aw) = $\sum_{(a,t) \in \text{supp(s)}}$ c (s) (a,t) · trace (t) (w)

Infinite Traces

c: S → $\mathcal{D}$(A x S)



trace (s) (B_ε) = 1

trace (s) (B_{aw}) = $\sum_{(a,t) \in \text{supp(s)}}$ c (s) (a,t) · trace (t) (B_w)

Trace Equivalence and History

Two states s and t are trace equivalent, notation $s =_{tr} t$, if and only if they have the same trace:

$$s =_{tr} t \iff \text{trace}(s) = \text{trace}(t)$$

Trace Equivalence and History

Two states s and t are trace equivalent, notation $s =_{tr} t$, if and only if they have the same trace:

$$s =_{tr} t \iff \text{trace}(s) = \text{trace}(t)$$

Previous work on infinite trace equivalence (coalgebraically):

Long history
in formal methods
origins in fractal theory

Trace Equivalence and History

Two states s and t are trace equivalent, notation $s =_{tr} t$, if and only if they have the same trace:

$$s =_{tr} t \iff \text{trace}(s) = \text{trace}(t)$$

Previous work on infinite trace equivalence (coalgebraically):

Jacobs ’04
Cirstea ’10
Kerstan & König ’13
Urabe & Hasuo ’15
Goy & Rot ‘18

Long history
in formal methods
origins in fractal theory

Trace Equivalence and History

Two states s and t are trace equivalent, notation $s =_{tr} t$, if and only if they have the same trace:

$$s =_{tr} t \iff \text{trace}(s) = \text{trace}(t)$$

Previous work on infinite trace equivalence (coalgebraically):

Jacobs ’04
Cirstea ’10
Kerstan & König ’13
Urabe & Hasuo ’15
Goy & Rot ‘18

Axiomatization of finite trace equivalence (coalgebraically):

Long history
in formal methods
origins in fractal theory

Trace Equivalence and History

Two states s and t are trace equivalent, notation $s =_{tr} t$, if and only if they have the same trace:

$$s =_{tr} t \iff \text{trace}(s) = \text{trace}(t)$$

Previous work on infinite trace equivalence (coalgebraically):

Jacobs ’04
Cirstea ’10
Kerstan & König ’13
Urabe & Hasuo ’15
Goy & Rot ‘18

Long history
in formal methods
origins in fractal theory

Axiomatization of finite trace equivalence (coalgebraically): Silva & S. ‘11

Expressions

Expressions

Expressions for labelled Markov chains: Stark & Smolka ’00

Expressions

Expressions for labelled Markov chains: Stark & Smolka ’00

$$e, f \;:=\; v \mid e \oplus_p f \mid a.e \mid \mathrm{fix}\ v.e$$

for $p \in [0,1]$, $a \in A$

Expressions

Expressions for labelled Markov chains: Stark & Smolka ’00

$$e, f \;:=\; v \mid e \oplus_p f \mid a.e \mid \mathrm{fix}\ v.e \mid *$$

for $p \in [0,1]$, $a \in A$

Expressions

Expressions for labelled Markov chains: Stark & Smolka ’00

$$e, f \;:=\; v \mid e \oplus_p f \mid a.e \mid \mathrm{fix}\ v.e \mid *$$

for $p \in [0,1]$, $a \in A$

Process terms are expressions that are guarded and closed.

Expressions

Expressions for labelled Markov chains: Stark & Smolka ’00

$$e, f \;:=\; v \mid e \oplus_p f \mid a.e \mid \mathrm{fix}\ v.e \mid *$$

for $p \in [0,1]$, $a \in A$

Process terms are expressions that are guarded and closed.

every v in scope of a.(-)

Expressions

Expressions for labelled Markov chains: Stark & Smolka ’00

$$e, f \;:=\; v \mid e \oplus_p f \mid a.e \mid \text{fix } v.e \mid *$$

for $p \in [0,1]$, $a \in A$

Process terms are expressions that are guarded and closed.

every v in scope of a.(-)

every v in scope of fix v.(-)

Expressions

Expressions for labelled Markov chains: Stark & Smolka ’00

$$e, f \;:=\; v \mid e \oplus_p f \mid a.e \mid \text{fix } v.e \mid *$$

for $p \in [0,1]$, $a \in A$

Process terms are expressions that are guarded and closed.

every v in scope of a.(-)

every v in scope of fix v.(-)

Process terms behave:

γ : PTerms → $\mathcal{D}$ (A x PTerms)

Expressions

Expressions for labelled Markov chains: Stark & Smolka ’00

$$e, f \;:=\; v \mid e \oplus_p f \mid a.e \mid \text{fix } v.e \mid *$$

for $p \in [0,1]$, $a \in A$

Process terms are expressions that are guarded and closed.

every v in scope of a.(-)

every v in scope of fix v.(-)

Process terms behave:

γ : PTerms → $\mathcal{D}$ (A x PTerms)

Theorem (Kleene-style) :

Expressions

Expressions for labelled Markov chains: Stark & Smolka ’00

$$e, f \;:=\; v \mid e \oplus_p f \mid a.e \mid \text{fix } v.e \mid *$$

for $p \in [0,1]$, $a \in A$

Process terms are expressions that are guarded and closed.

every v in scope of a.(-)

every v in scope of fix v.(-)

Process terms behave:

γ : PTerms → $\mathcal{D}$ (A x PTerms)

