



From Bisimilarity to ϵ -Bisimilarity

Ana Sokolova

Why formal methods?

Therac-25

Therac-25 is a computer-controlled [radiation therapy](#) machine produced by [Atomic Energy of Canada Limited](#) (AECL) in 1982 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]

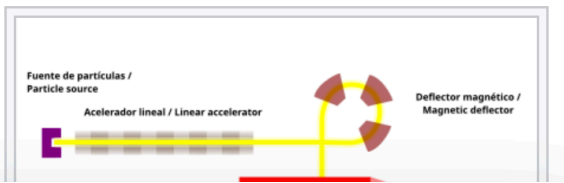
The Therac-25 was involved in at least six accidents between 1985 and 1987, in which some patients were given massive [overdoses of radiation](#).^{[2]:425} Because of [concurrent programming errors](#) (also known as race conditions), it sometimes gave its patients radiation doses that were hundreds 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

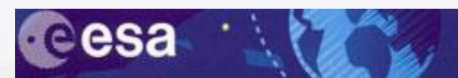
Cluster (spacecraft)" redirects here. This article is about the failed Cluster launch. For the successful space mission, see Cluster II (spacecraft).

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



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* magazine estimated that 1 in 9 billion floating-point divides with random parameters would



66 MHz Intel Pentium

Why formal methods?

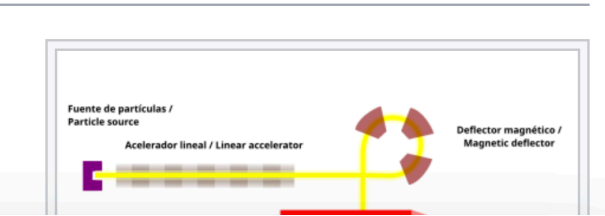
Therac-25

 **Therac-25** is a computer-controlled [radiation therapy](#) machine produced by [Atomic Energy of Canada Limited](#) (AECL) in 1982 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]

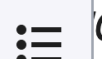
The Therac-25 was involved in at least six accidents between 1985 and 1987, in which some patients were given massive [overdoses of radiation](#).^{[2]:425} Because of [concurrent programming errors](#) (also known as race conditions), it sometimes gave its patients radiation doses that were hundreds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

Patients were seriously injured and died

... and [computer ethics](#). It highlights ... leading to severe



Ariane-5

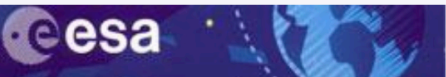
 *[Cluster \(spacecraft\)](#)" redirects here. This article is about the failed Cluster launch. For the successful space mission, see [Cluster II \(spacecraft\)](#).*

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



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* magazine estimated that 1 in 9 billion floating-point divides with random parameters would



Why formal methods?

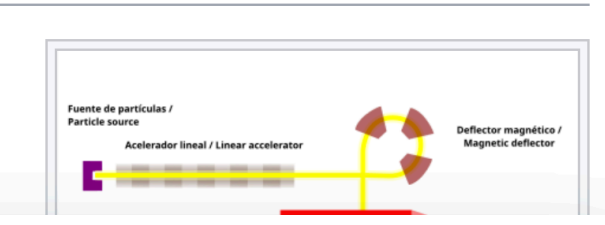
Therac-25

 **Therac-25** is a computer-controlled [radiation therapy](#) machine produced by [Atomic Energy of Canada Limited](#) (AECL) in 1982 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.

Patients were seriously injured and died

... and [computer ethics](#). It highlights ... leading to severe



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



Intel Pentium

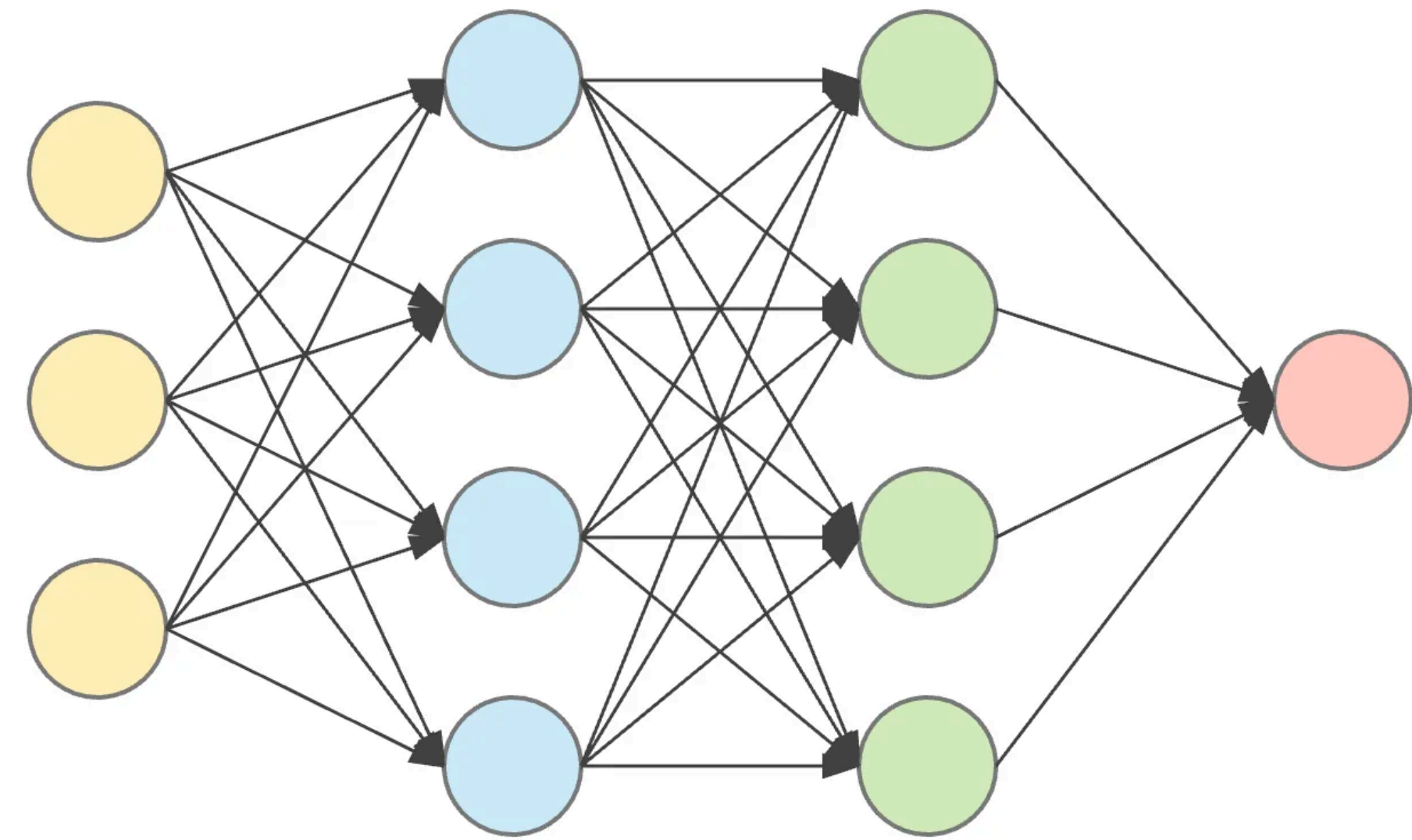
 "Pentium bug" redirects here. For the 1997 bug affecting Pentium processors, see [Pentium F00F bug](#).

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* magazine estimated that 1 in 9 billion floating-point divides with random parameters would



Why formal methods?



input layer

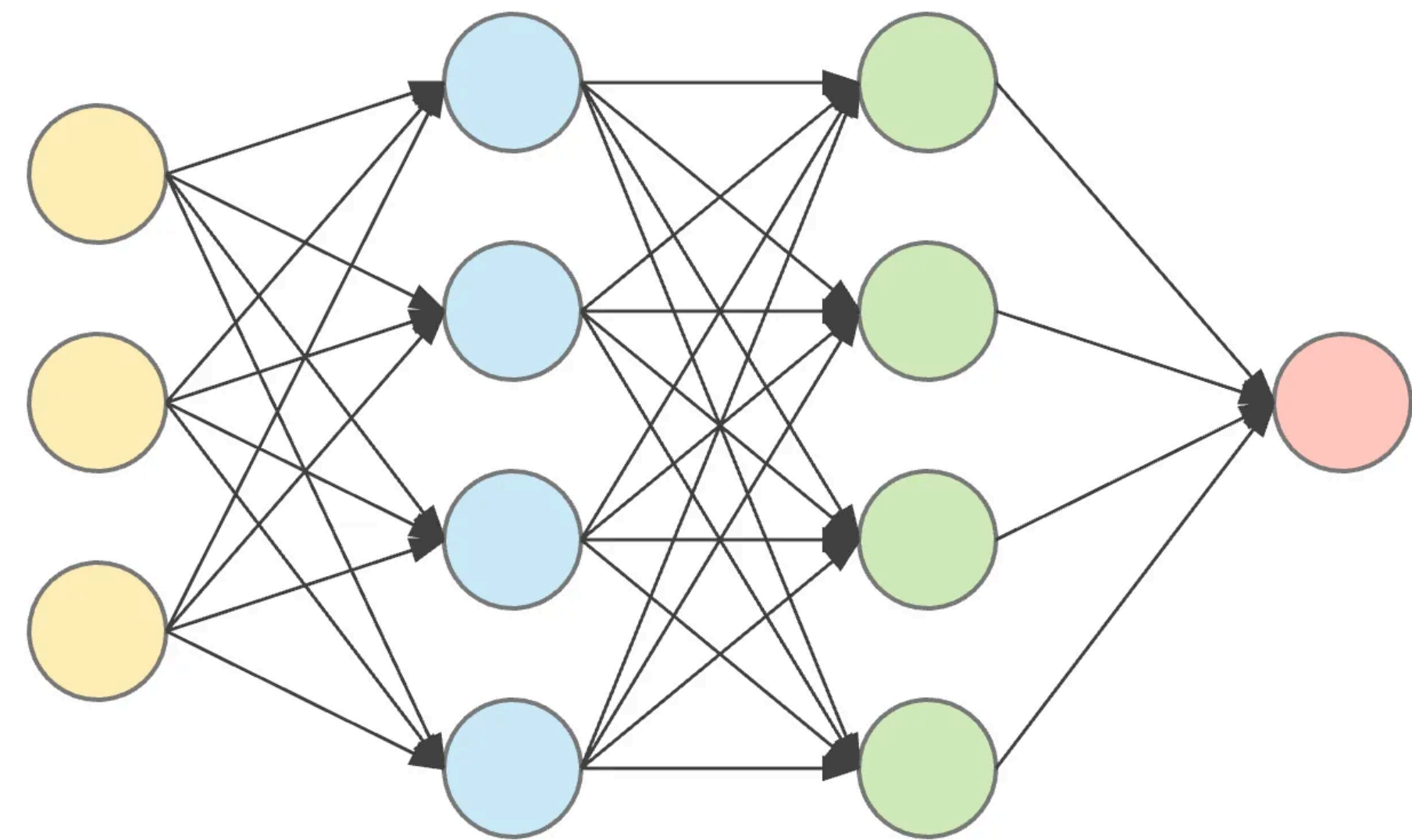
hidden layer 1

hidden layer 2

output layer



Why formal methods?



input layer

hidden layer 1

hidden layer 2

output layer



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?

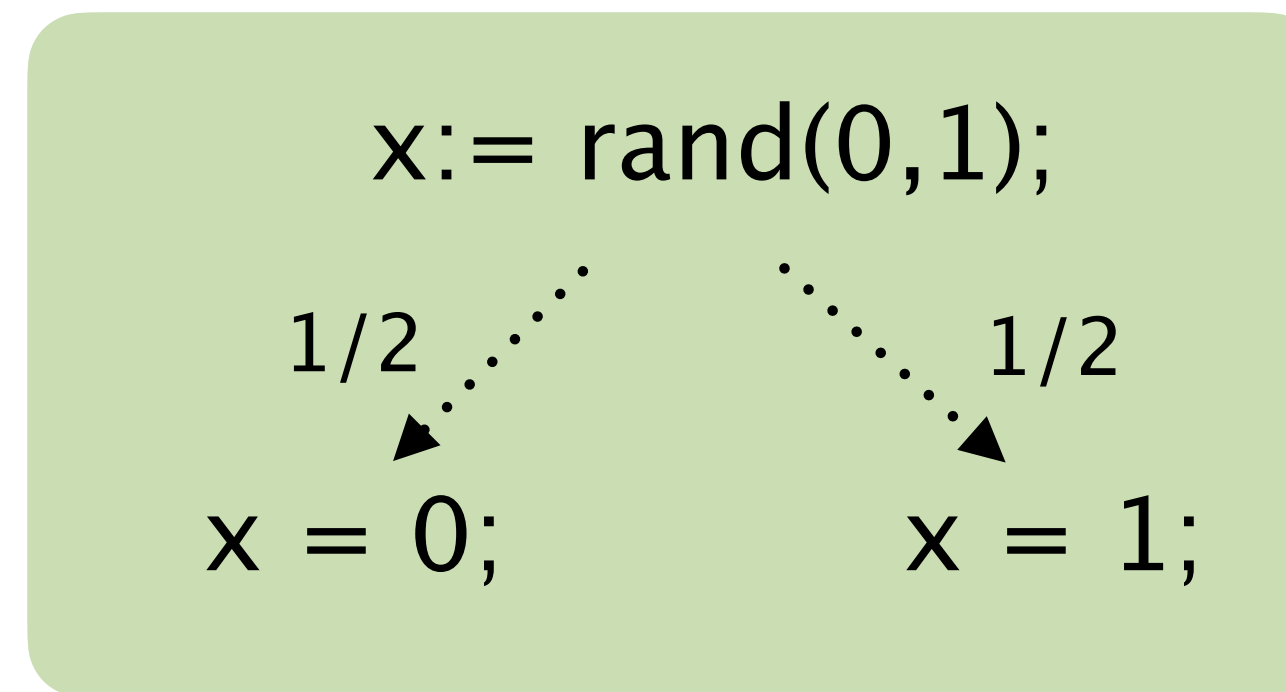
In my work: with behavioural equivalences, like bisimilarity or trace semantics

probabilistic programs

How do formal methods help?

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

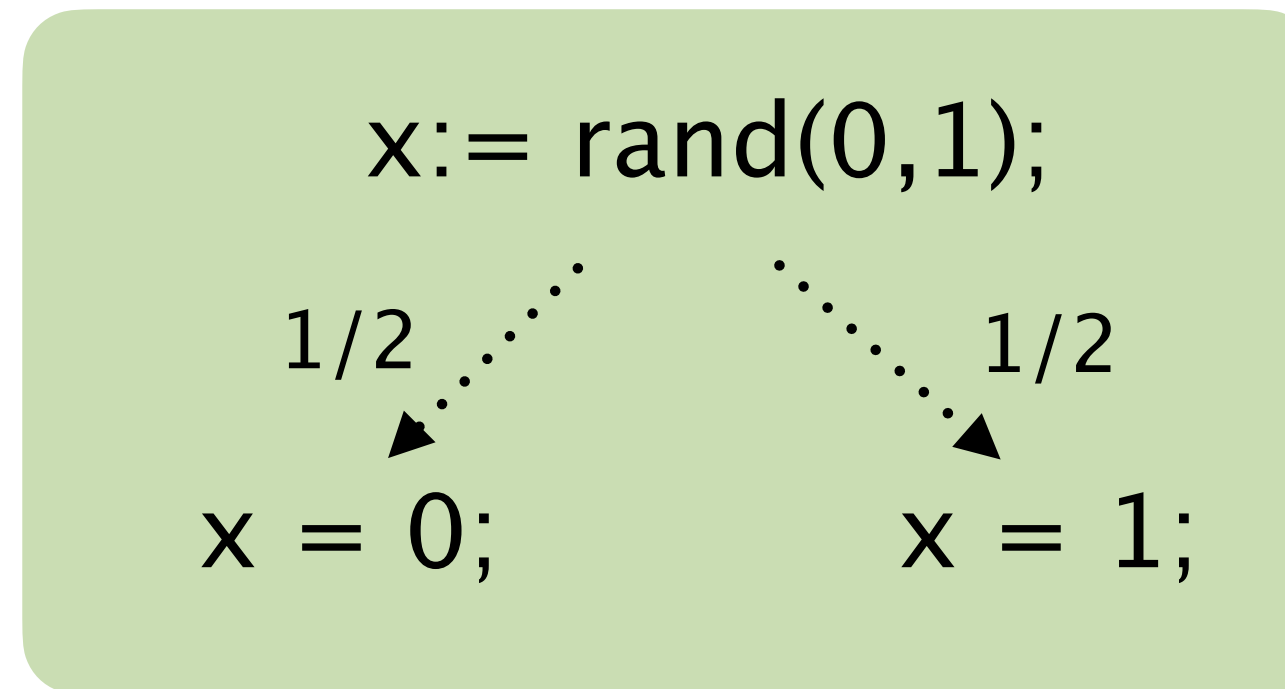
probabilistic programs



How do formal methods help?

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

probabilistic programs

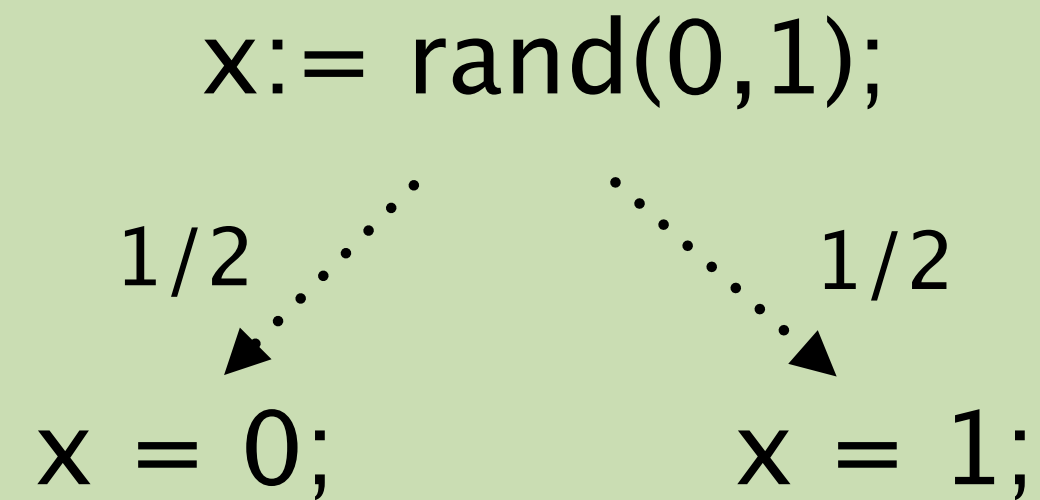


verifying **equivalence** of programs

How do formal methods help?

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

probabilistic programs



verifying **equivalence** of programs

$P \equiv \text{Optimised}(P)$

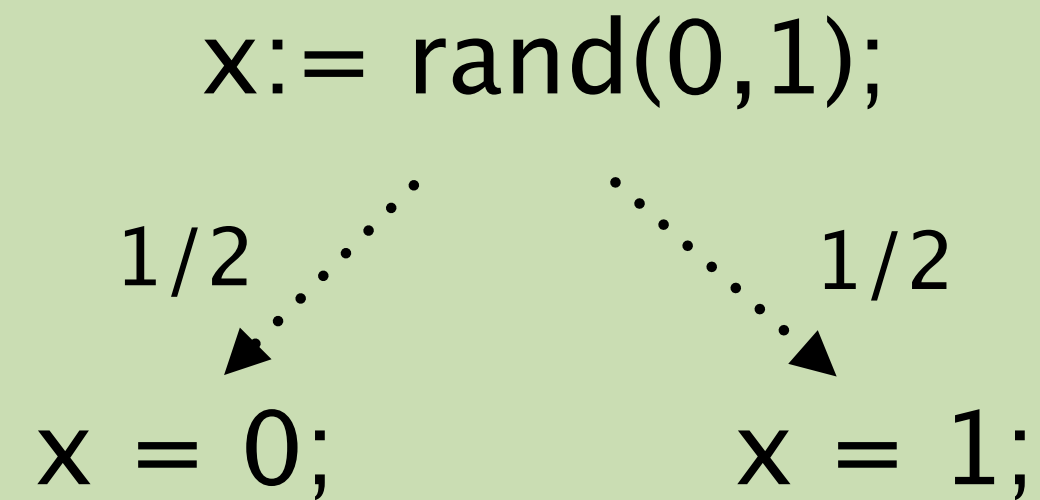
$\text{Source} \equiv \text{Compiled}$

$\text{Implementation} \equiv \text{Specification}$

How do formal methods help?