Theorem (Kleene-style) : For every state s in an LMC, there exists a process term t_s with

$$s =_{tr} t_s$$

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

$$e \oplus_1 f = e$$

$$e \oplus_p e = e$$

$$e \oplus_p f = f \oplus_{1-p} e$$

$$e \oplus_p (f \oplus_q g) = (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g$$

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

$e \oplus_1 f = e$

$e \oplus_p e = e$

$e \oplus_p f = f \oplus_{1-p} e$

$e \oplus_p (f \oplus_q g) = (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g$

convex algebras

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

$e \oplus_1 f = e$

$e \oplus_p e = e$

$e \oplus_p f = f \oplus_{1-p} e$

$e \oplus_p (f \oplus_q g) = (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g$

convex algebras

$a.(e \oplus_p f) = a.e \oplus_p a.f$

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

$e \oplus_1 f = e$

$e \oplus_p e = e$

$e \oplus_p f = f \oplus_{1-p} e$

$e \oplus_p (f \oplus_q g) = (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g$

convex algebras

$a.(e \oplus_p f) = a.e \oplus_p a.f$

distributivity

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

$e \oplus_1 f = e$

$e \oplus_p e = e$

$e \oplus_p f = f \oplus_{1-p} e$

$e \oplus_p (f \oplus_q g) = (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g$

convex algebras

$a.\ (e \oplus_p f) = a.e \oplus_p a.f$

distributivity

$\mathrm{fix}\ v.e = e[v := \mathrm{fix}\ v.e\ ]$

$g = e[v := g] \Rightarrow\ g = \mathrm{fix}\ v.e$

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

$e \oplus_1 f = e$

$e \oplus_p e = e$

$e \oplus_p f = f \oplus_{1-p} e$

$e \oplus_p (f \oplus_q g) = (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g$

convex algebras

$a.\ (e \oplus_p f) = a.e \oplus_p a.f$

distributivity

$\mathrm{fix}\ v.e = e[v := \mathrm{fix}\ v.e\ ]$

$g = e[v := g] \Rightarrow\ g = \mathrm{fix}\ v.e$

fixpoints

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

$e \oplus_1 f = e$

$e \oplus_p e = e$

$e \oplus_p f = f \oplus_{1-p} e$

$e \oplus_p (f \oplus_q g) = (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g$

convex algebras

Basically the same from Silva & S. ‘11

$a.\ (e \oplus_p f) = a.e \oplus_p a.f$

distributivity

$\mathrm{fix}\ v.e = e[v := \mathrm{fix}\ v.e\ ]$

$g = e[v := g] \Rightarrow\ g = \mathrm{fix}\ v.e$

fixpoints

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

$e \oplus_1 f = e$

$e \oplus_p e = e$

$e \oplus_p f = f \oplus_{1-p} e$

$e \oplus_p (f \oplus_q g) = (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g$

convex algebras

Basically the same from Silva & S. ‘11

$a.\ (e \oplus_p f) = a.e \oplus_p a.f$

distributivity

$\mathrm{fix}\ v.e = e[v := \mathrm{fix}\ v.e\ ]$

$g = e[v := g] \Rightarrow\ g = \mathrm{fix}\ v.e$

fixpoints

Theorem (Soundness&Completeness):

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

$e \oplus_1 f = e$

$e \oplus_p e = e$

$e \oplus_p f = f \oplus_{1-p} e$

$e \oplus_p (f \oplus_q g) = (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g$

convex algebras

Basically the same from Silva & S. ‘11

$a.\ (e \oplus_p f) = a.e \oplus_p a.f$

distributivity

$\mathrm{fix}\ v.e = e[v := \mathrm{fix}\ v.e\ ]$

$g = e[v := g] \Rightarrow\ g = \mathrm{fix}\ v.e$

fixpoints

Theorem (Soundness&Completeness):

Two states s and t in an LMC are trace equivalent iff their corresponding process terms are provably equivalent

$$s =_{tr} t \iff \vdash t_s = t_t$$

Axioms

Axioms for (infinite) trace semantics of labelled Markov chains:

$e \oplus_1 f = e$

$e \oplus_p e = e$

$e \oplus_p f = f \oplus_{1-p} e$

$e \oplus_p (f \oplus_q g) = (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g$

convex algebras

Basically the same from Silva & S. ‘11

$a.\ (e \oplus_p f) = a.e \oplus_p a.f$

distributivity

$\mathrm{fix}\ v.e = e[v := \mathrm{fix}\ v.e\ ]$

$g = e[v := g] \Rightarrow\ g = \mathrm{fix}\ v.e$

fixpoints

Theorem (Soundness&Completeness):

Two states s and t in an LMC are trace equivalent iff their corresponding process terms are provably equivalent

$$s =_{tr} t \iff \vdash t_s = t_t$$

The Role of Algebras

A lot of work happens in categories of algebras, for LMC - convex algebras

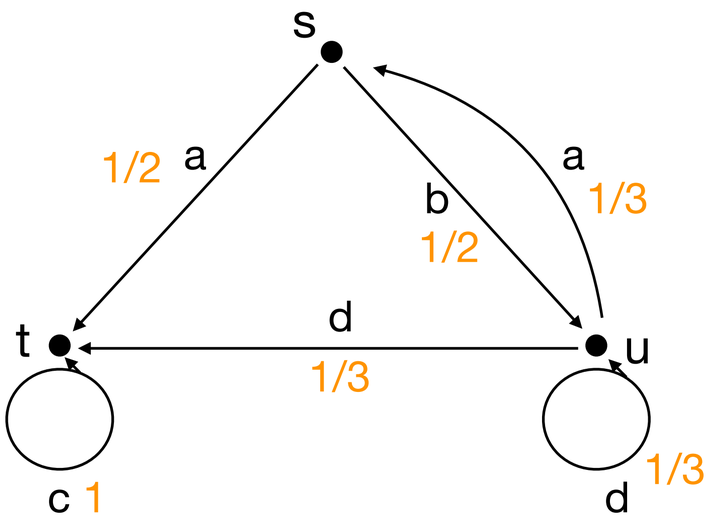
Eilenberg-Moore algebras for $\mathcal{D}$

The Role of Algebras

A lot of work happens in categories of algebras, for LMC - convex algebras

Eilenberg-Moore algebras for $\mathcal{D}$

“Determinization” of

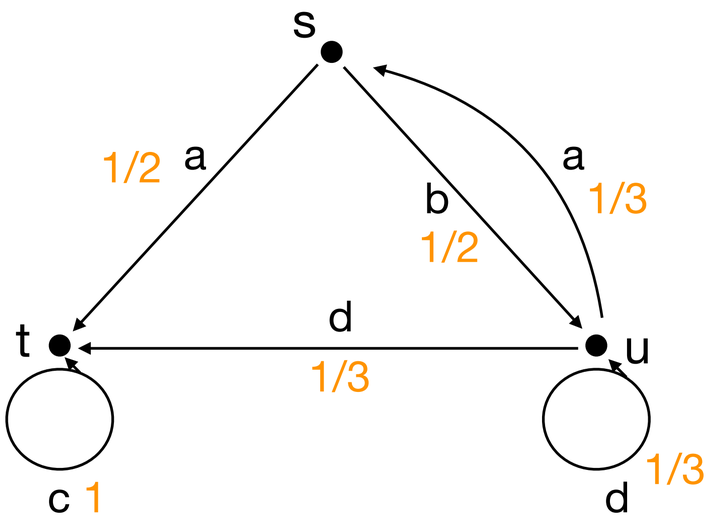


The Role of Algebras

A lot of work happens in categories of algebras, for LMC - convex algebras

Eilenberg-Moore algebras for $\mathcal{D}$

"Determinization" of



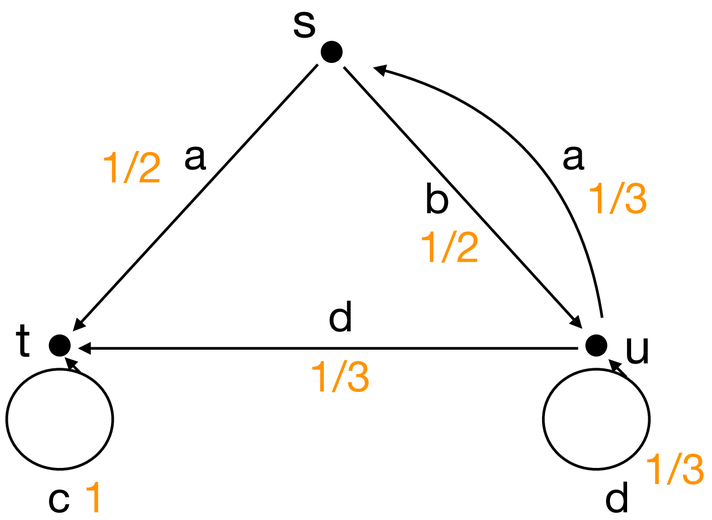
Freely generated by:

The Role of Algebras

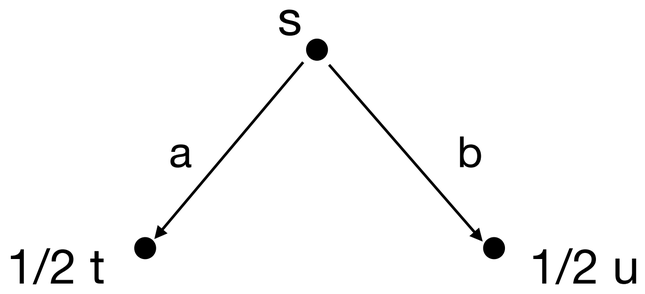
A lot of work happens in categories of algebras, for LMC - convex algebras

Eilenberg-Moore algebras for $\mathcal{D}$

“Determinization” of



Freely generated by:

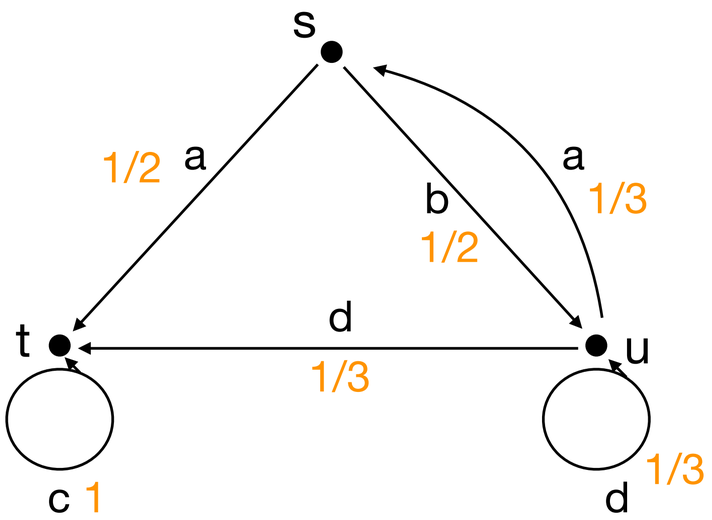


The Role of Algebras

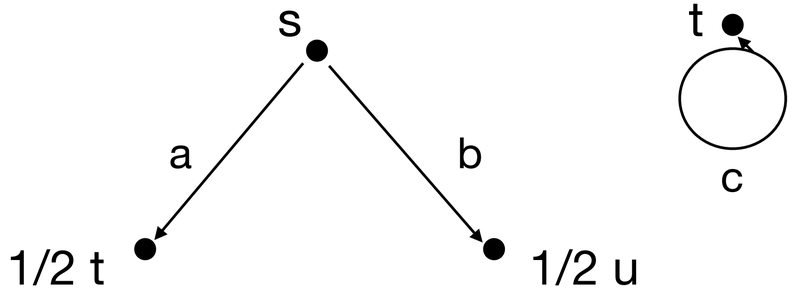
A lot of work happens in categories of algebras, for LMC - convex algebras

Eilenberg-Moore algebras for $\mathcal{D}$

"Determinization" of



Freely generated by:

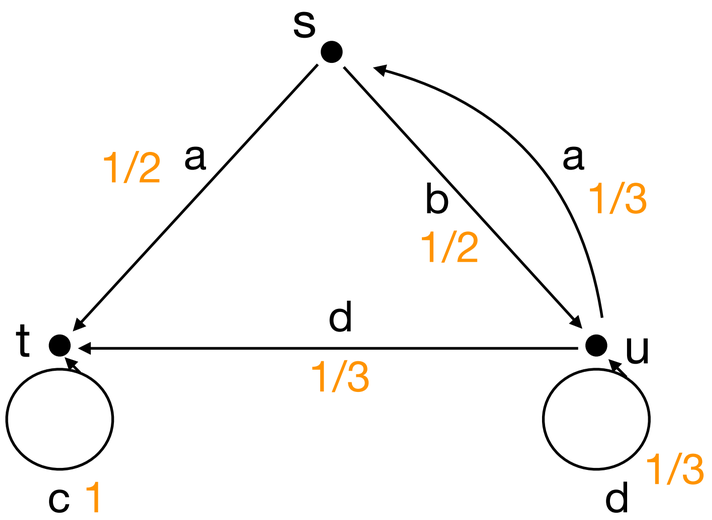


The Role of Algebras

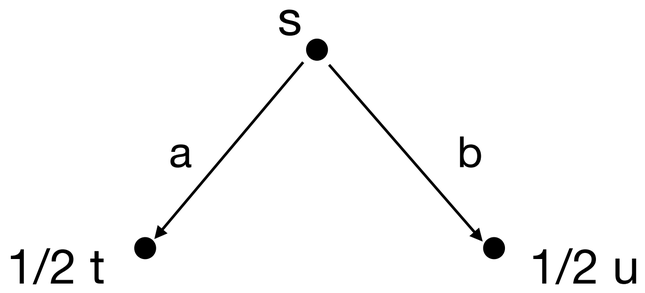
A lot of work happens in categories of algebras, for LMC - convex algebras

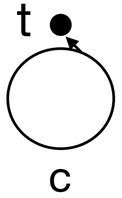
Eilenberg-Moore algebras for $\mathcal{D}$

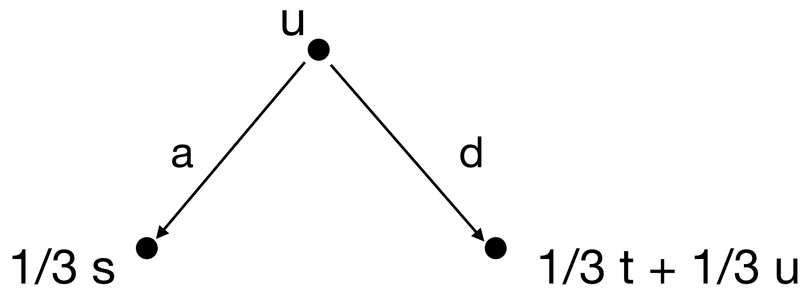
"Determinization" of



Freely generated by:

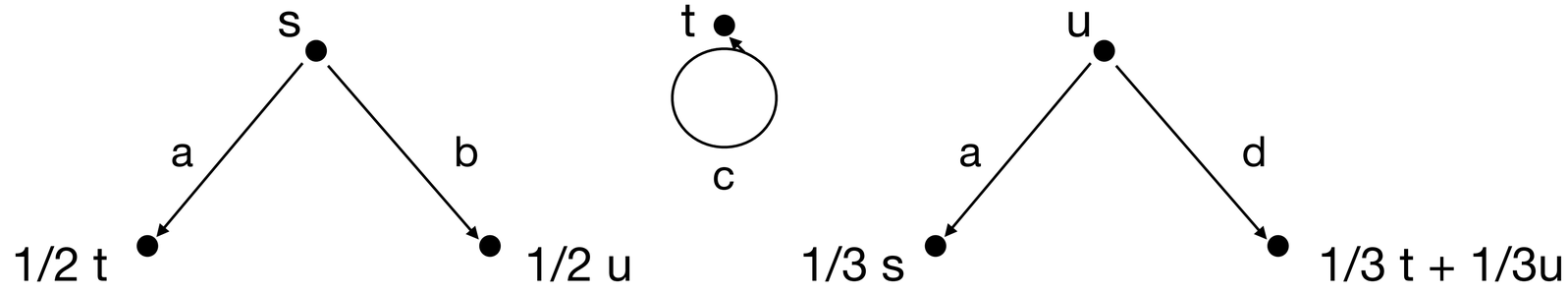






Determinization

Freely generated by:



There are different ways to represent this as a coalgebra $\mathcal{D}_{\leq}\,S \to G\mathcal{D}_{\leq}\,S$

The role of G

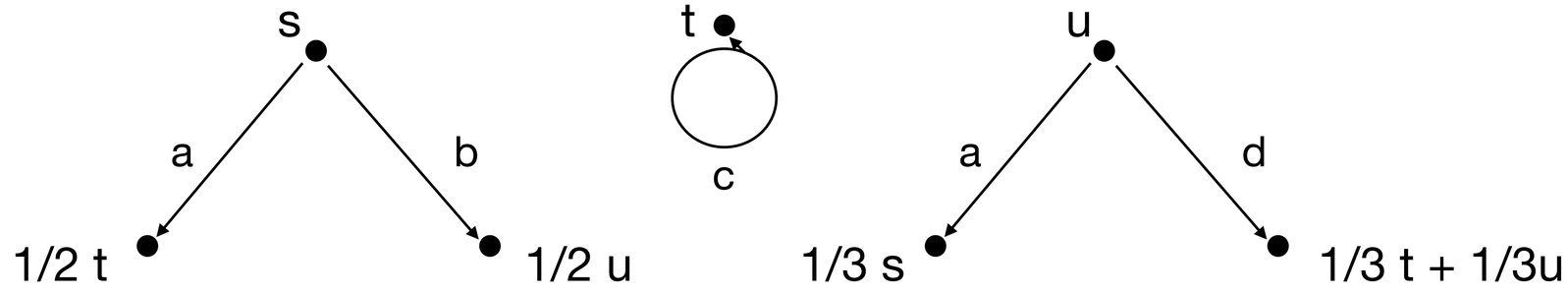
1. Every coalgebra c: S → $\mathcal{D}$ (A x S)
   determinizes to c#: $\mathcal{D}$S → G$\mathcal{D}$S

2. Prob(A^ω) carries the final
   G-coalgebra

   Prob(A^ω) → G Prob(A^ω)

3. Infinite trace equivalence
   is the final coalgebra semantics

Freely generated by:



There are different ways to represent this as a coalgebra $\mathcal{D}_{\leq}$ S → G$\mathcal{D}_{\leq}$ S

The functor G

The functor G

First functor on Convex Algebras:

$$X_{\perp} = \{\perp\} \cup \{r \cdot x \mid r \in (0,1], x \in X\}$$

The functor G

First functor on Convex Algebras:

$$X_\perp = \{\perp\} \cup \{r \cdot x \mid r \in (0,1], x \in X\}$$

adds a free generator to X

The functor G

First functor on Convex Algebras:

$$X_\perp = \{\perp\} \cup \{r \cdot x \mid r \in (0,1], x \in X\}$$

adds a free generator to X

convention: $\perp = 0 \cdot -$

The functor G

First functor on Convex Algebras:

$$X_\perp = \{\perp\} \cup \{r \cdot x \mid r \in (0,1], x \in X\}$$

adds a free generator to X

convention: $\perp = 0 \cdot -$

Now G is just an exponent:

$$GX = \{ f : A \to X_\perp \mid f(a) = r_a \cdot x, \sum_{a \in A} r_a = 1 \}$$

The functor G

First functor on Convex Algebras:

$$X_\perp = \{\perp\} \cup \{r \cdot x \mid r \in (0,1], x \in X\}$$

adds a free generator to X

convention: $\perp = 0 \cdot -$

Now G is just an exponent:

$$GX = \{ f : A \to X_\perp \mid f(a) = r_a \cdot x, \sum_{a \in A} r_a = 1 \}$$

Idea (total weight) from Goy & Rot ‘18

The functor G

First functor on Convex Algebras:

$$X_\perp = \{\perp\} \cup \{r \cdot x \mid r \in (0,1], x \in X\}$$

adds a free generator to X

convention: $\perp = 0 \cdot -$

Now G is just an exponent:

$$GX = \{ f : A \to X_\perp \mid f(a) = r_a \cdot x, \sum_{a \in A} r_a = 1 \}$$

The functor G

First functor on Convex Algebras:

$$X_\perp = \{\perp\} \cup \{r \cdot x \mid r \in (0,1], x \in X\}$$

adds a free generator to X

convention: $\perp = 0 \cdot -$

Now G is just an exponent:

$$GX = \{ f : A \to X_\perp \mid f(a) = r_a \cdot x, \textstyle\sum_{a \in A} r_a = 1 \}$$

Proof obligations:

G is finitary, preserves surjective homomorphisms,… is proper !