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

probabilistic programs



verifying **equivalence** of programs

$P \equiv \text{Optimised}(P)$

$\text{Source} \equiv \text{Compiled}$

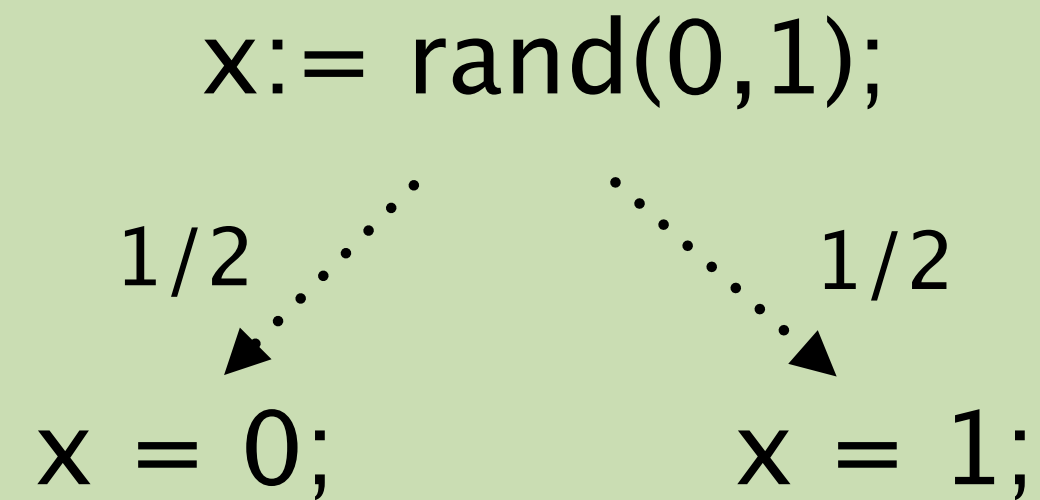
$\text{Implementation} \equiv \text{Specification}$

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

$P \equiv \text{Optimised}(P)$

$\text{Source} \equiv \text{Compiled}$

$\text{Implementation} \equiv \text{Specification}$

developing **proof techniques**

e.g. by axiomatisations / up-to techniques

Coalgebras and Bisimilarity

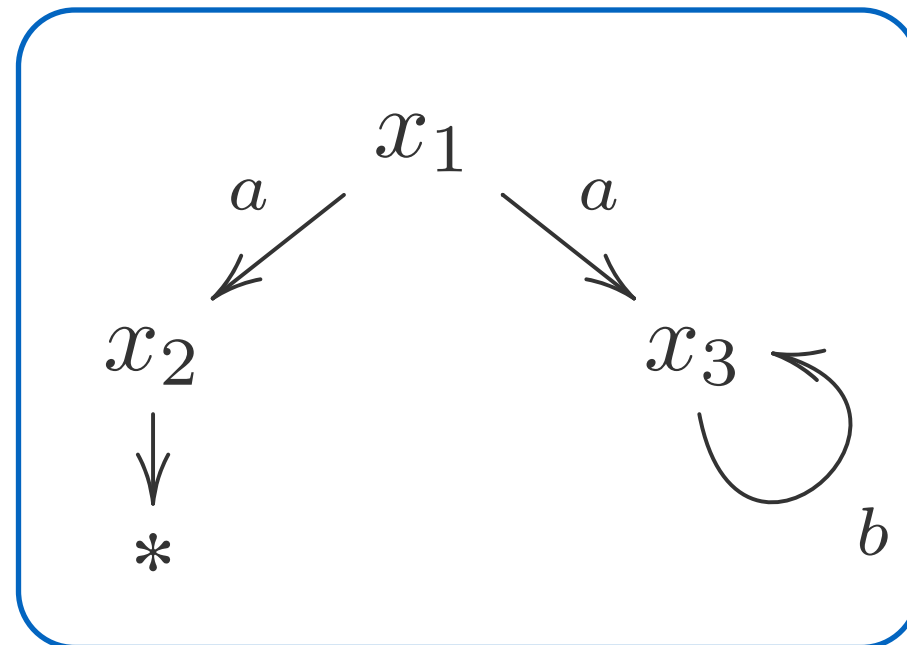


Some coalgebras

Some coalgebras

NFA

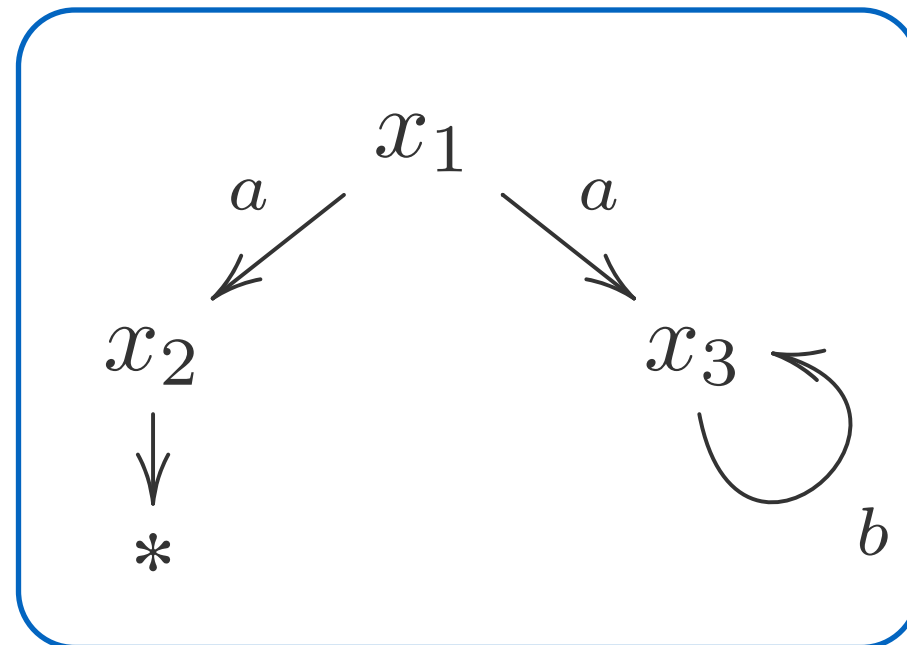
$$X \rightarrow 2 \times (\mathcal{P}X)^A$$



Some coalgebras

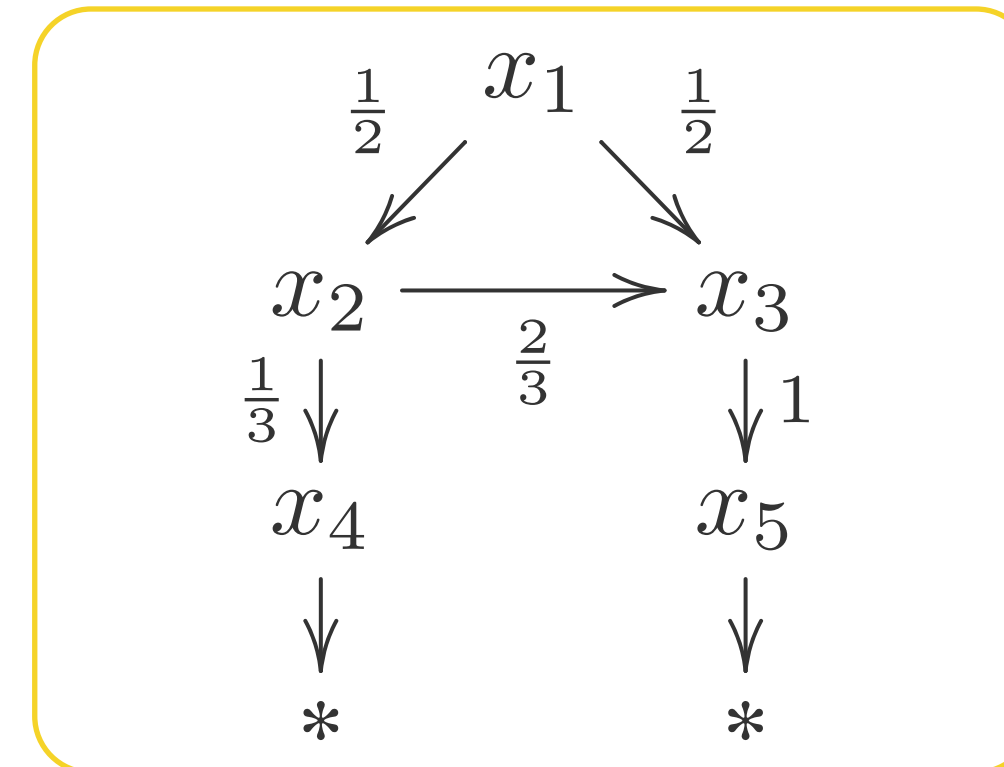
NFA

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



MC

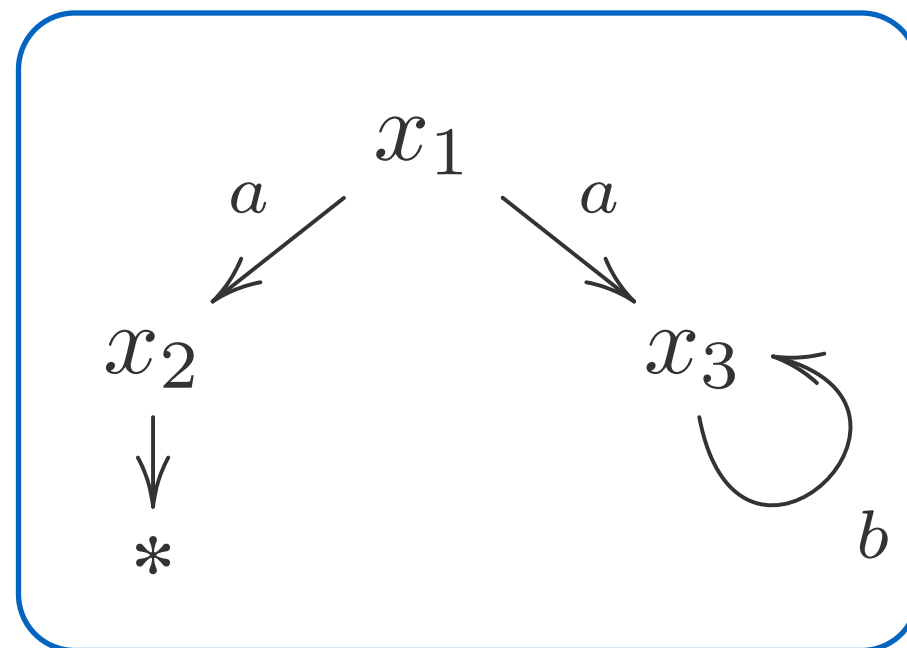
$$X \rightarrow \mathcal{D}X + I$$



Some coalgebras

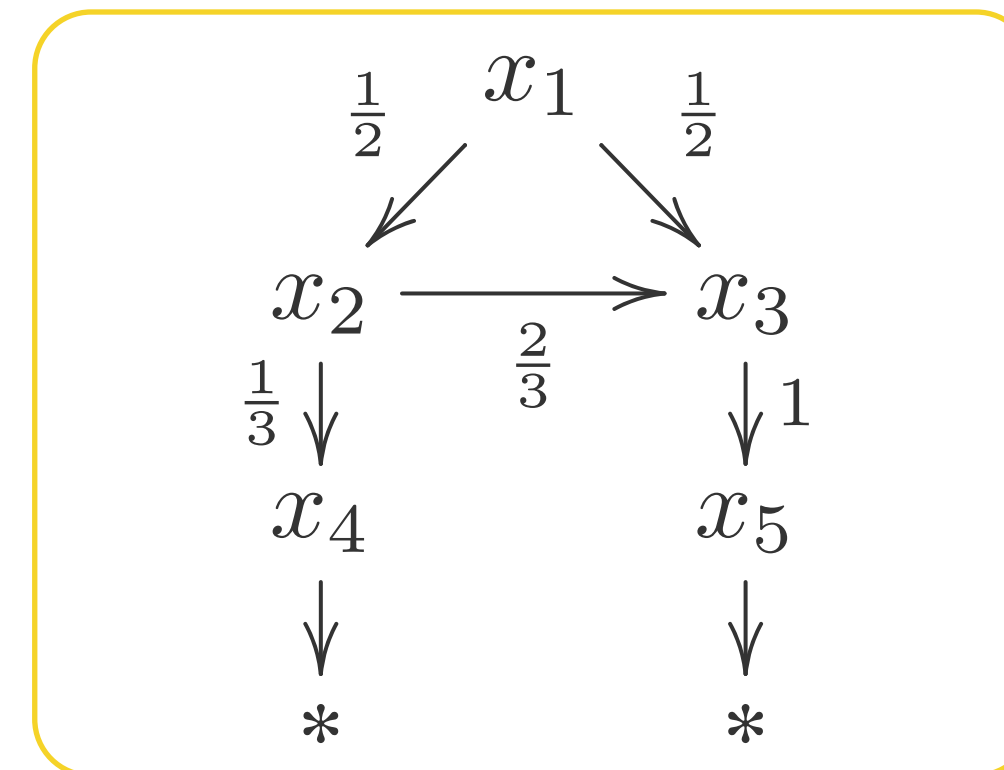
NFA

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



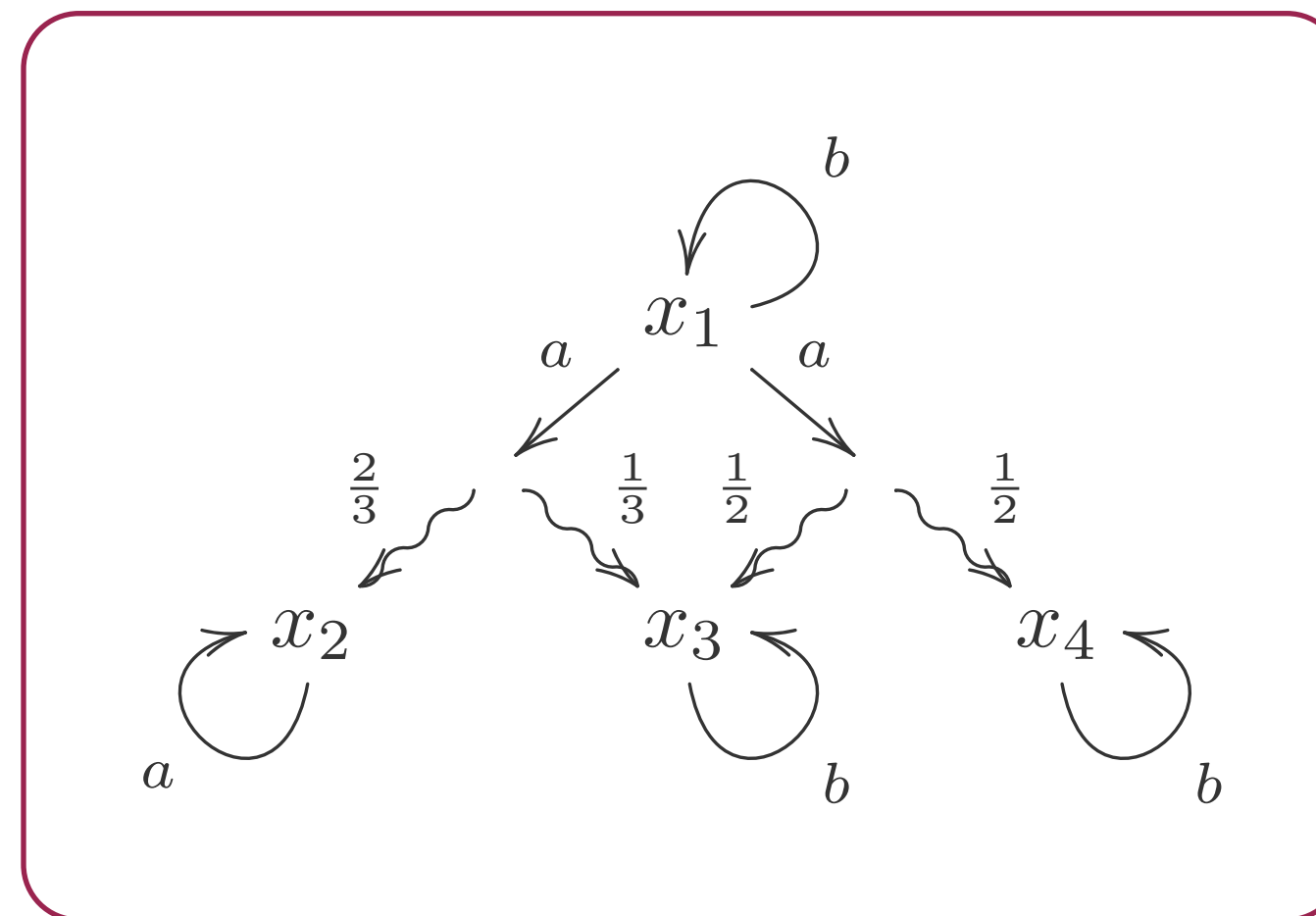
MC

$$X \rightarrow \mathcal{D}X + I$$



PA

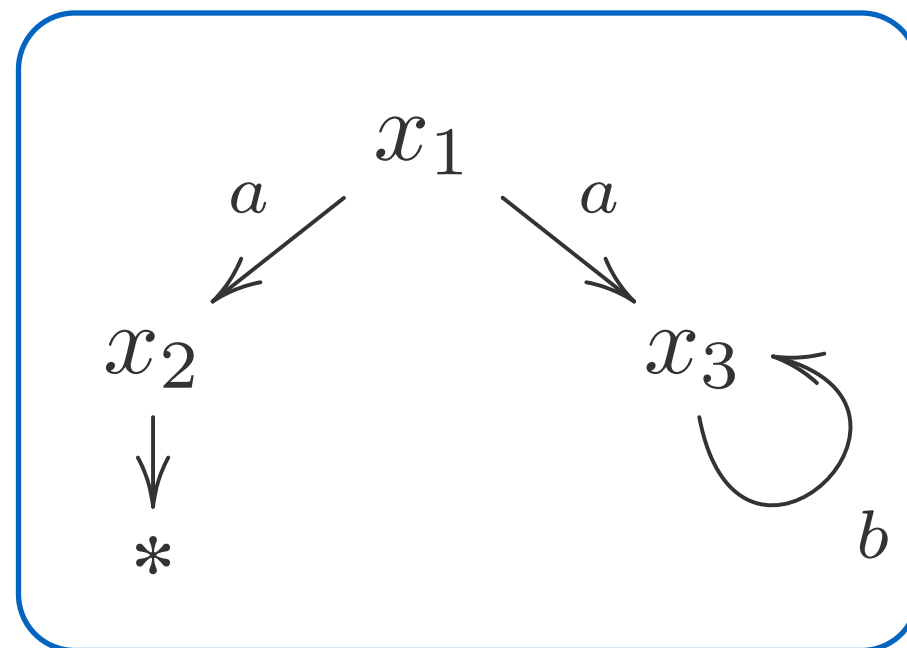
$$X \rightarrow (\mathcal{P}\mathcal{D}X)^A$$



Some coalgebras

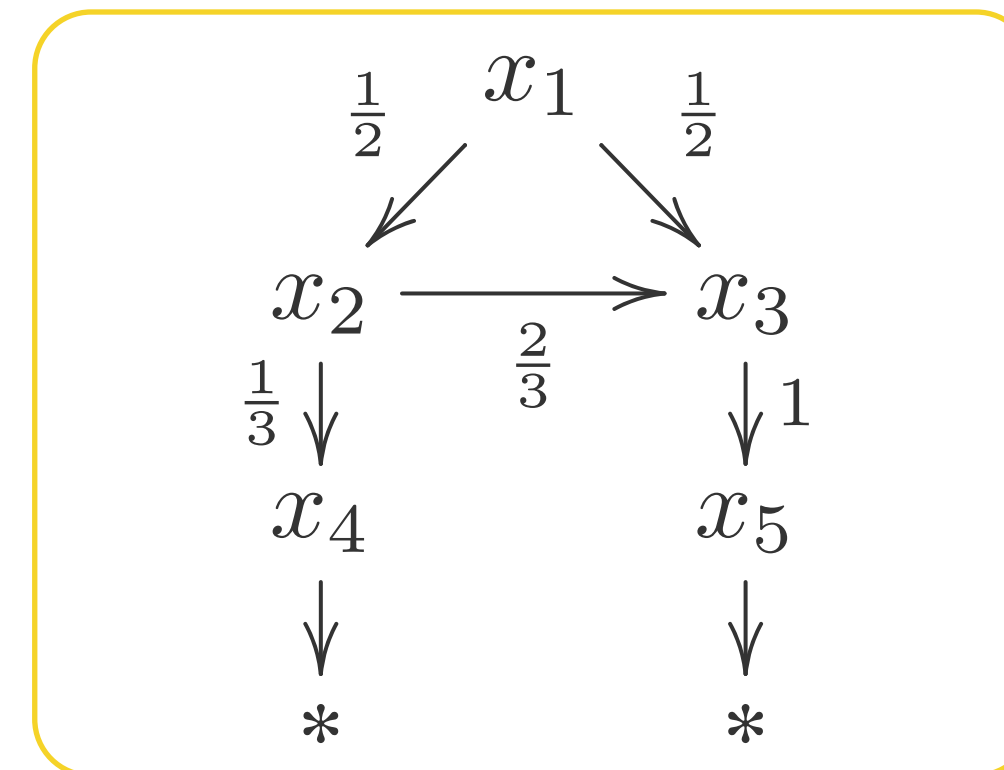
NFA

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



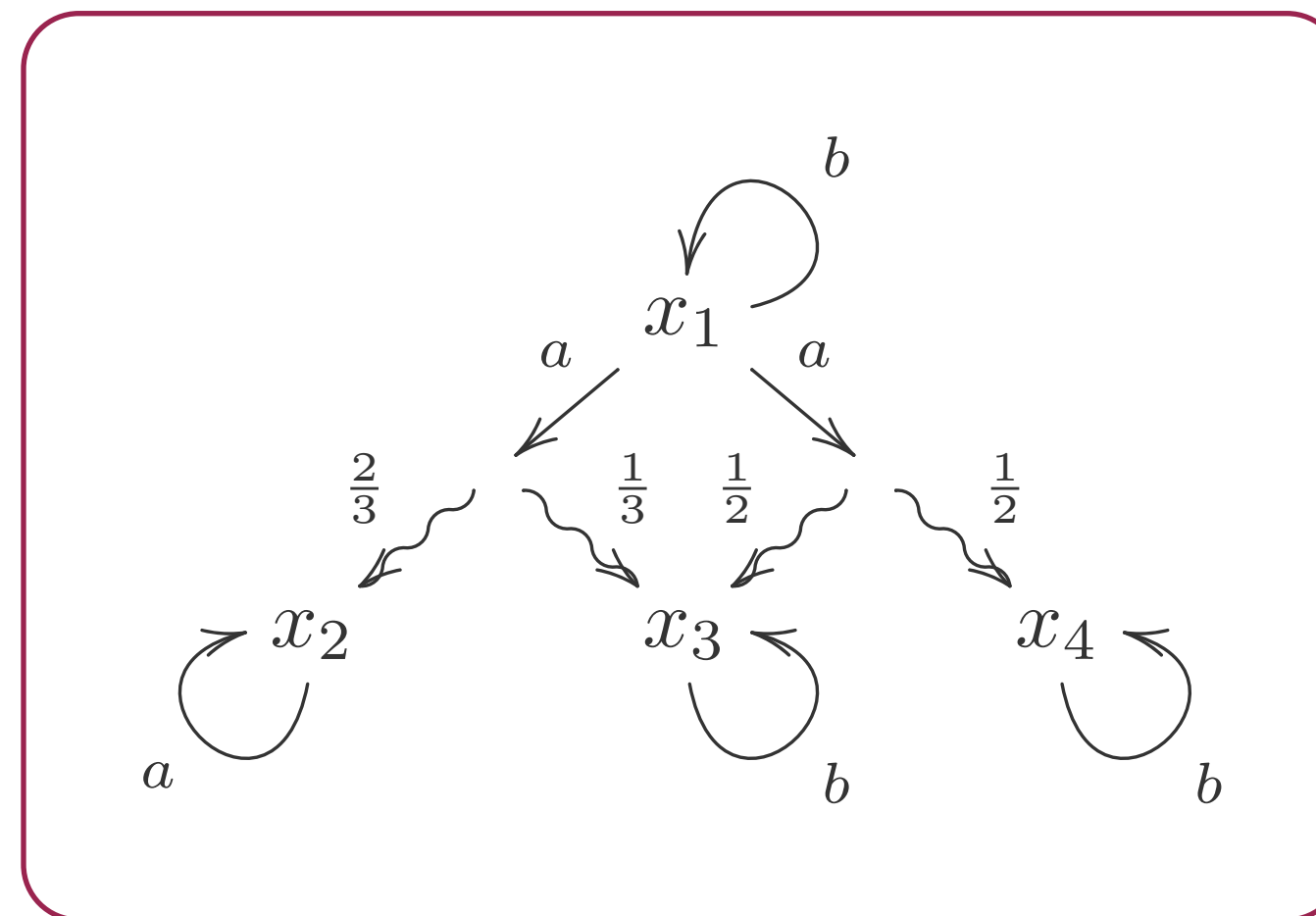
MC

$$X \rightarrow \mathcal{D}X + I$$



PA

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



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:

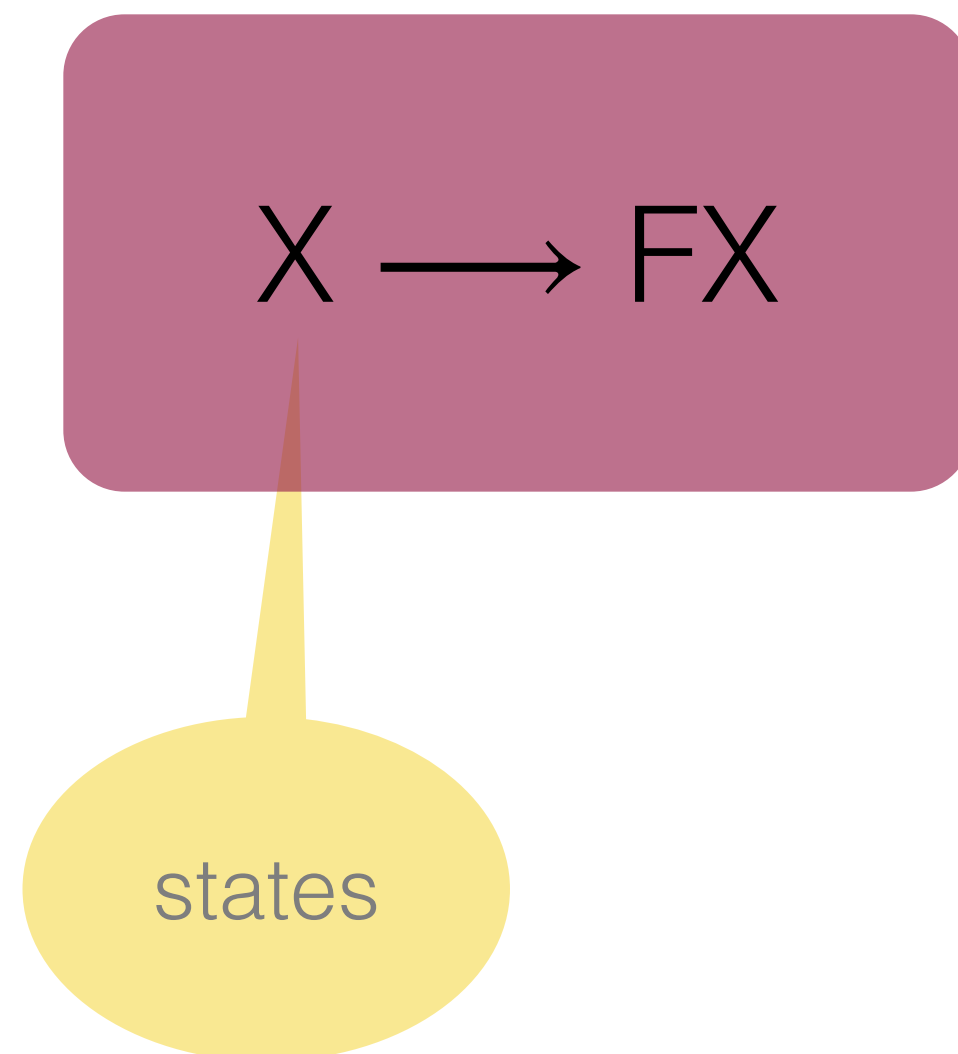
$$X \longrightarrow FX$$



Coalgebra

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

A coalgebra is a generic transition system:

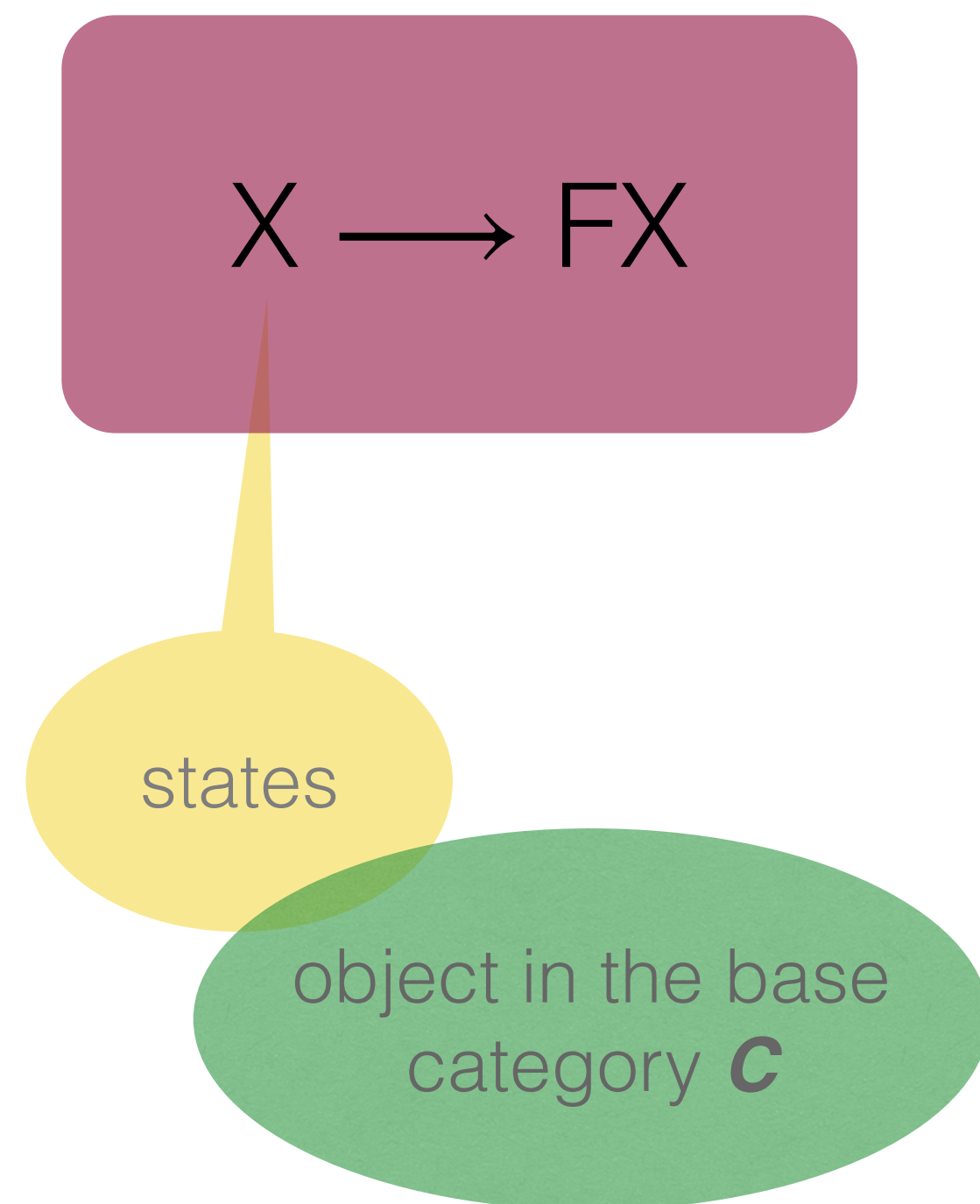




Coalgebra

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

A coalgebra is a generic transition system:

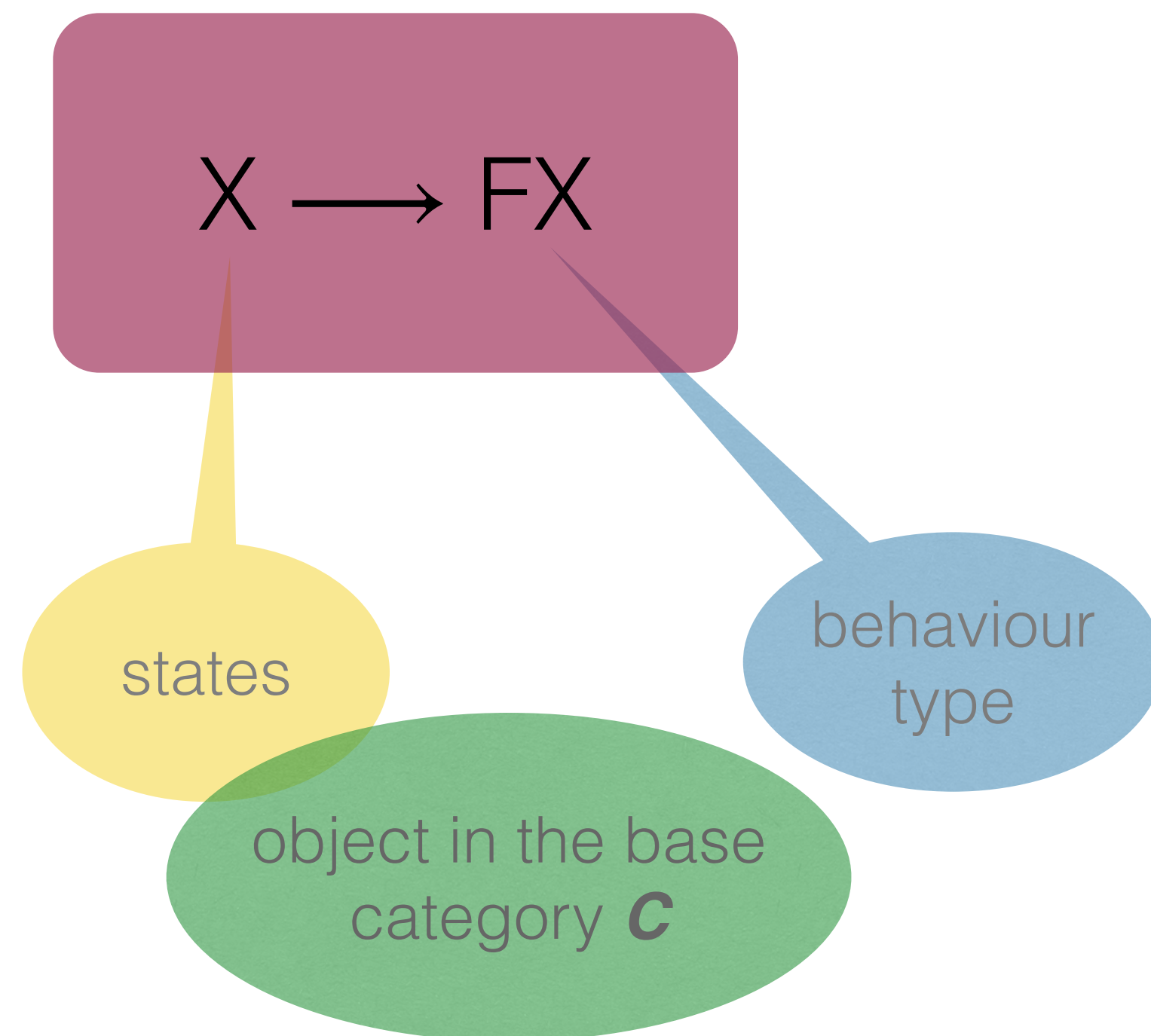




Coalgebra

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

A coalgebra is a generic transition system:

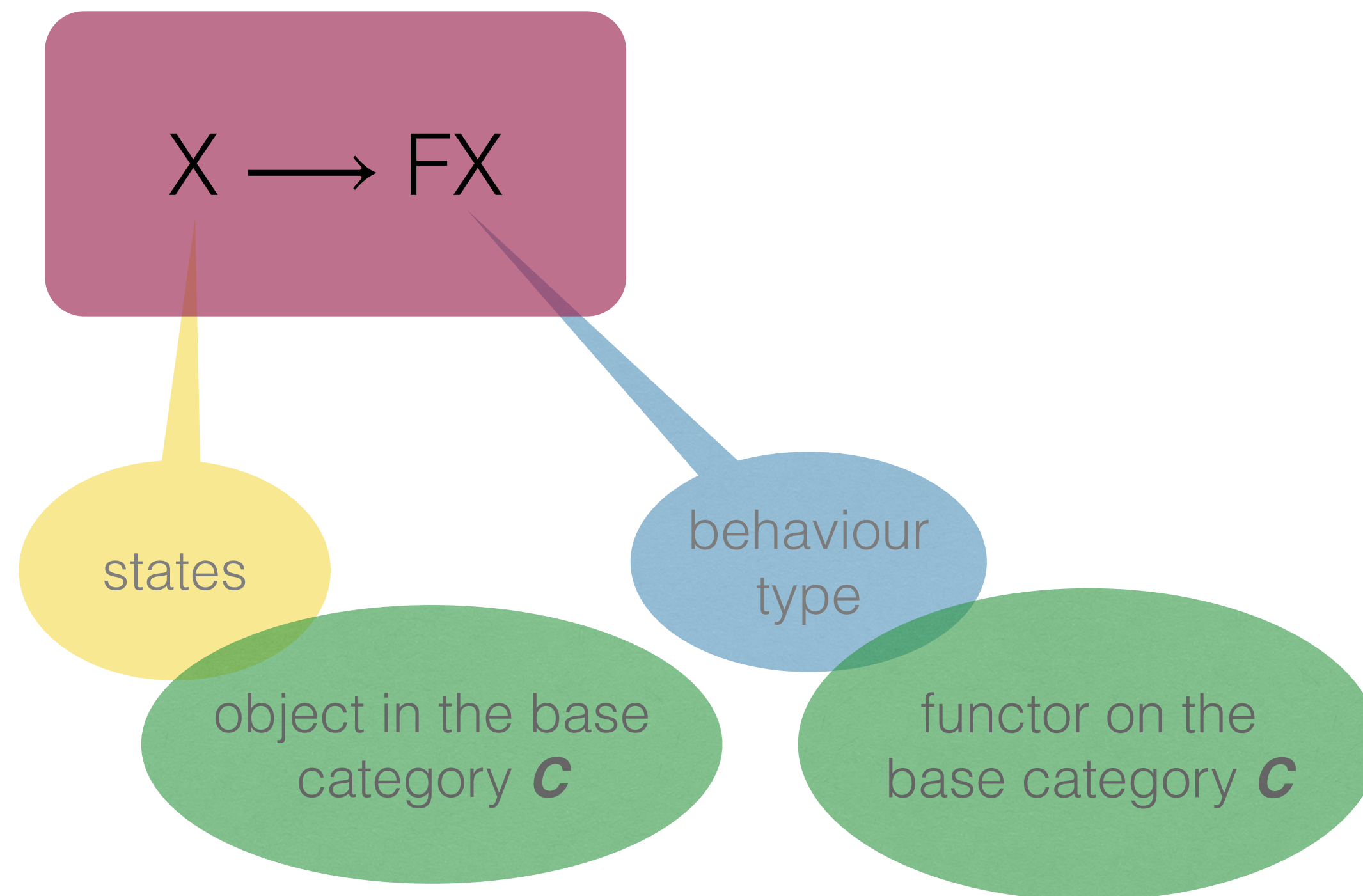




Coalgebra

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

A coalgebra is a generic transition system:

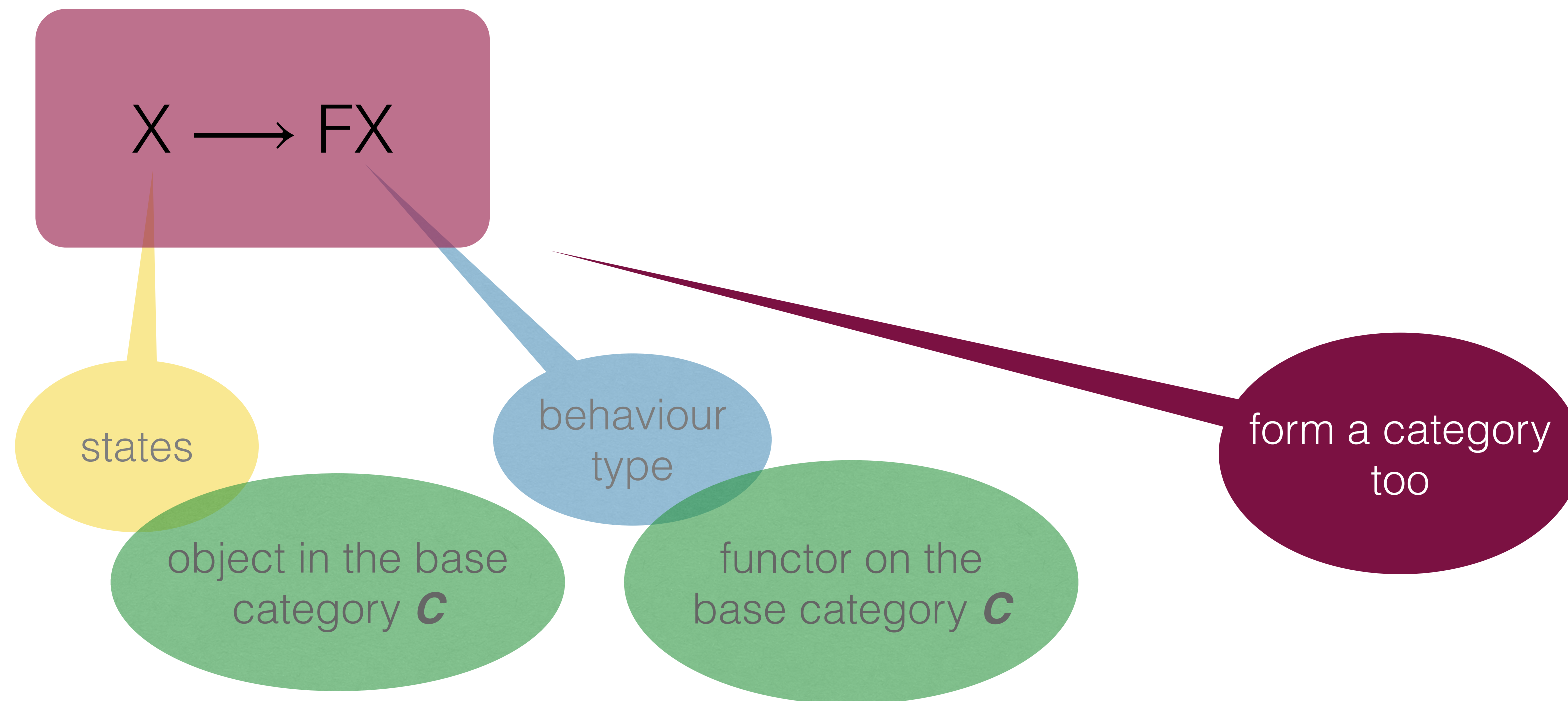




Coalgebra

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

A coalgebra is a generic transition system:

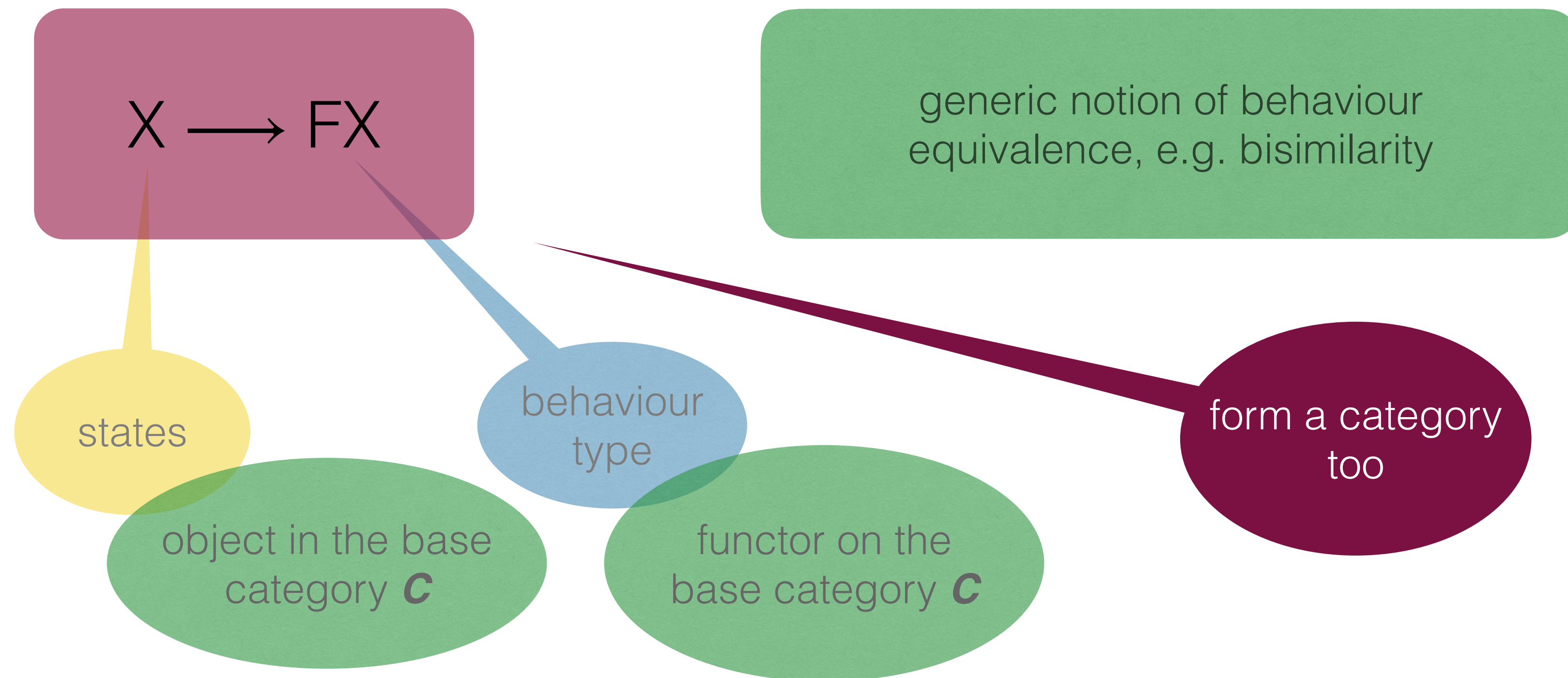




Coalgebra

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

A coalgebra is a generic transition system:



Bisimilarity of Labelled Transition Systems

$$S \rightarrow (\mathcal{P} S)^A$$

Bisimilarity of Labelled Transition Systems

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

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

$$S \rightarrow (\mathcal{P} S)^A$$

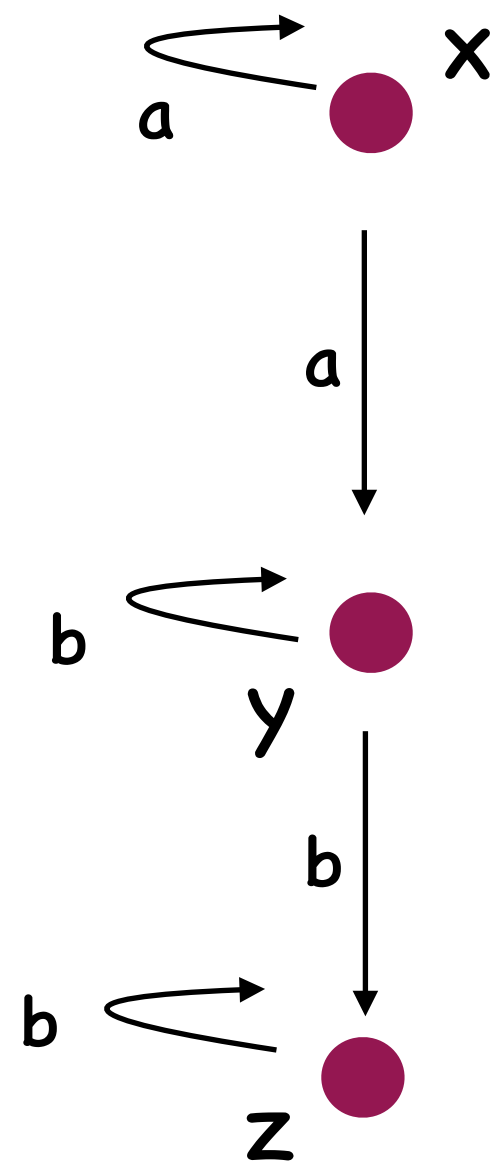
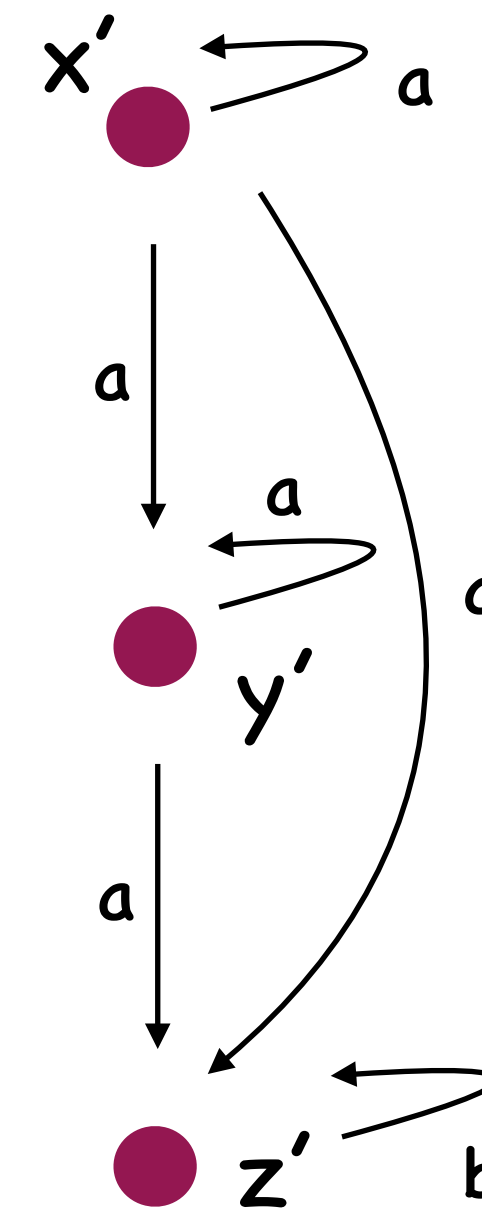
a coalgebra

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.

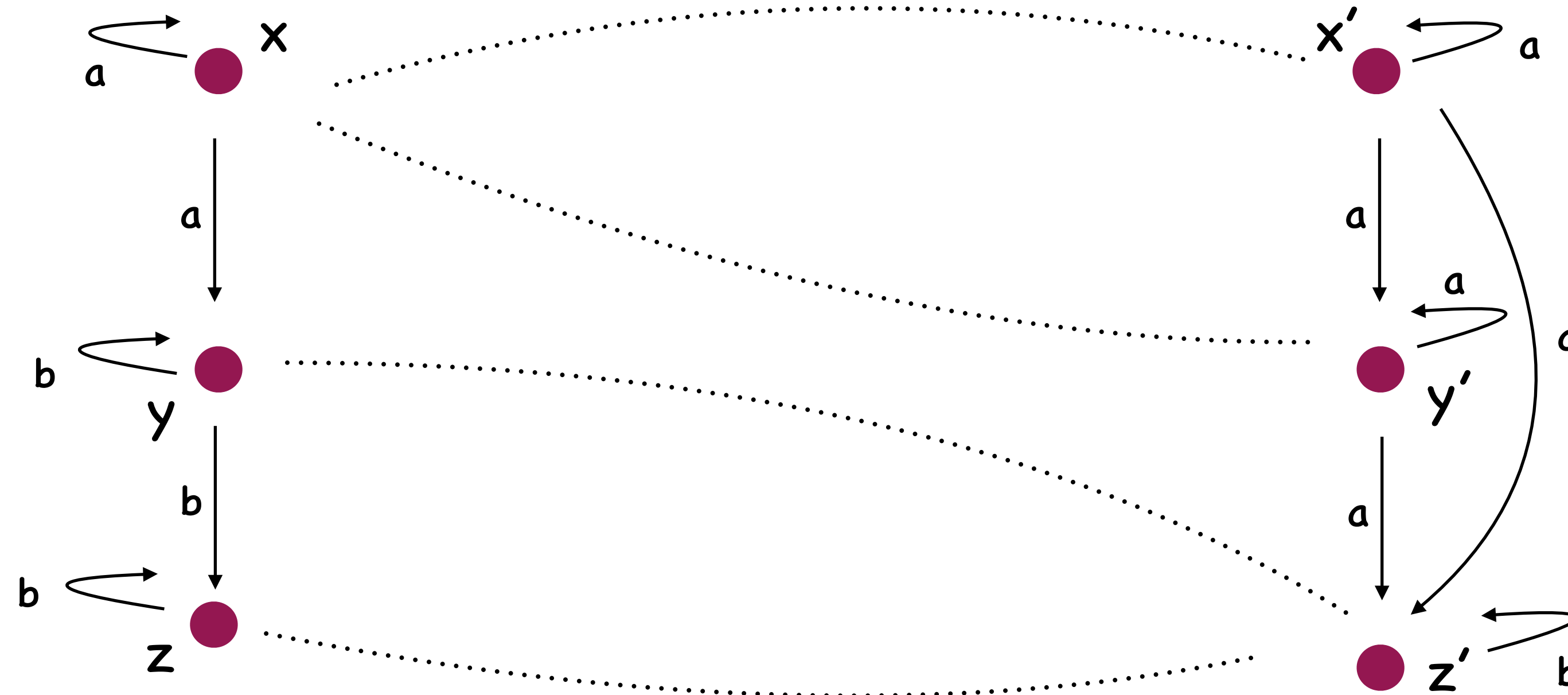
Two states are bisimilar iff they are related by a bisimulation.

Bisimilarity of Labelled Transition Systems



$$S \rightarrow (\mathcal{P} S)^A$$

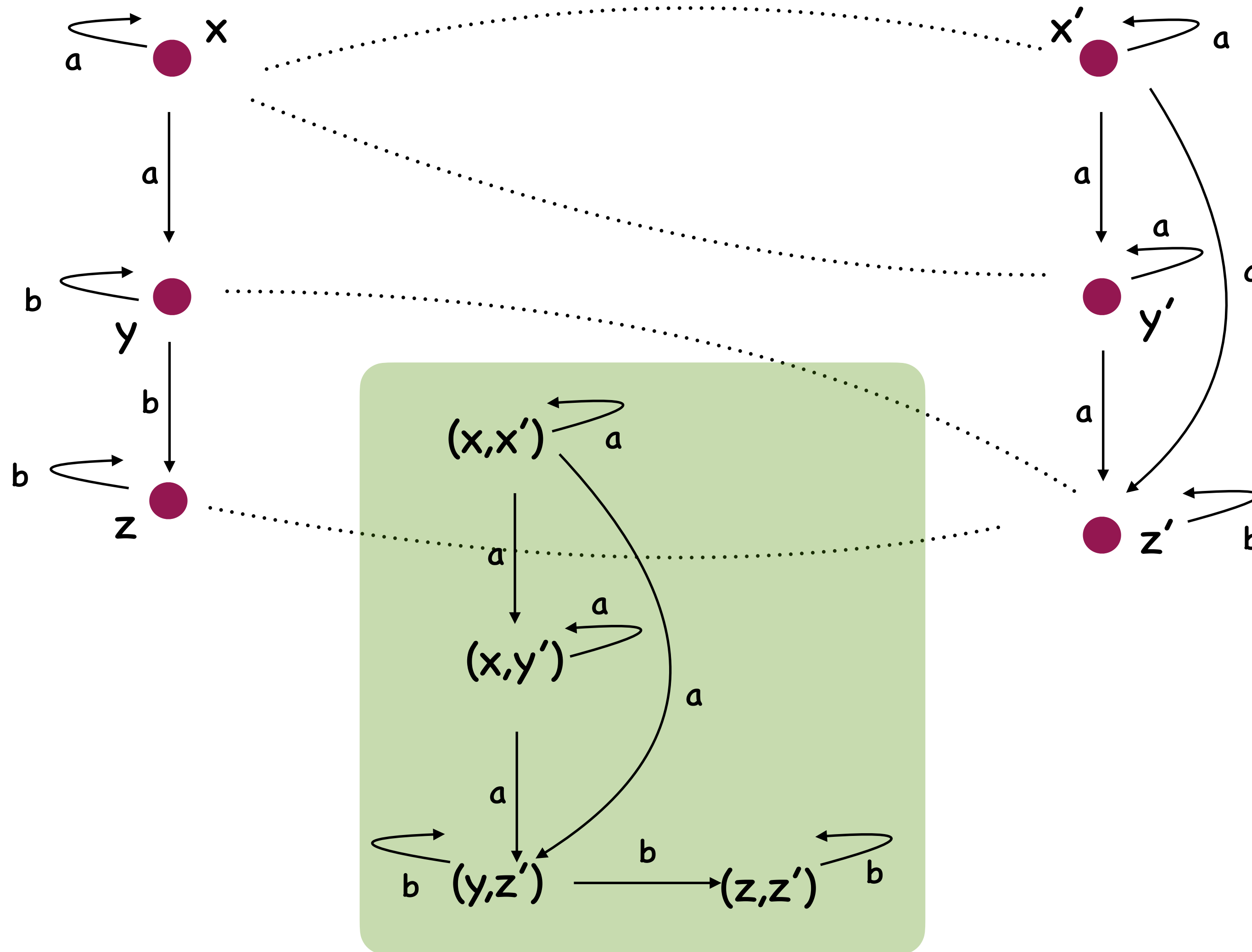
Bisimilarity of Labelled Transition Systems



$$S \rightarrow (\mathcal{P} S)^A$$

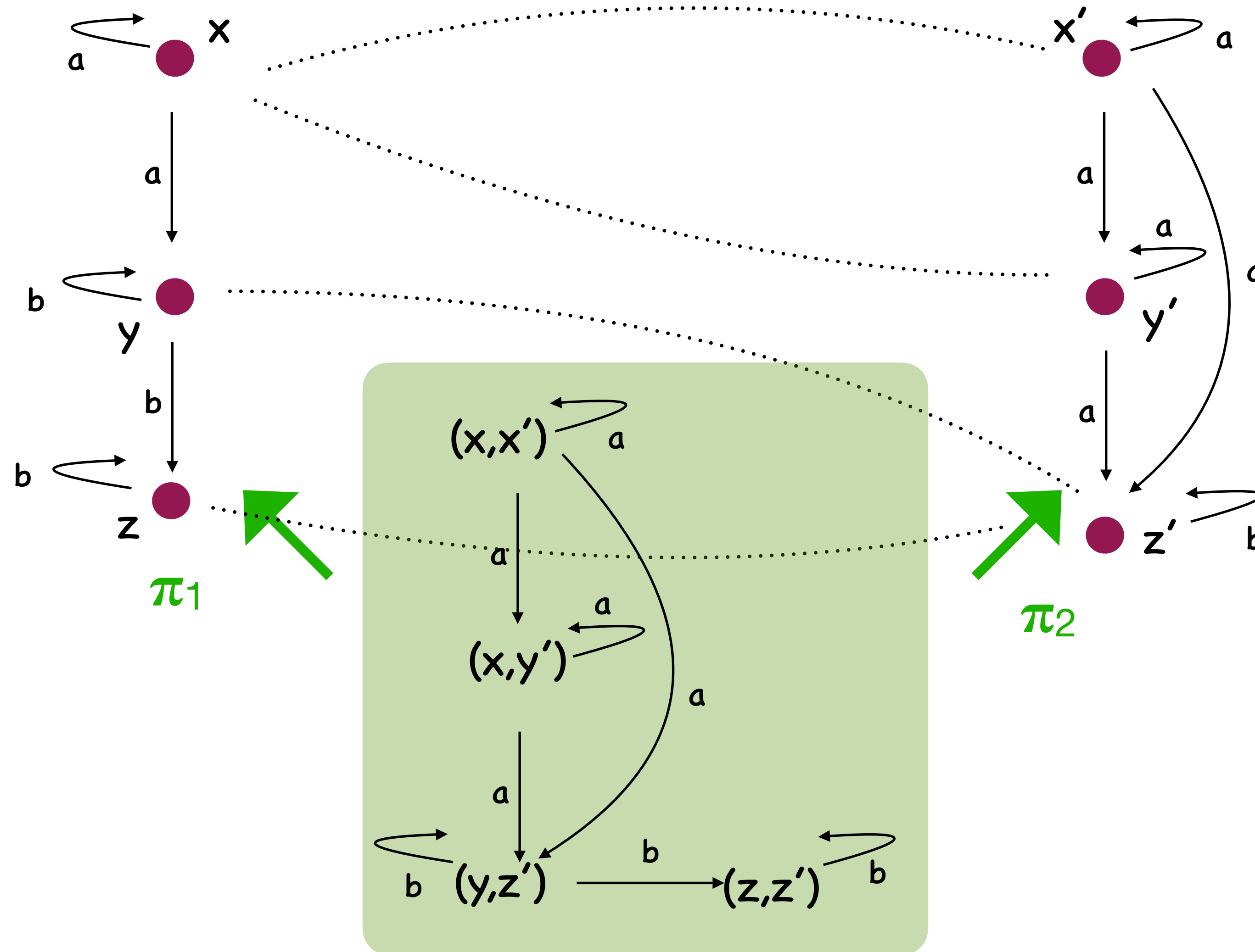
Bisimilarity of Labelled Transition Systems

$$S \rightarrow (\mathcal{P} S)^A$$



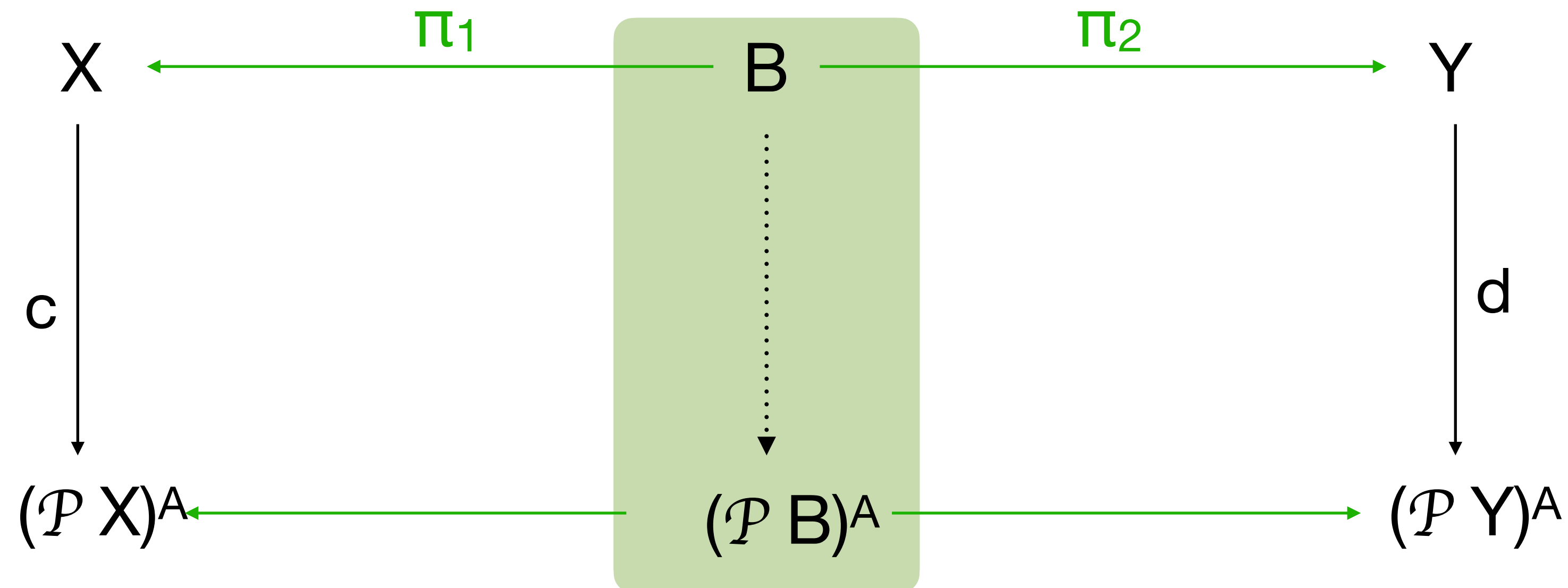
Bisimilarity of Labelled Transition Systems

$$S \rightarrow (\mathcal{P} S)^A$$



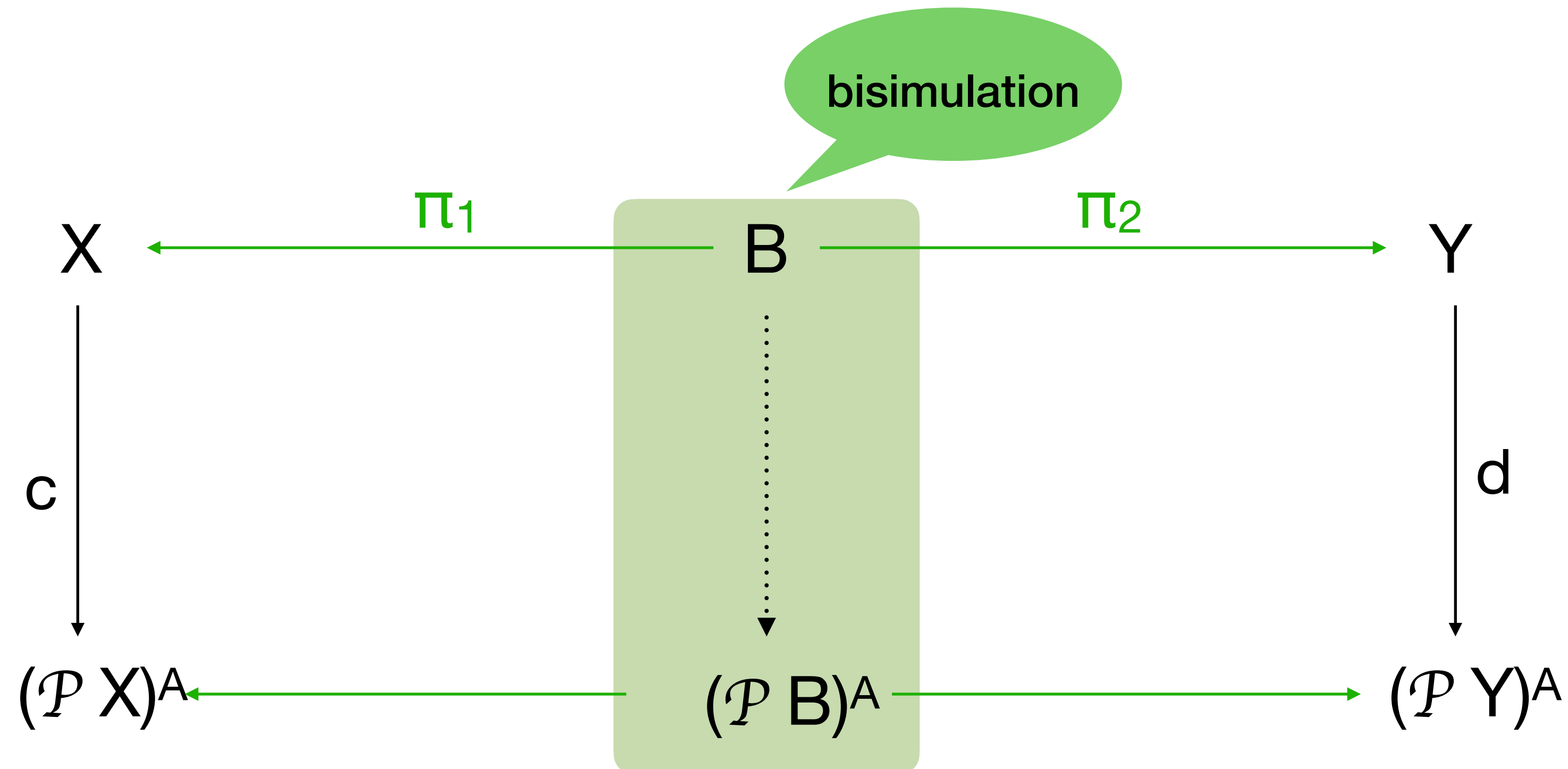
Bisimilarity of Labelled Transition Systems