Properness

Properness

A zigzag is a diagram in F-coalgebras over convex algebras ($\mathcal{D}$ - algebras) of the form:

Properness

A zigzag is a diagram in F-coalgebras over convex algebras ($\mathcal{D}$ - algebras) of the form:

Properness

A zigzag is a diagram in F-coalgebras over convex algebras ($\mathcal{D}$ - algebras) of the form:

It relates x ∈X and y ∈Y, denoted x ~ y, if there exist $z_{2k} \in Z_{2k}$, for k = 1,.., n-1 with $z_0 = x$, $z_{2n} = y$,

$$f_{2k-1}(z_{2(k-1)}) = f_{2k}(z_{2k})$$

Properness

A zigzag is a diagram in F-coalgebras over convex algebras ($\mathcal{D}$ - algebras) of the form:

It relates x ∈X and y ∈Y, denoted x ~ y, if there exist $z_{2k} \in Z_{2k}$, for k = 1,.., n-1 with $z_0 = x$, $z_{2n} = y$,

$$f_{2k-1}(z_{2(k-1)}) = f_{2k}(z_{2k})$$

F is proper if whenever X and Y have free fg carriers, and x ~ y, then there is a zigzag whose all nodes have free fg carriers.

Properness

A zigzag is a diagram in F-coalgebras over convex algebras ($\mathcal{D}$ - algebras) of the form:

It relates $x \in X$ and $y \in Y$, denoted $x \sim y$, if there exist $z_{2k} \in Z_{2k}$, for $k = 1,.., n-1$ with $z_0 = x$, $z_{2n} = y$,

$$f_{2k-1}(z_{2(k-1)}) = f_{2k}(z_{2k})$$

F is proper if whenever X and Y have free fg carriers, and $x \sim y$, then there is a zigzag whose all nodes have free fg carriers.

For the upper nodes, it suffices to be fg.

Some More History

Milius ’12 If fp = fg and  , then “completeness”

S. & Woracek ’15 fp = fg for convex algebras, but no 

Milius ’13/‘17 If proper functor and , then “completeness”

S. & Woracek ’18 Proper functor(s) hard proof

Some More History

Milius ’12 If fp = fg and ★, then “completeness”

S. & Woracek ’15 fp = fg for convex algebras, but no ★

Milius ’13/‘17 If proper functor and ● ● ●, then “completeness”

S. & Woracek ’18 Proper functor(s) hard proof

Now, for infinite traces: Remarkably beautiful and easy proof of properness for G

Some More History

Milius ’12 If fp = fg and  , then “completeness”

S. & Woracek ’15 fp = fg for convex algebras, but no 

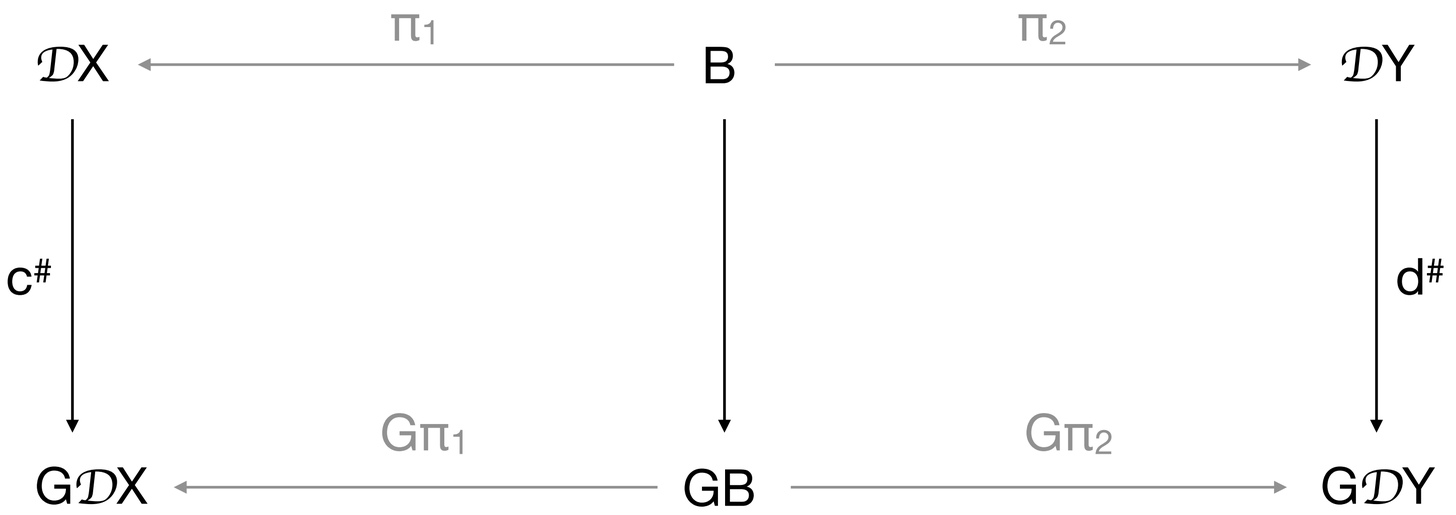
Milius ’13/‘17 If proper functor and , then “completeness”

S. & Woracek ’18 Proper functor(s) hard proof

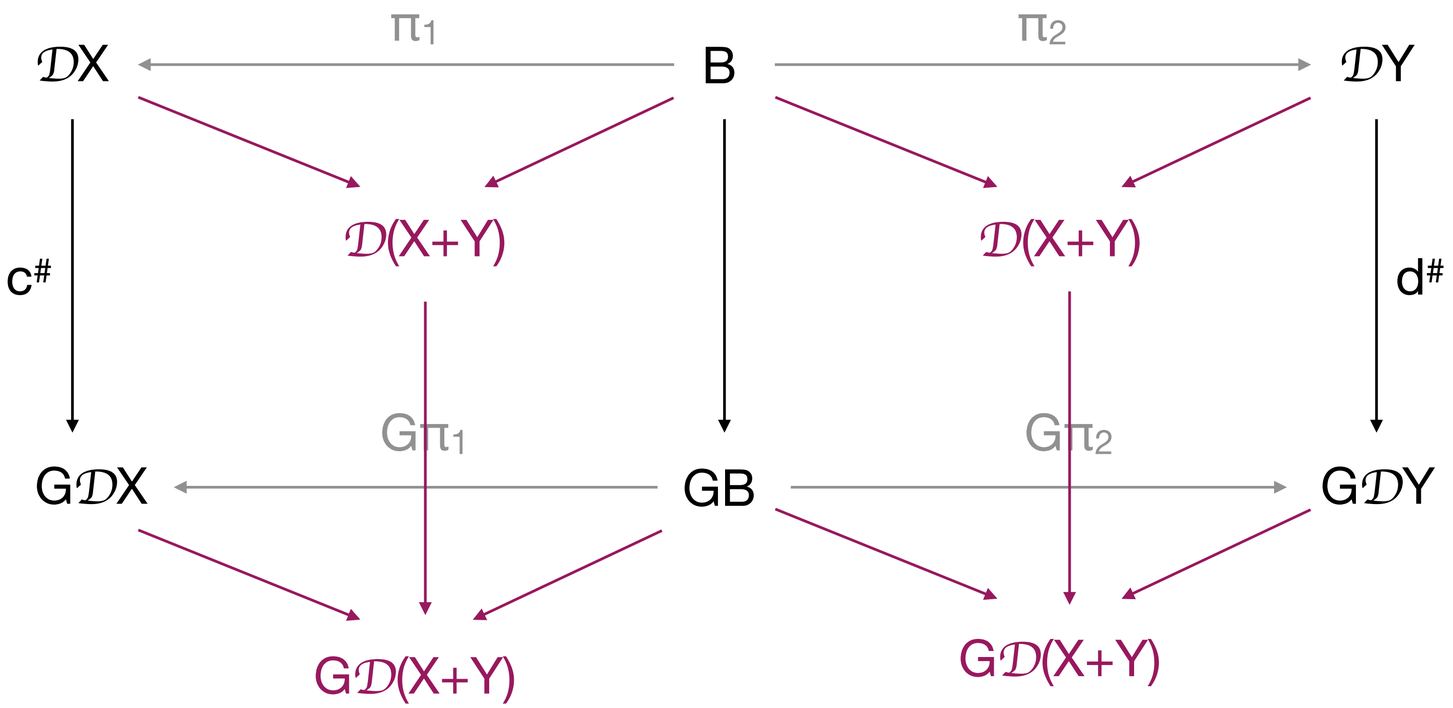
Now, for infinite traces: Remarkably beautiful and easy proof of properness for G