$$S \rightarrow (\mathcal{P} S)^A$$



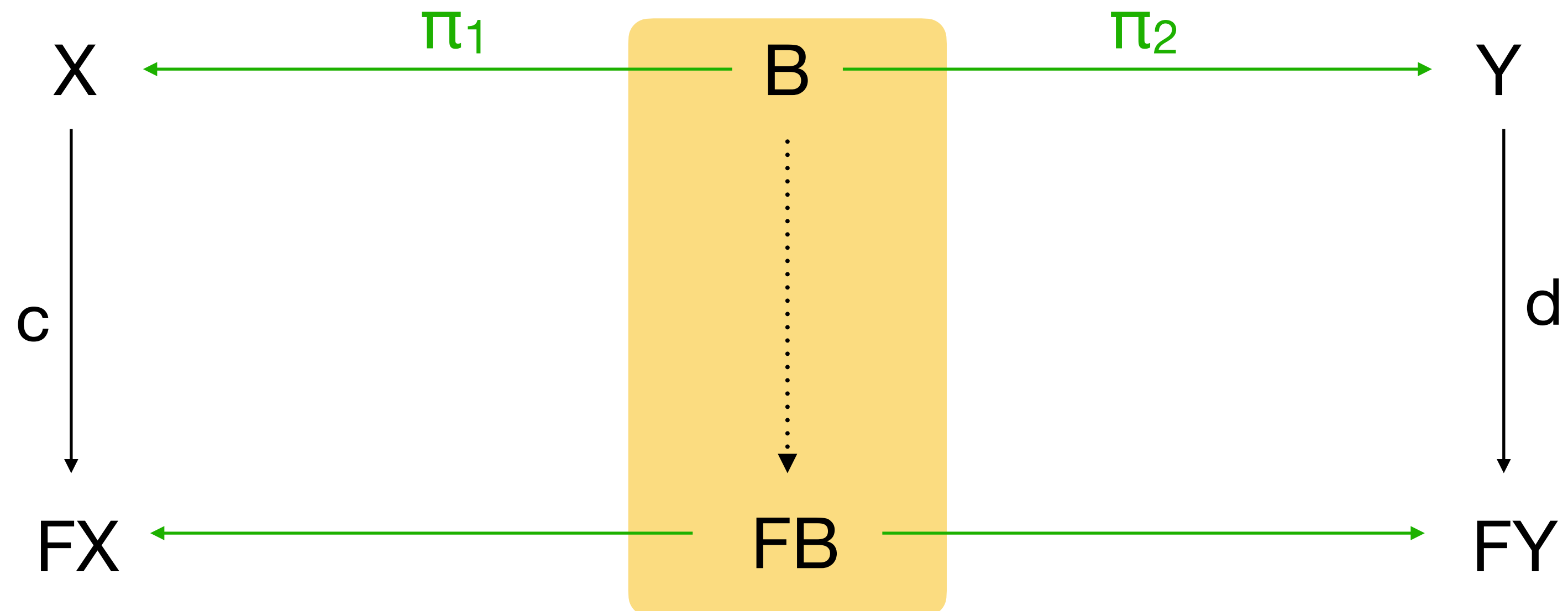
Bisimilarity of Labelled Transition Systems

$$S \rightarrow (\mathcal{P} S)^A$$



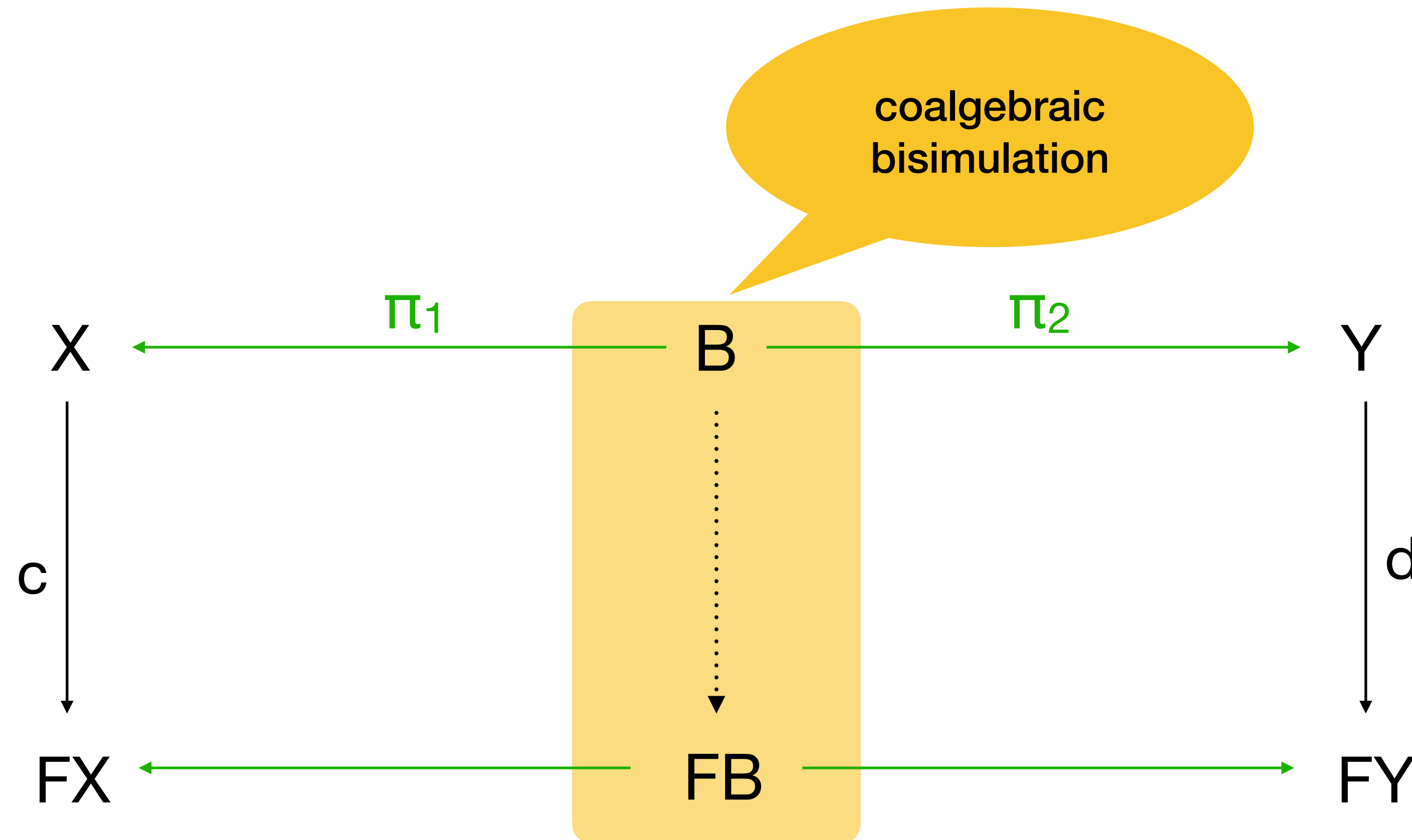
Bisimilarity of F-Coalgebras

$S \rightarrow FS$

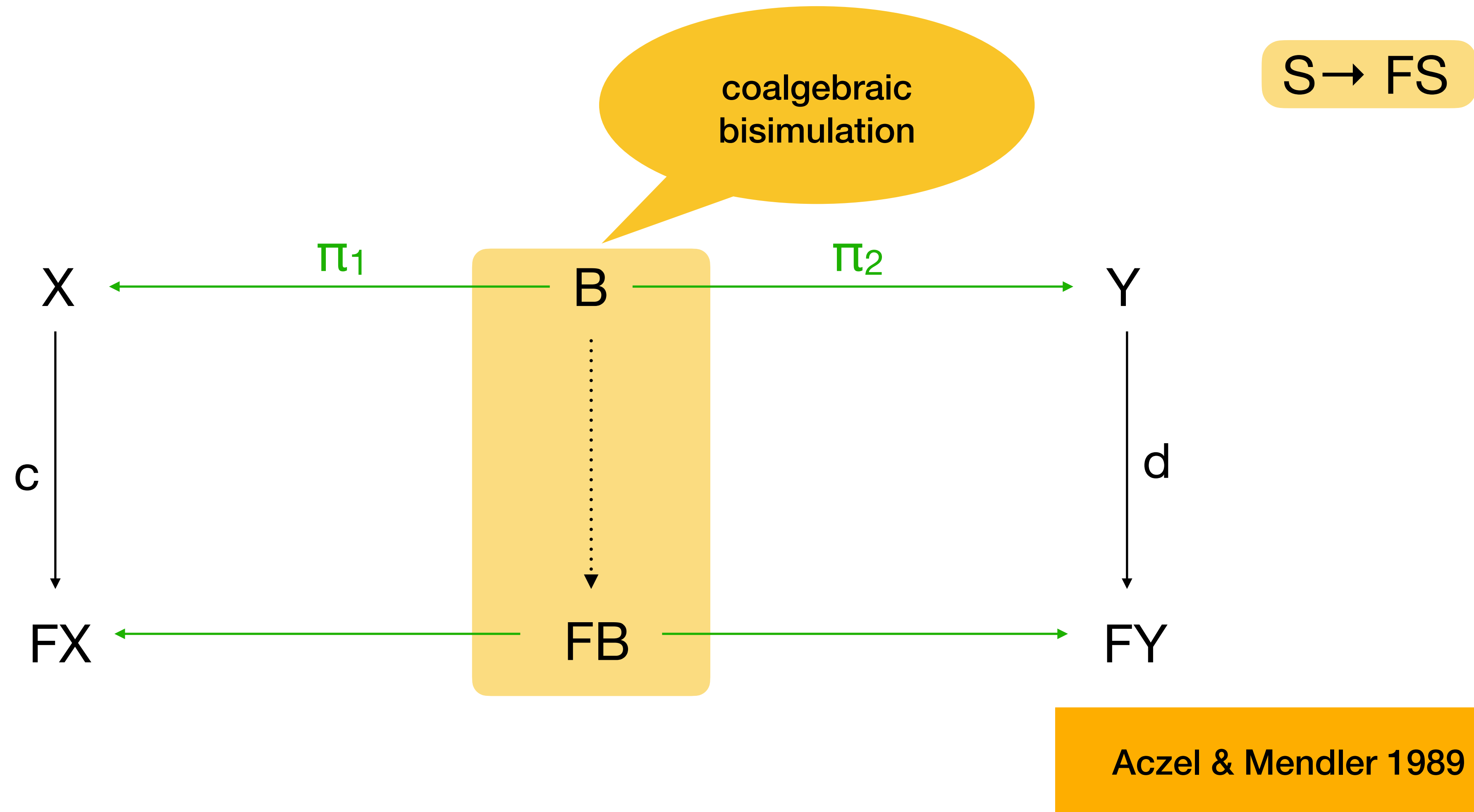


Bisimilarity of F-Coalgebras

$S \rightarrow FS$



Bisimilarity of F-Coalgebras

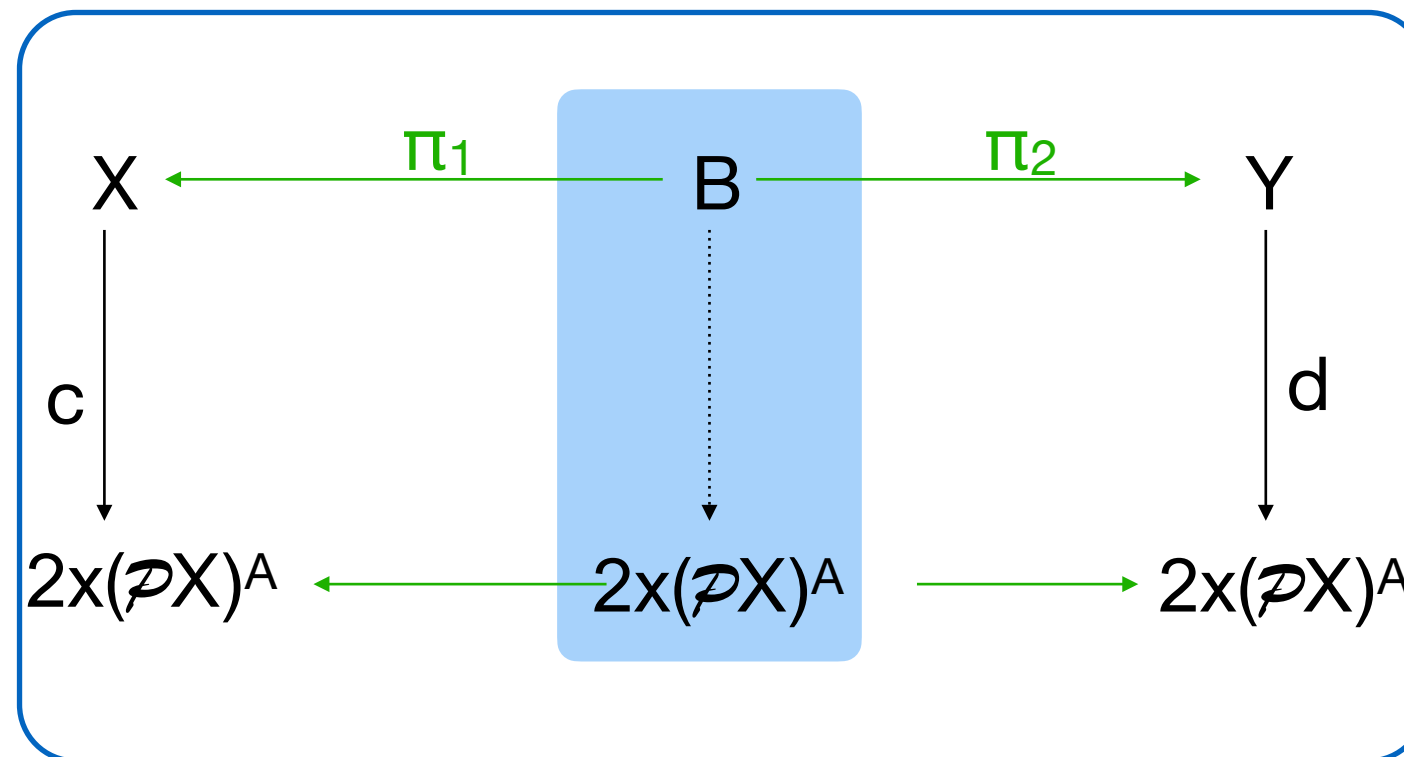


**Coalgebraic bisimilarity is
bisimilarity**

Coalgebraic bisimilarity is bisimilarity

NFA

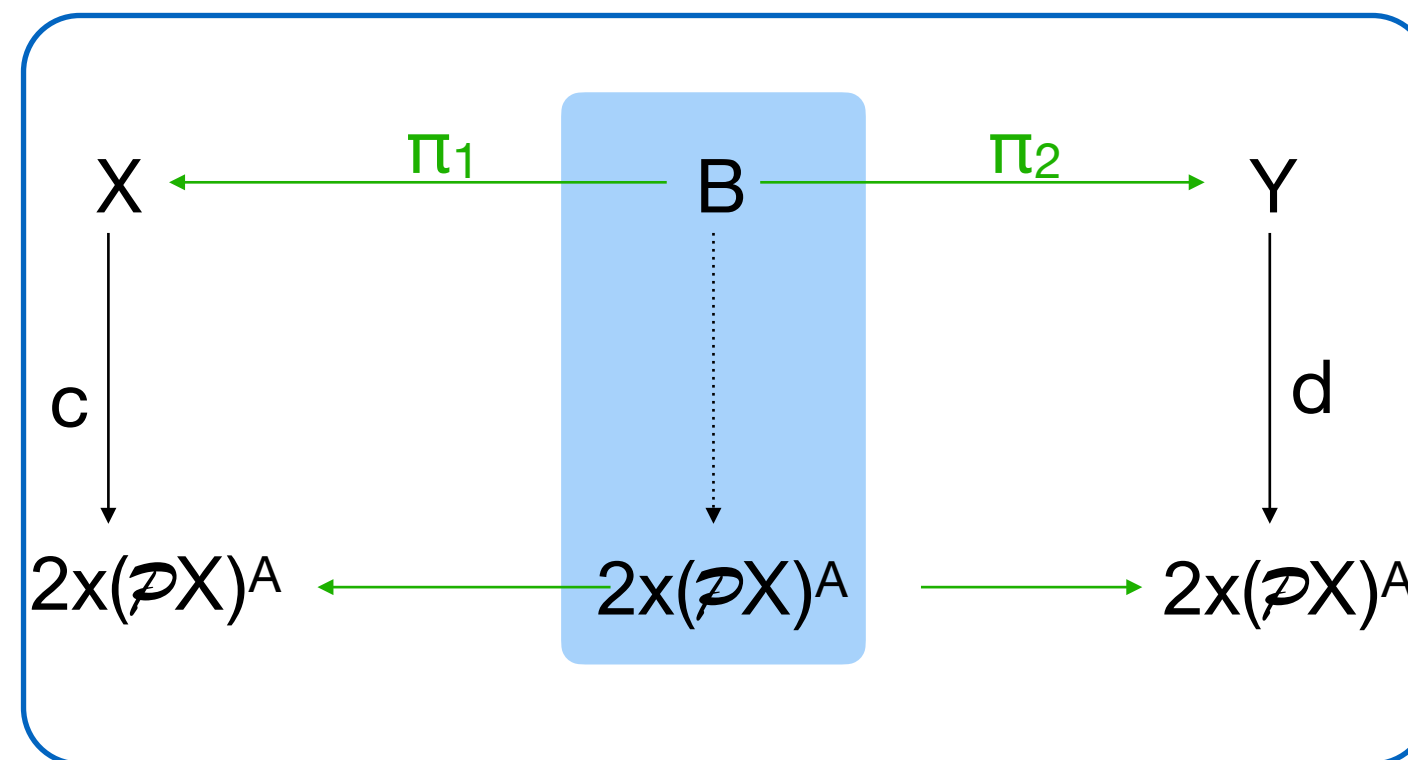
$$X \rightarrow 2 \times (\mathcal{P}X)^A$$



Coalgebraic bisimilarity is bisimilarity

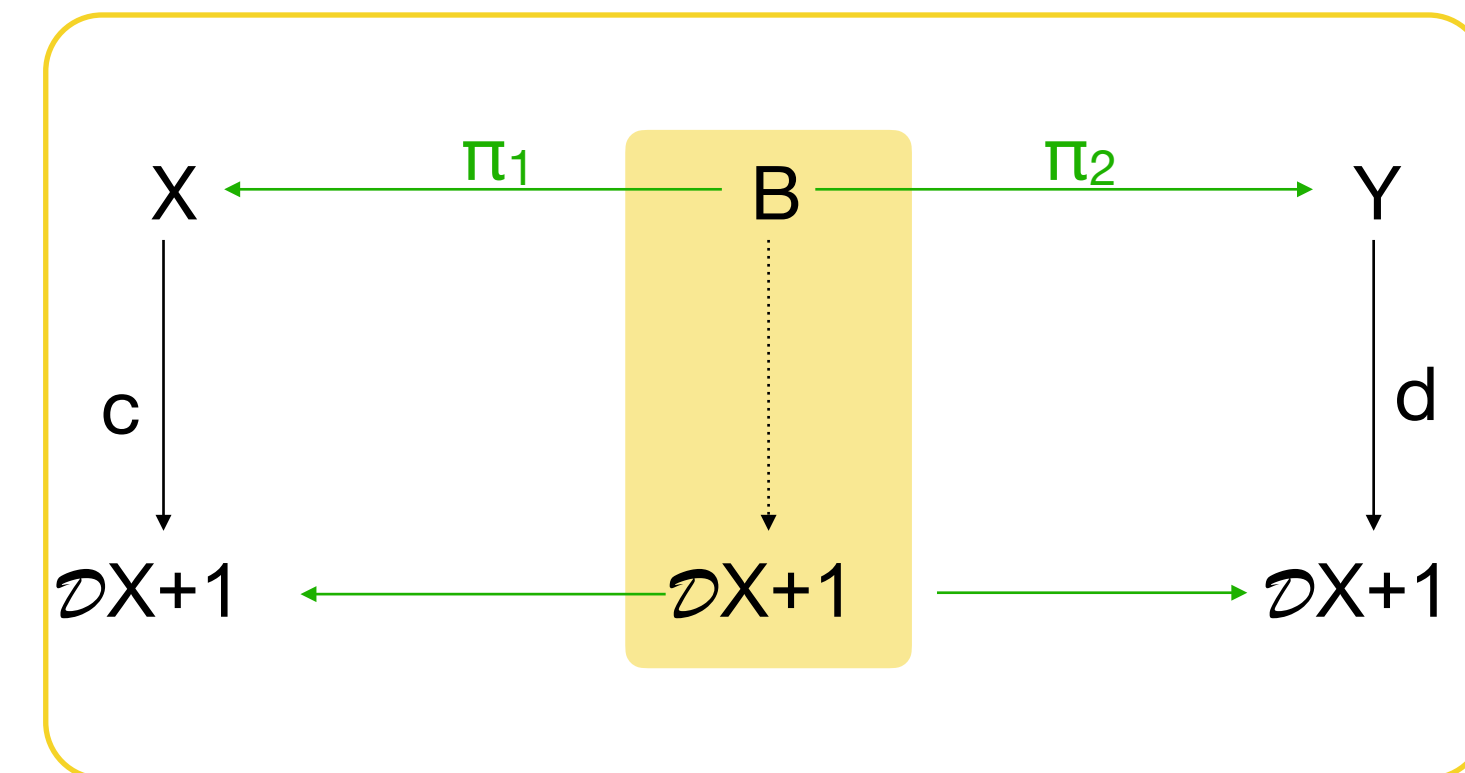
NFA

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



MC

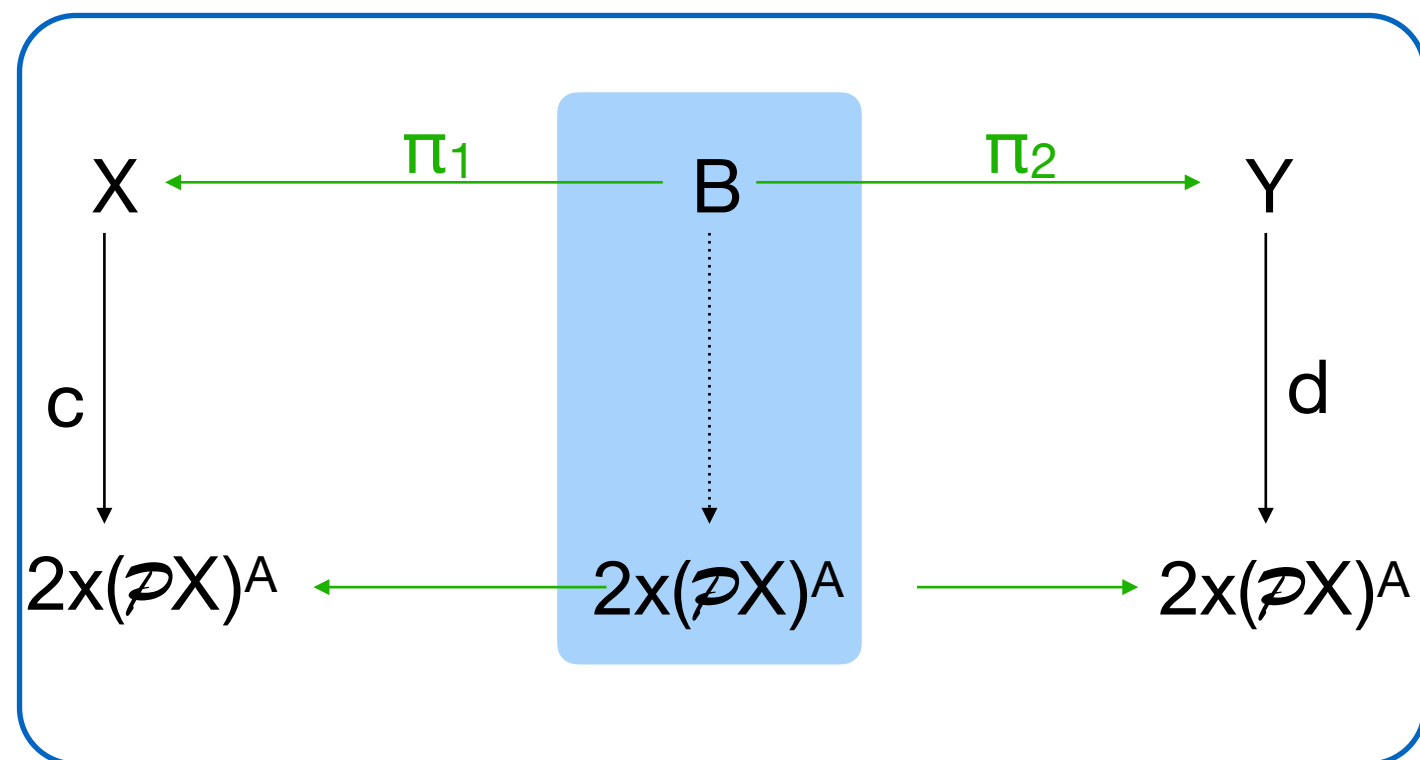
$$X \rightarrow \mathcal{D}X + I$$



Coalgebraic bisimilarity is bisimilarity

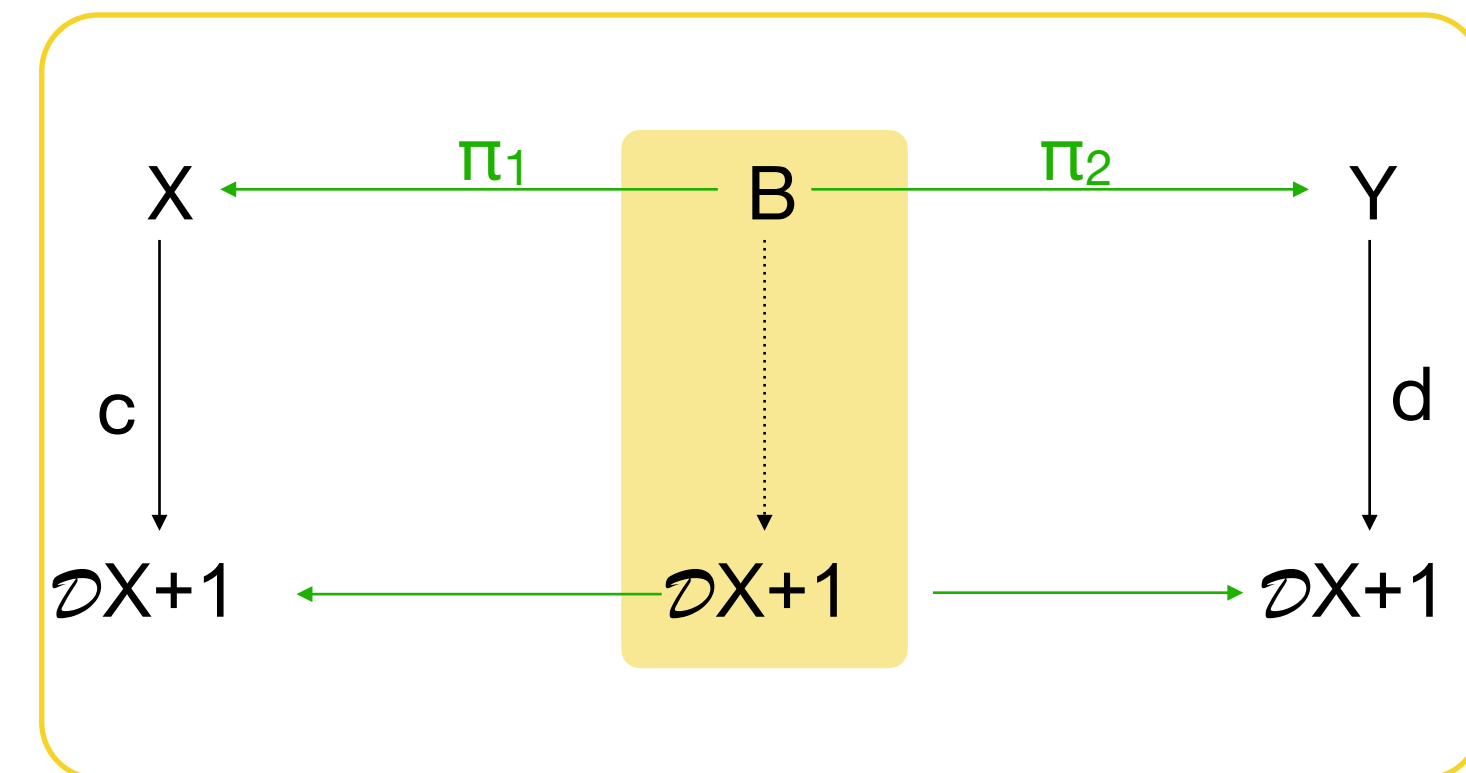
NFA

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



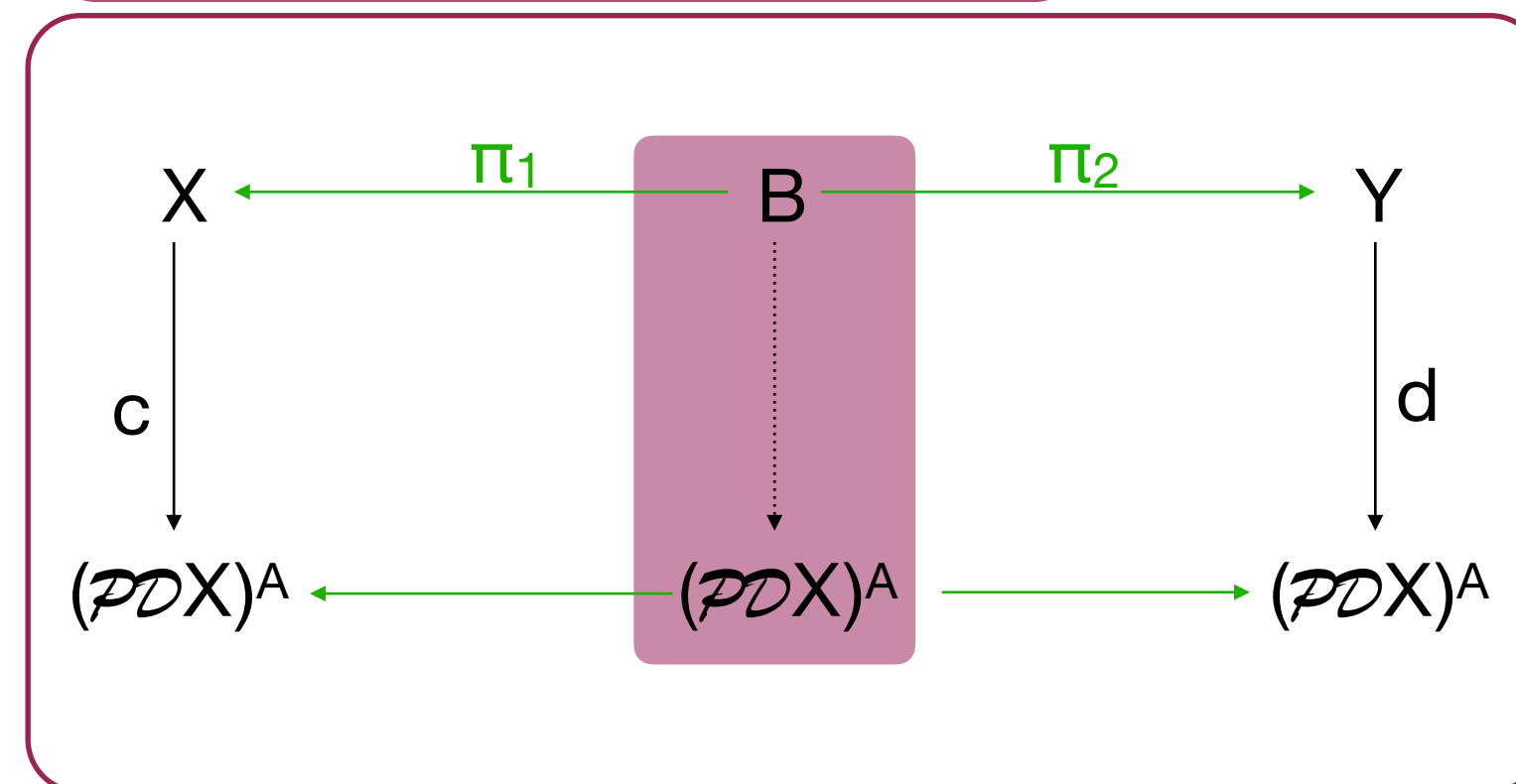
MC

$$X \rightarrow \mathcal{D}X + I$$



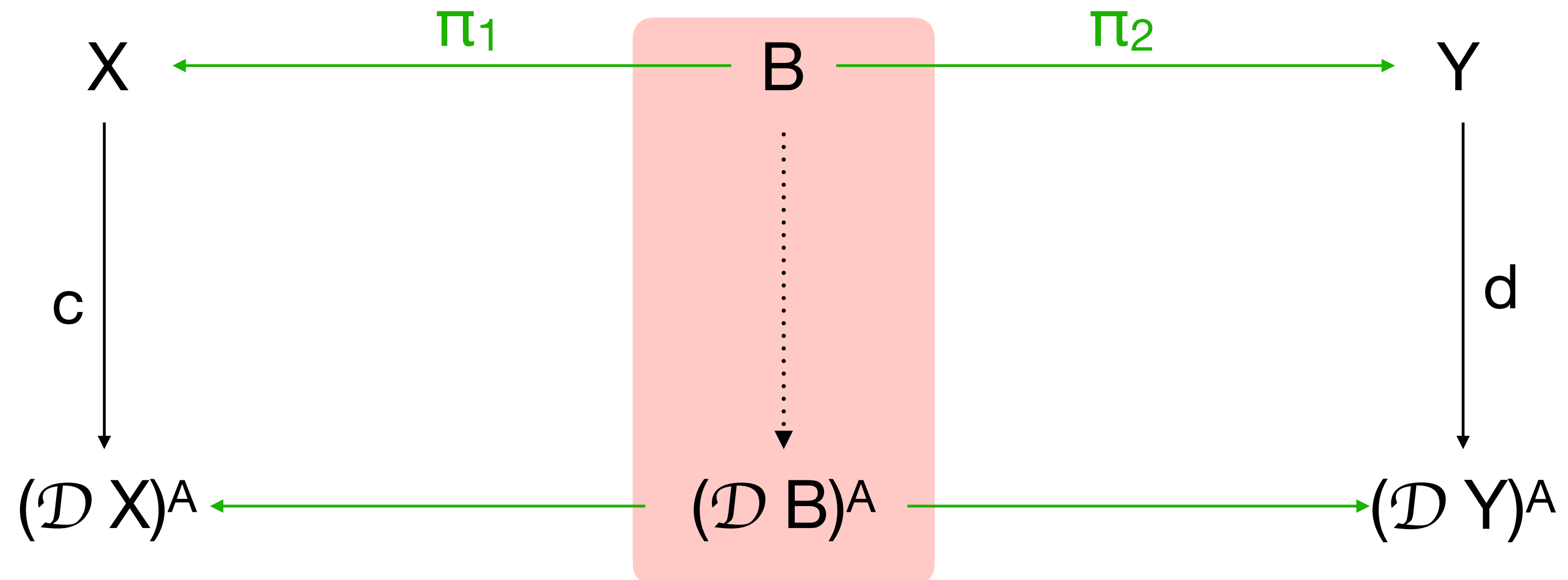
PA

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



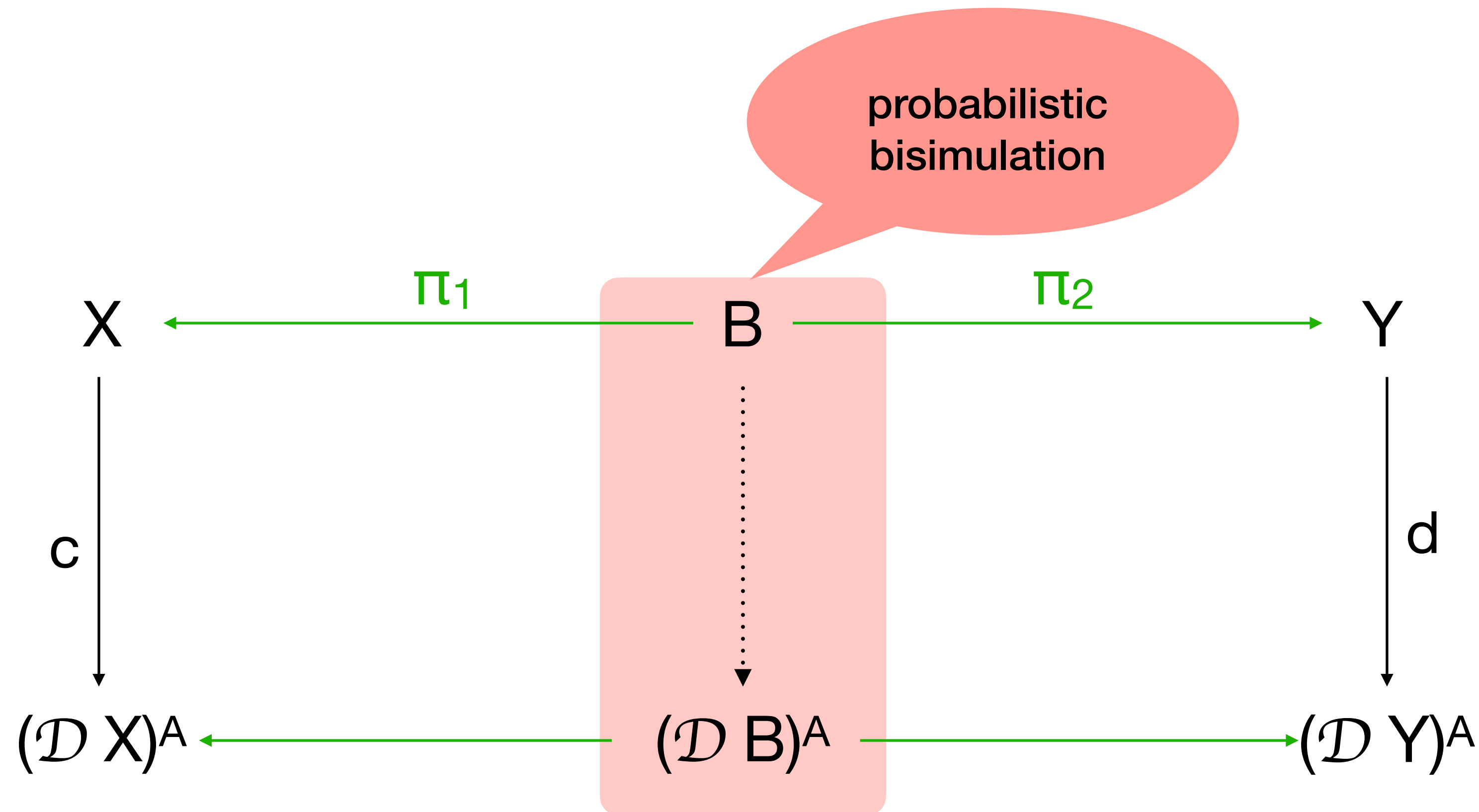
Bisimilarity of Probabilistic Transition Systems

$$S \rightarrow (\mathcal{D} S)^A$$

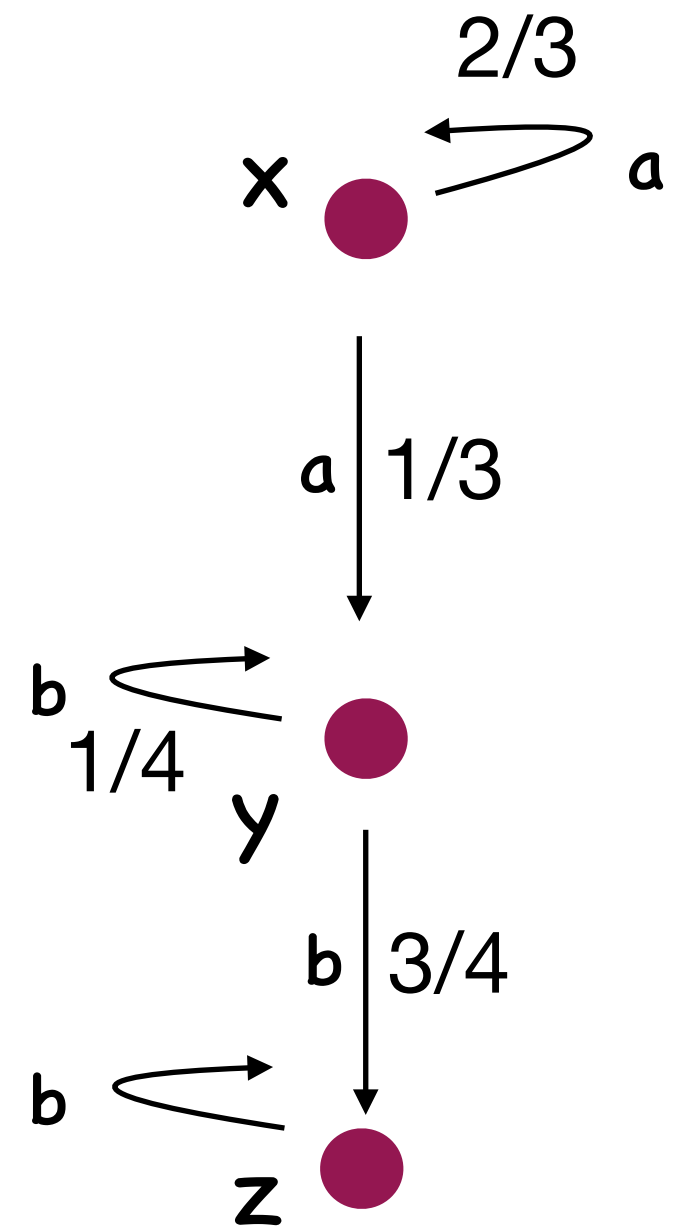
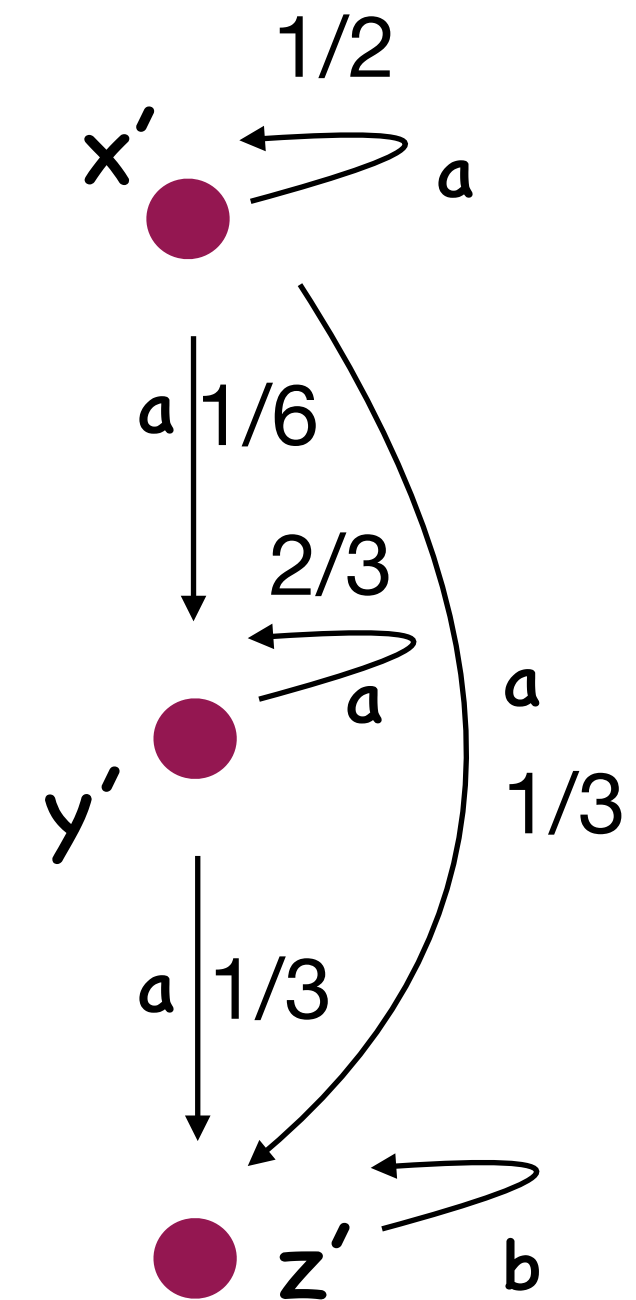