Bisimilarity gives the zigzag

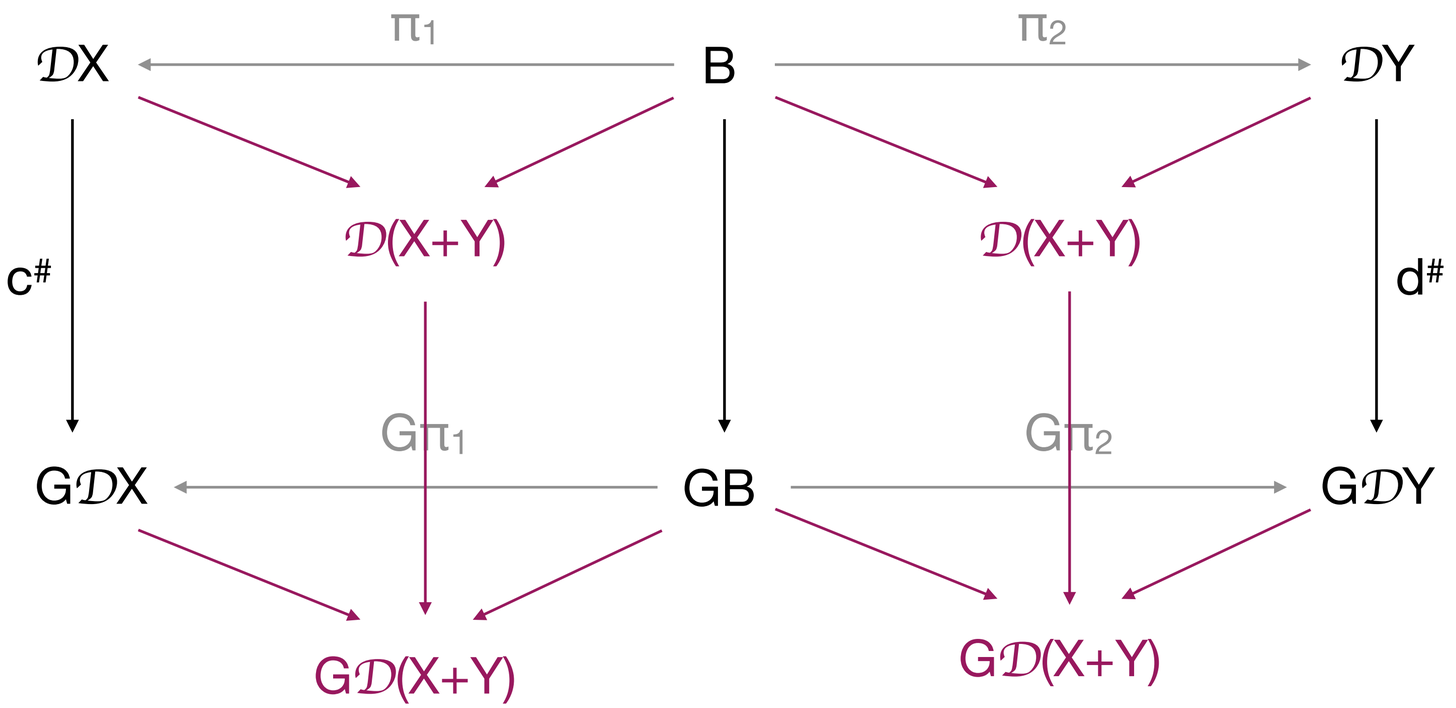
Bisimilarity Gives the Zigzag



Bisimilarity Gives the Zigzag

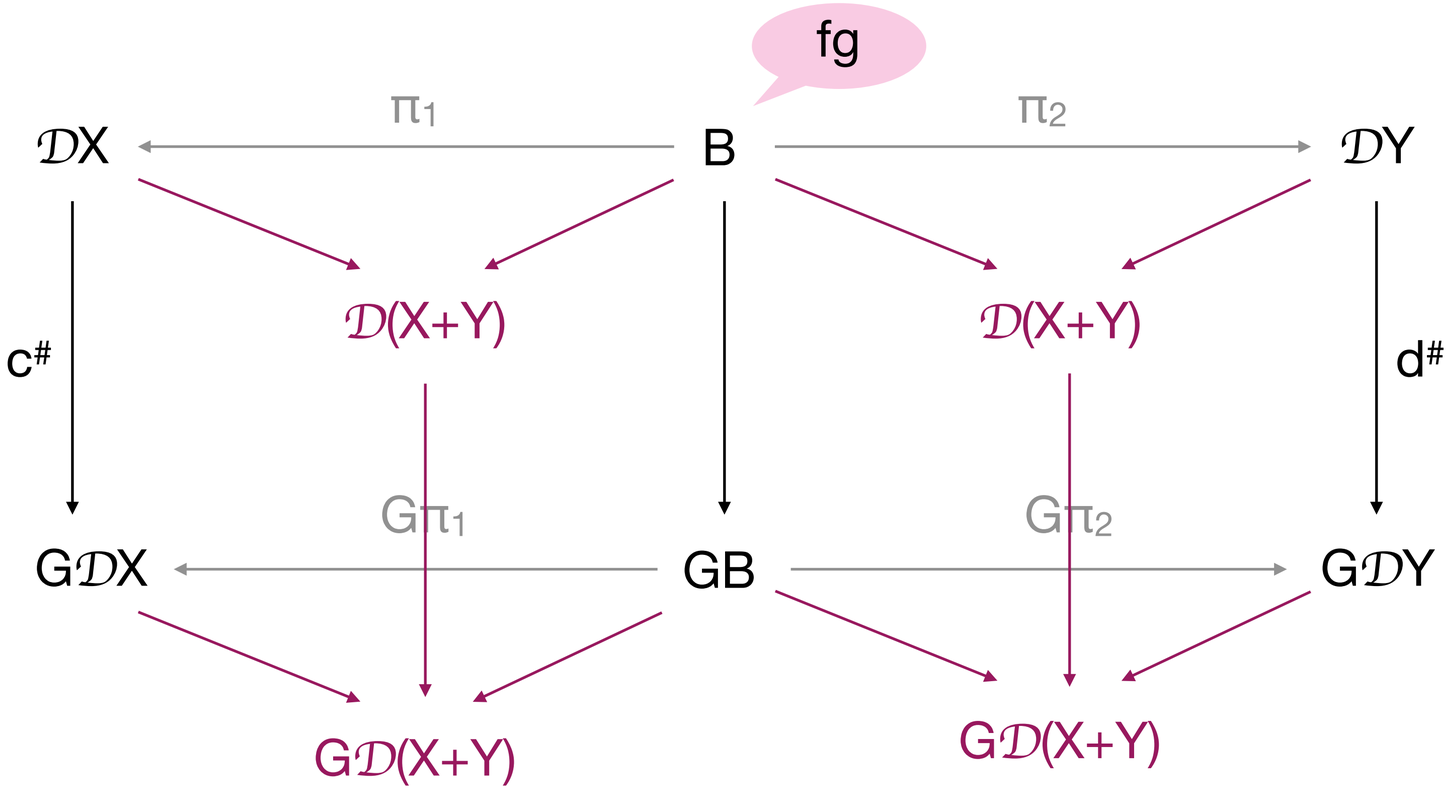


Bisimilarity Gives the Zigzag



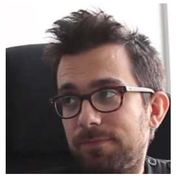
S. & Woracek ’15 B is fg as a subalgebra iff it is topologically closed in $\mathbb{R}^{2|X+Y|}$

Bisimilarity Gives the Zigzag



S. & Woracek ’15 B is fg as a subalgebra iff it is topologically closed in $\mathbb{R}^{2|X+Y|}$

Thank You and Thanks to:



Filippo Bonchi

Corina Cirstea

Josée Desharnais

Ichiro Hasuo

Bart Jacobs

Larry Moss

Tori Noquez

Todd Schmidt

Alexandra Silva

Valeria Vignudelli

Erik de Vink **TU/e**

Harald Woracek

Thank You !

Strong foundations enhance developments in formal methods.

Coalgebraic generalisations may give strong foundations.

Two recent cases in favour of bisimilarity.