Bisimilarity of Probabilistic Transition Systems

$$S \rightarrow (\mathcal{D} S)^A$$

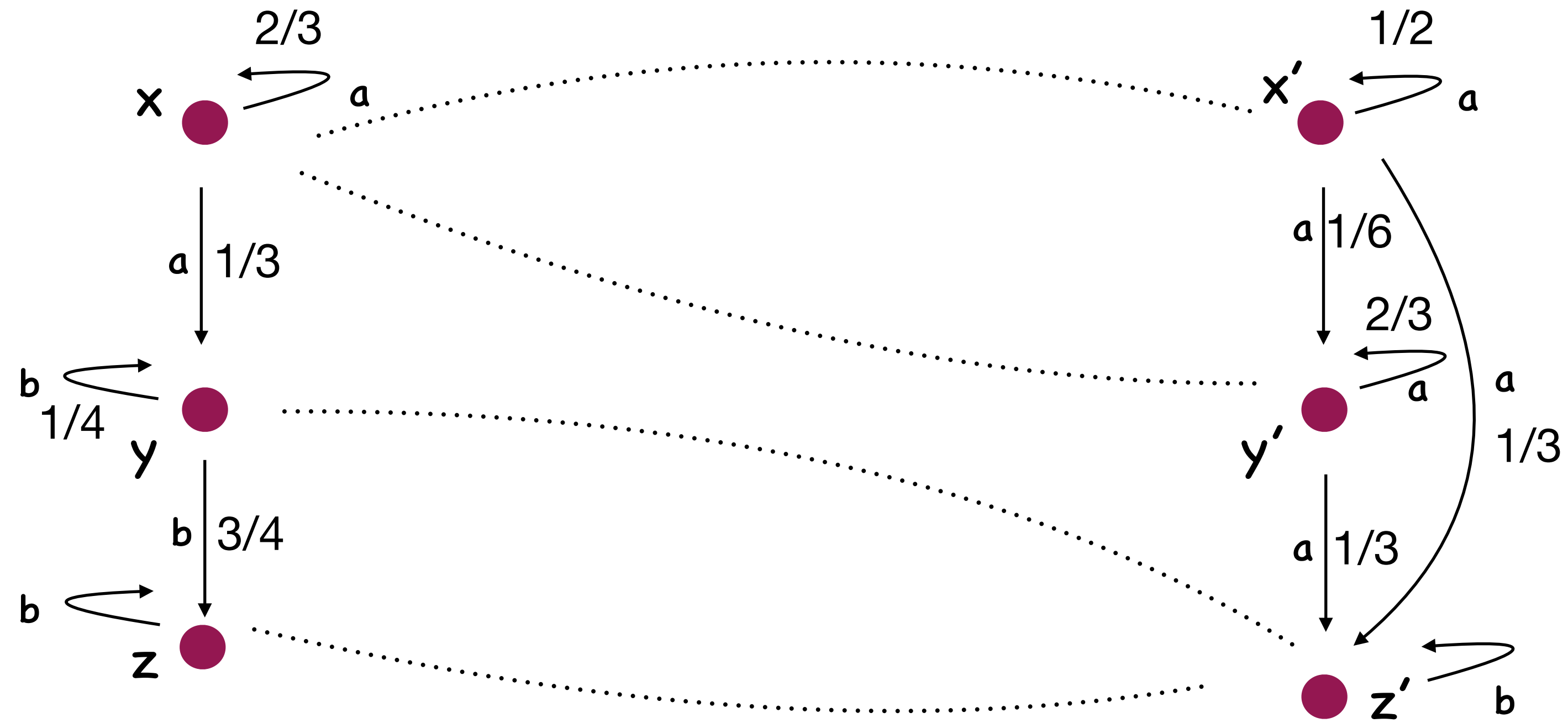


Bisimilarity of Probabilistic Transition Systems



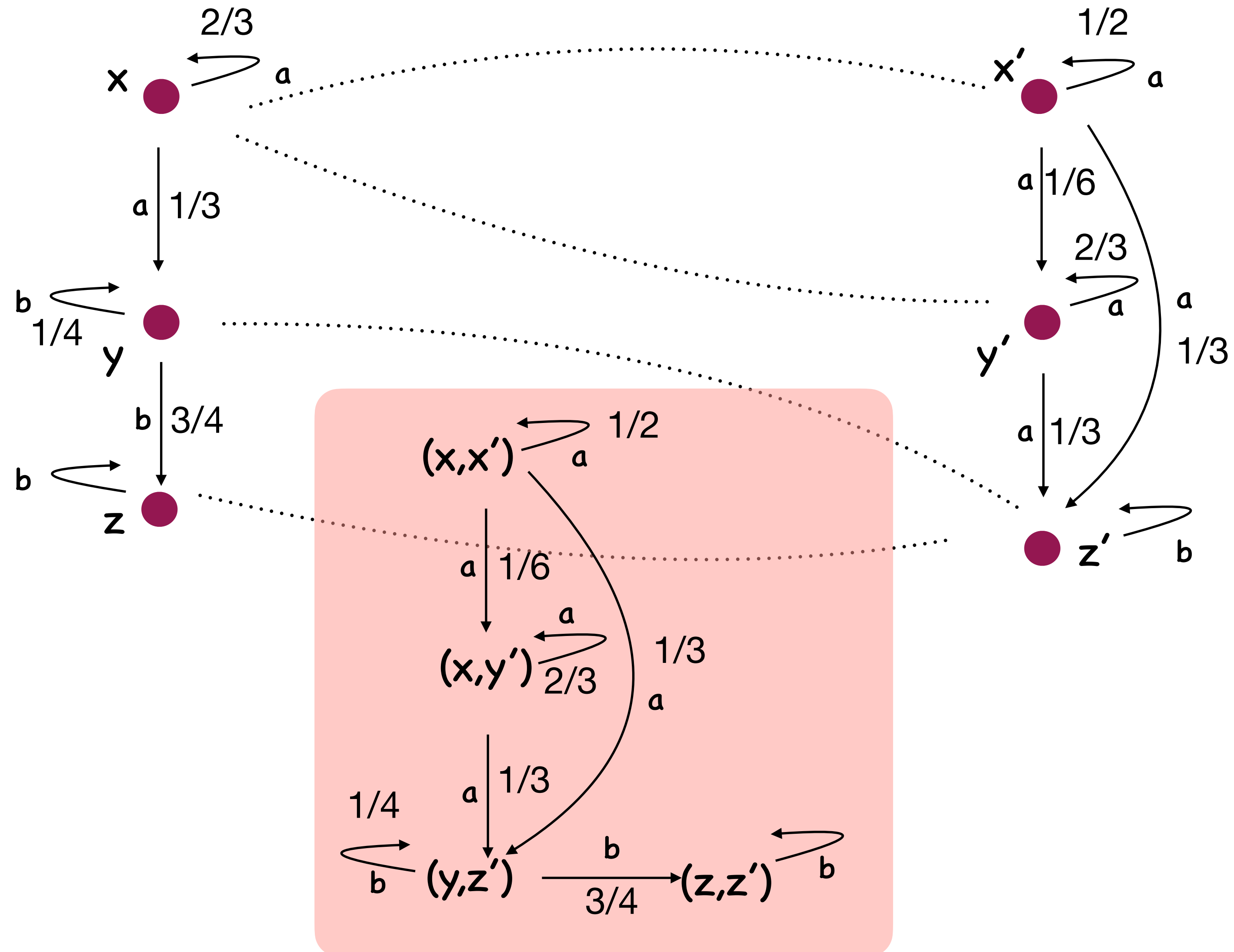
$$S \rightarrow (\mathcal{D} S)^A$$

Bisimilarity of Probabilistic Transition Systems



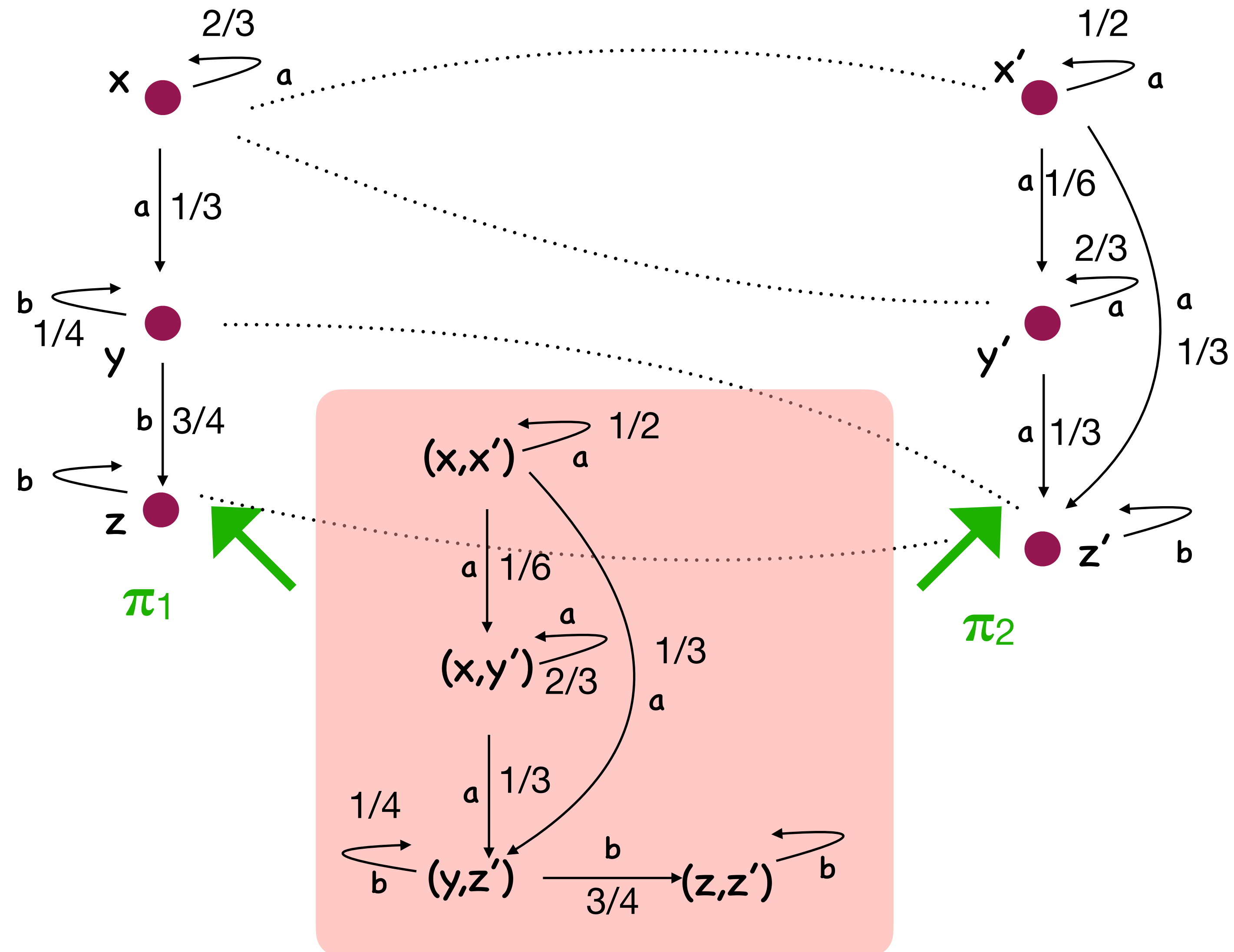
$$S \rightarrow (\mathcal{D} S)^A$$

Bisimilarity of Probabilistic Transition Systems

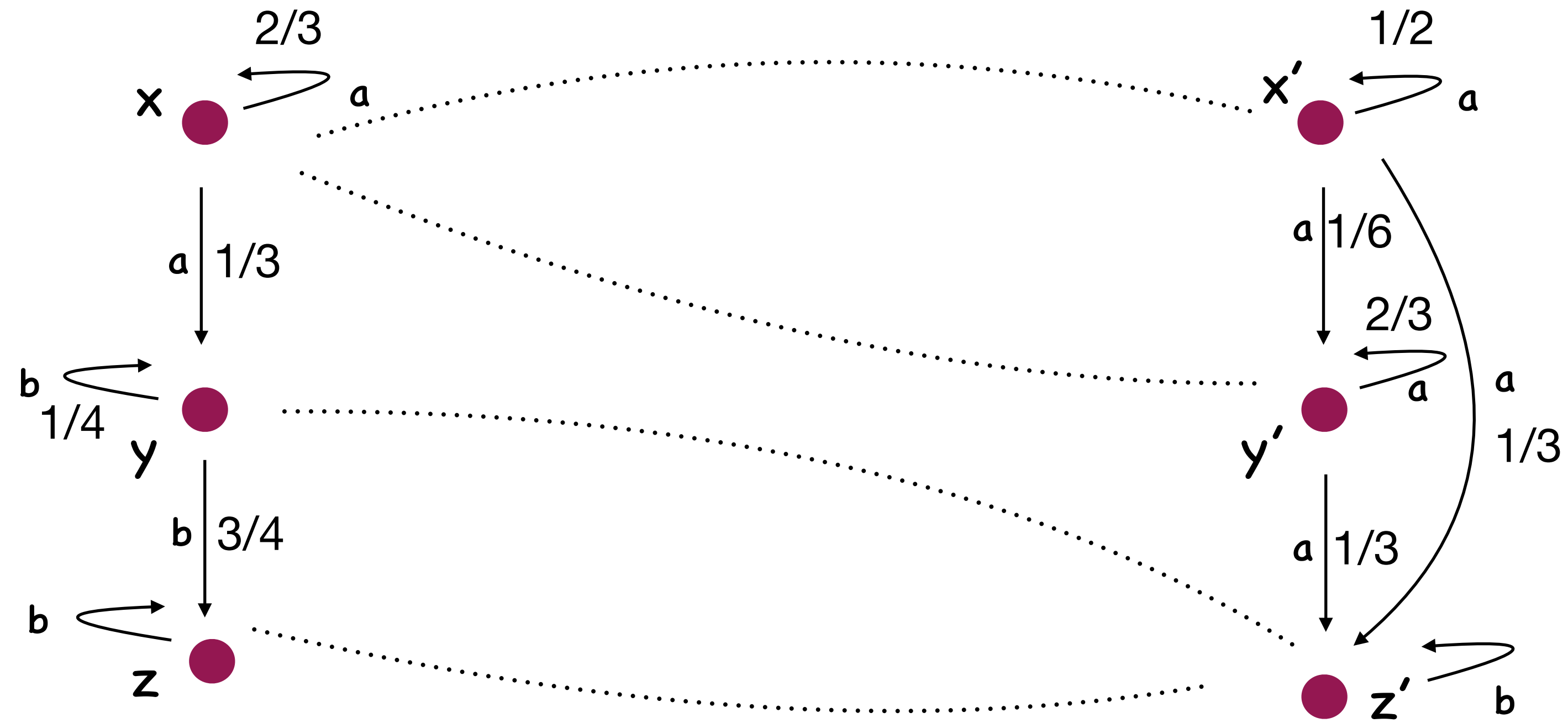


$$S \rightarrow (\mathcal{D} S)^A$$

Bisimilarity of Probabilistic Transition Systems

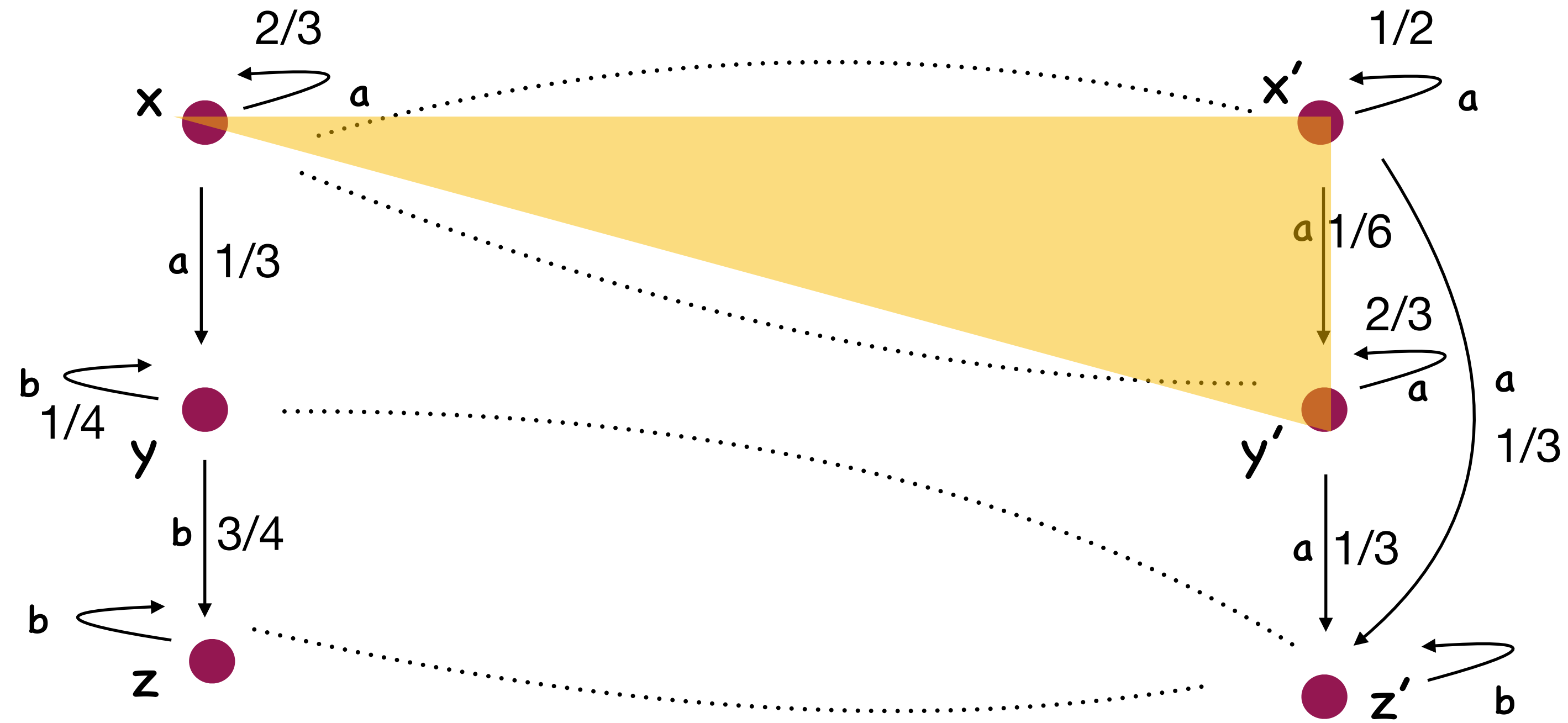


Bisimilarity of Probabilistic Transition Systems



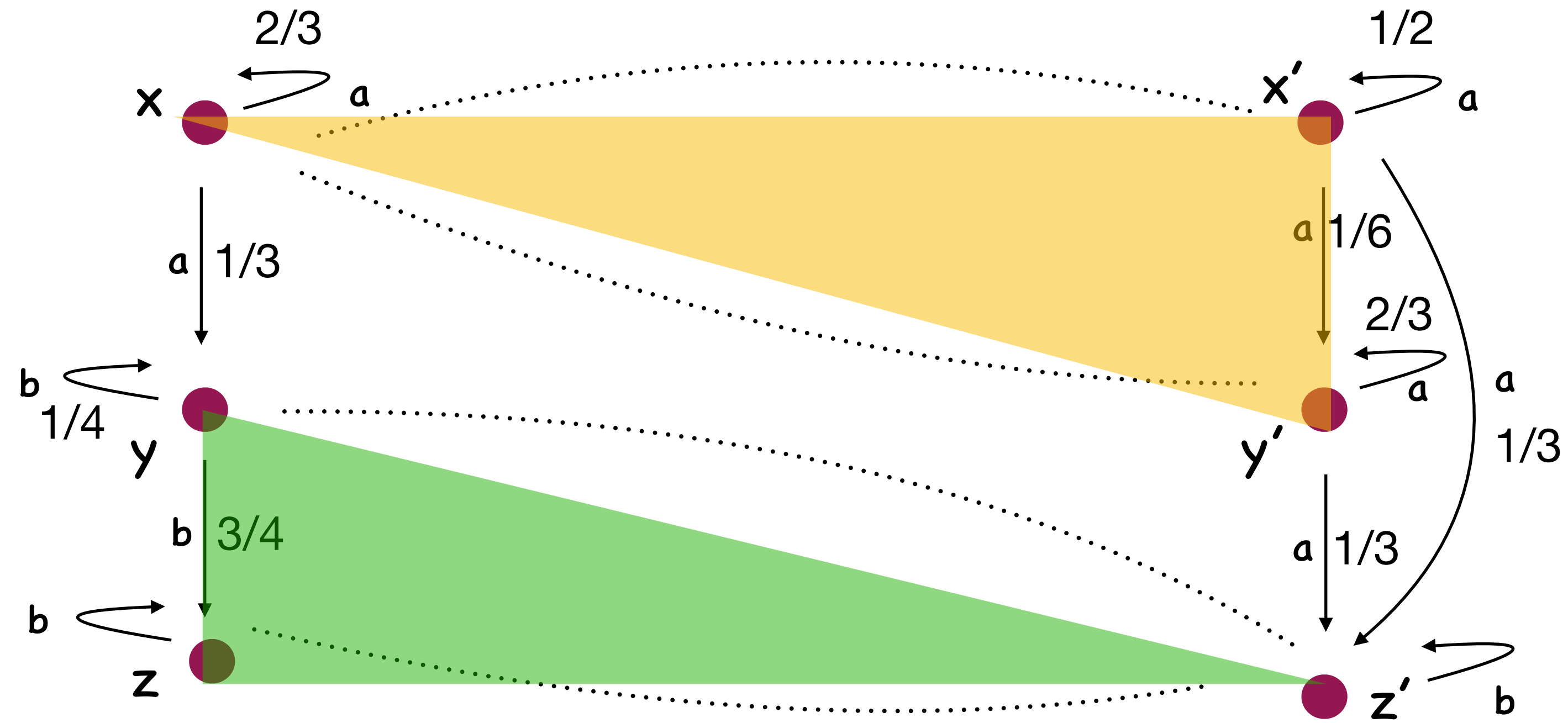
$$S \rightarrow (\mathcal{D} S)^A$$

Bisimilarity of Probabilistic Transition Systems



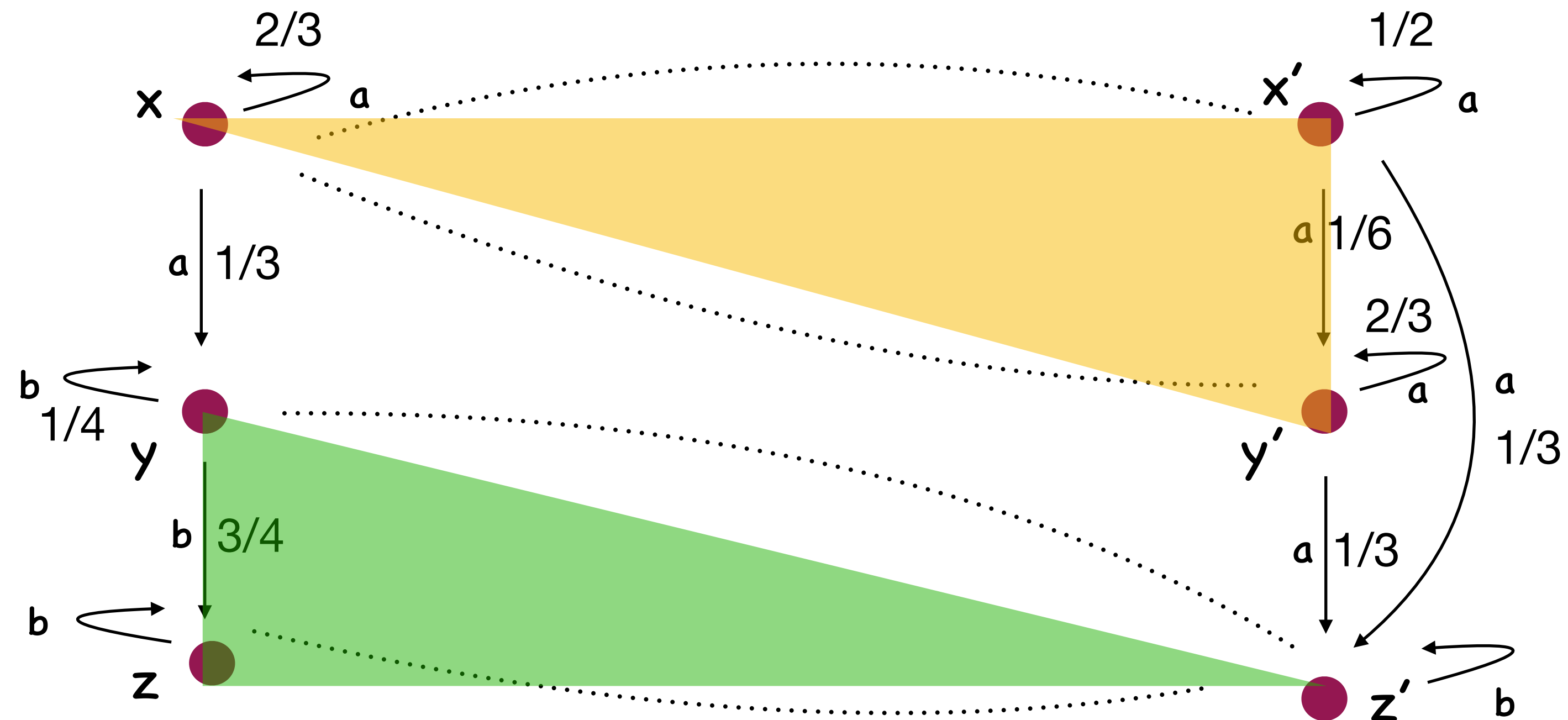
$$S \rightarrow (\mathcal{D} S)^A$$

Bisimilarity of Probabilistic Transition Systems



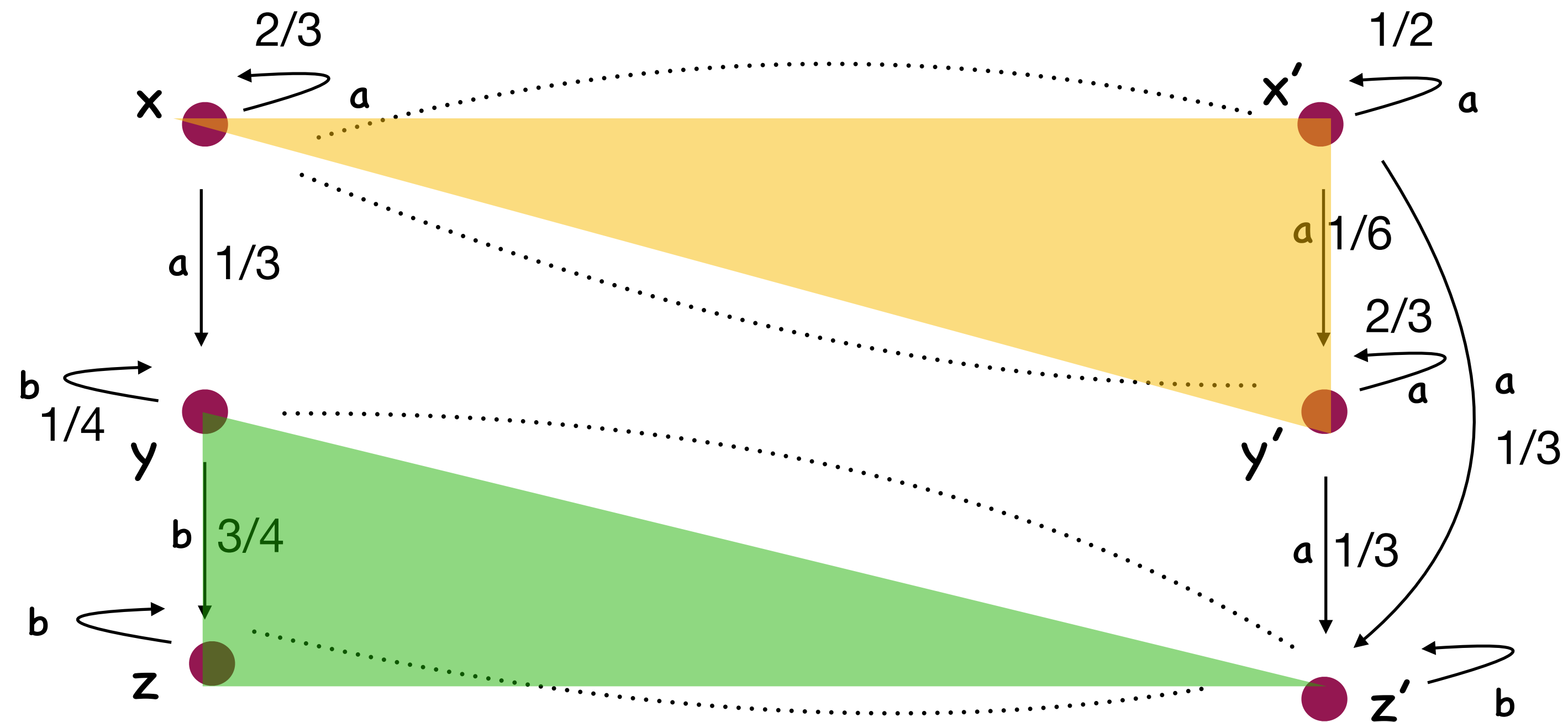
$$S \rightarrow (\mathcal{D} S)^A$$

Bisimilarity of Probabilistic Transition Systems



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

Bisimilarity of Probabilistic Transition Systems

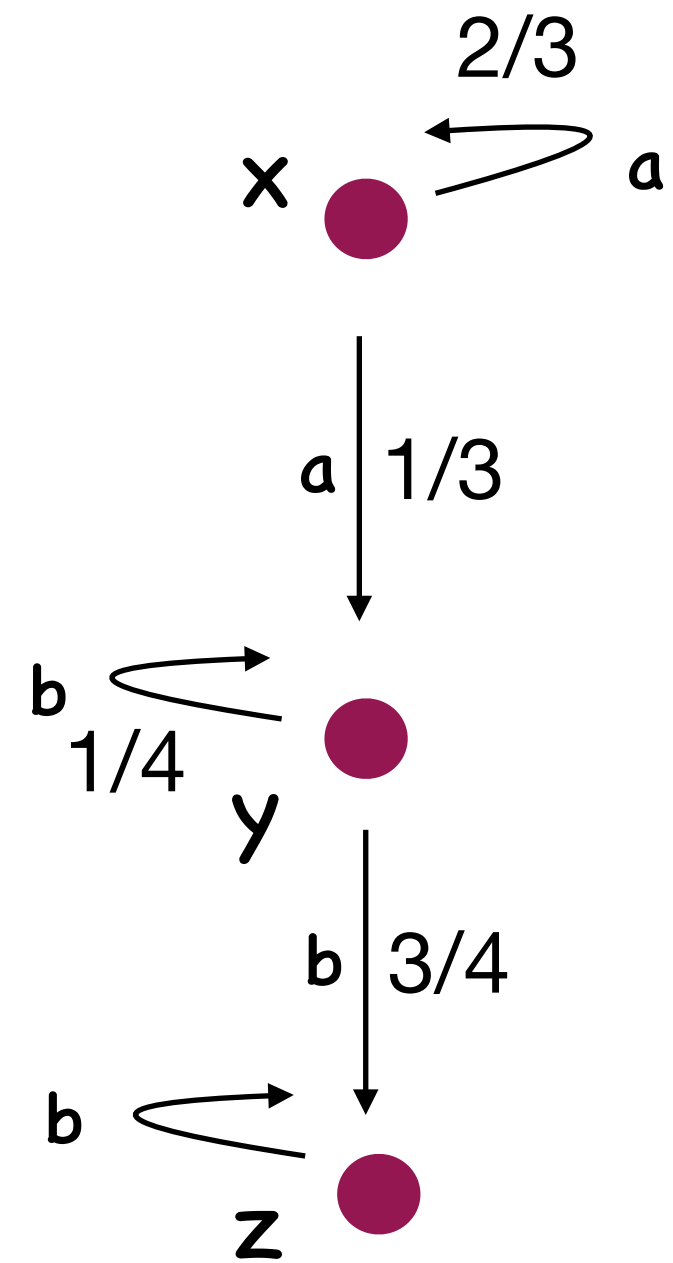
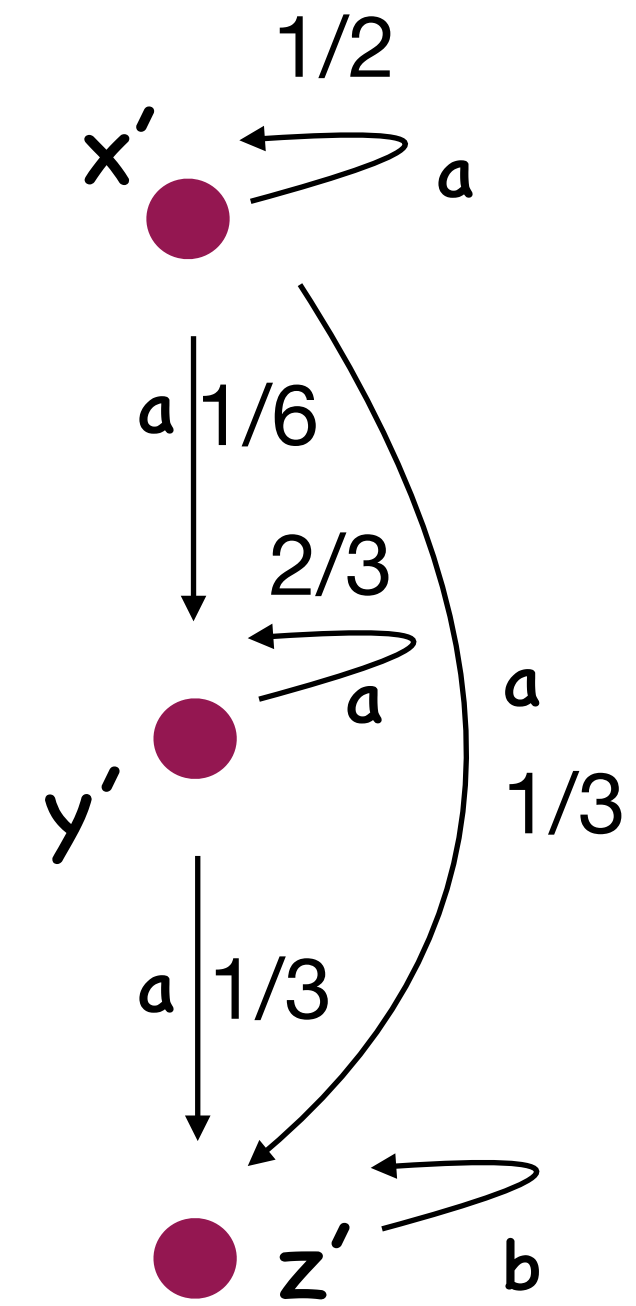


$$S \rightarrow (\mathcal{D} S)^A$$

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

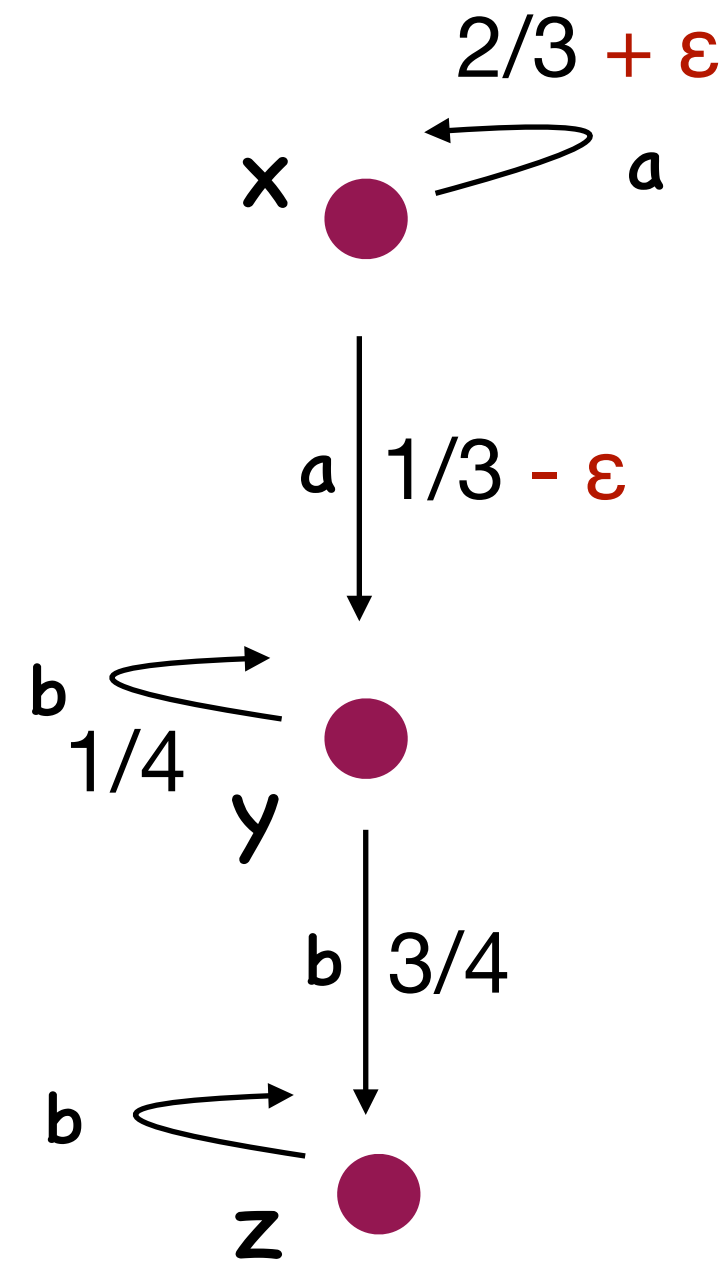
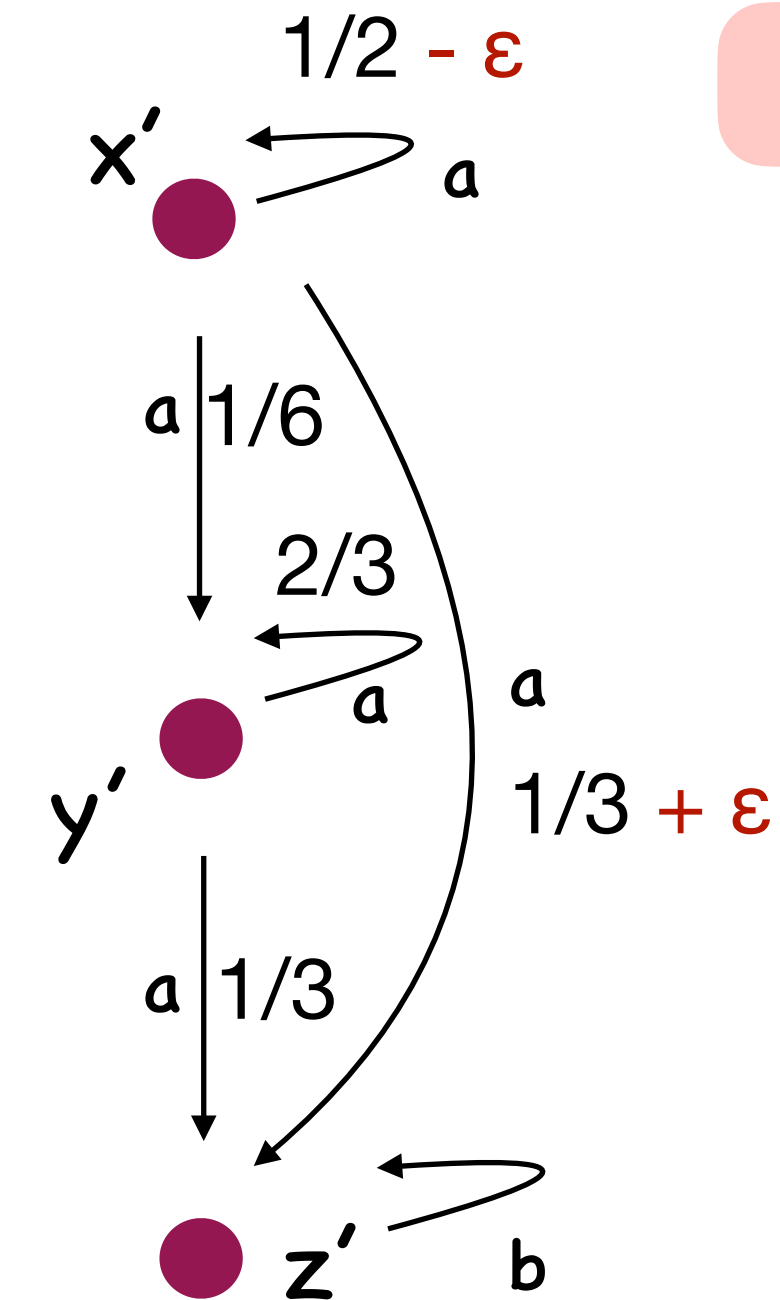
Larsen & Skou 1991

Sometimes bisimilarity is too strong / not robust



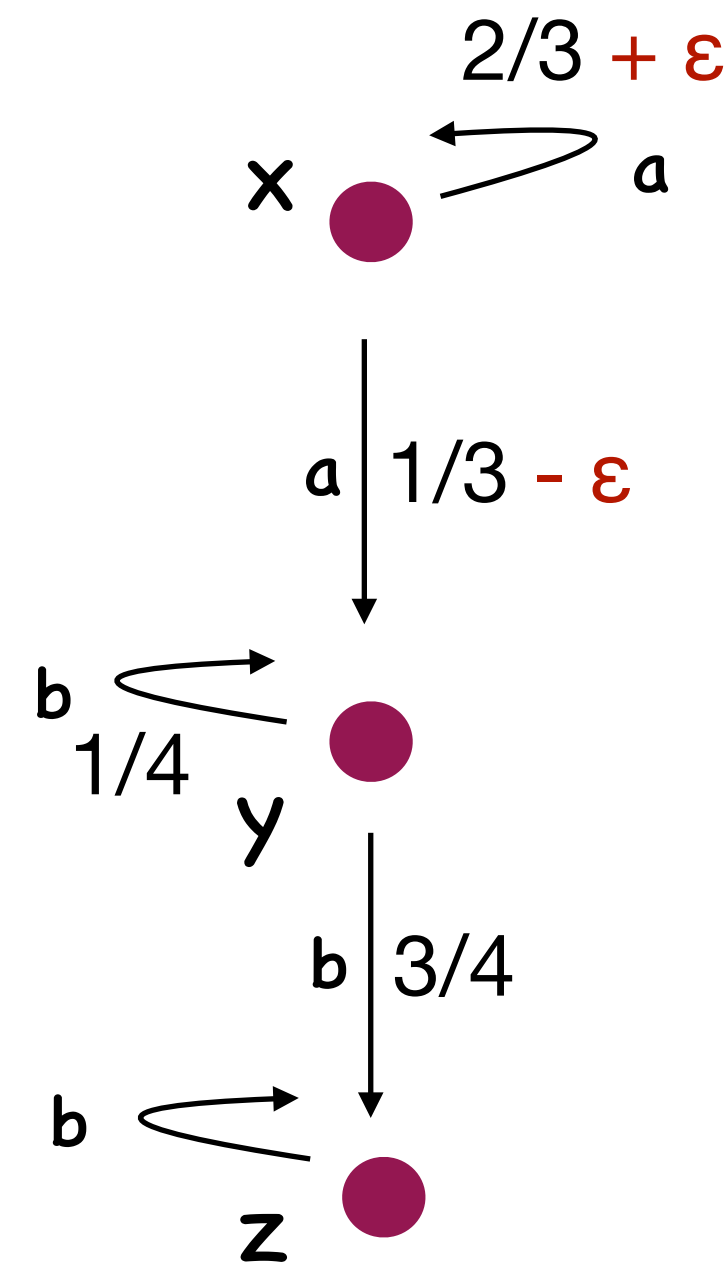
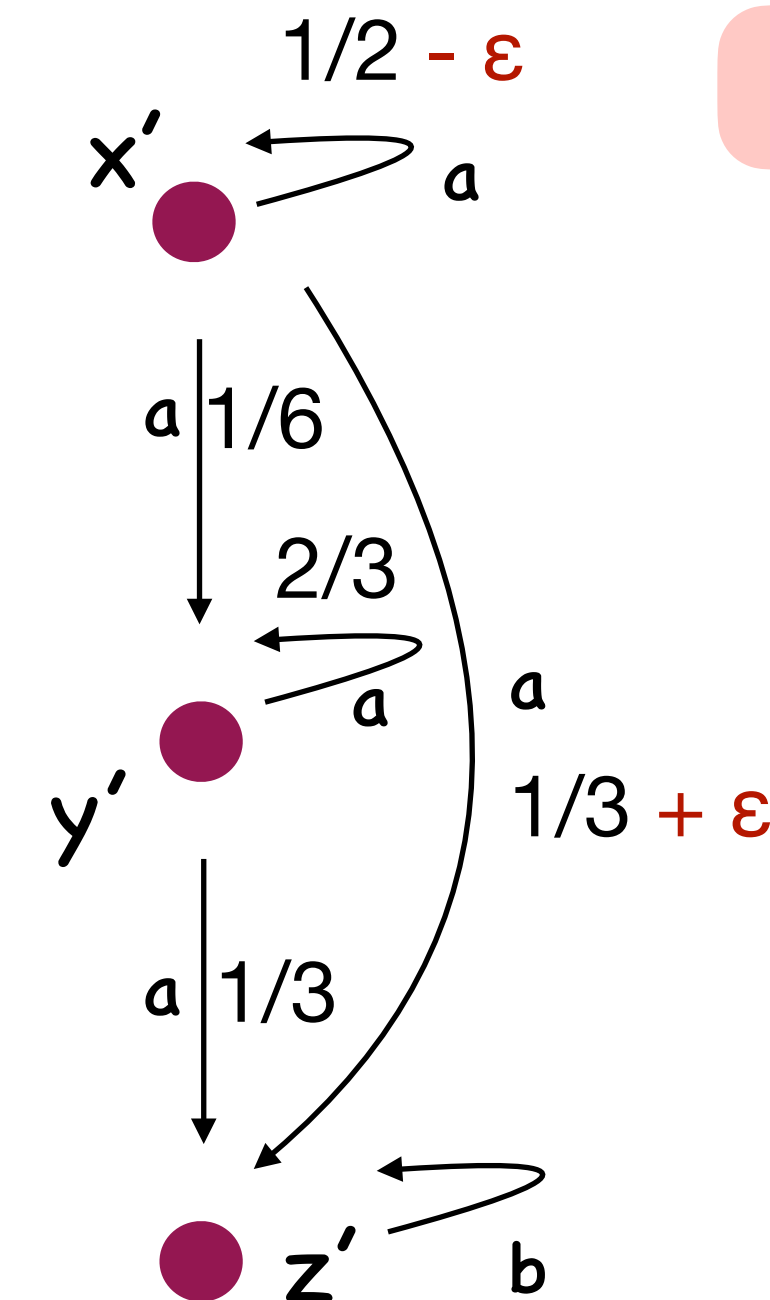
$$S \rightarrow (\mathcal{D} S)^A$$

Sometimes bisimilarity is too strong / not robust



$$S \rightarrow (\mathcal{D} S)^A$$

Sometimes bisimilarity is too strong / not robust

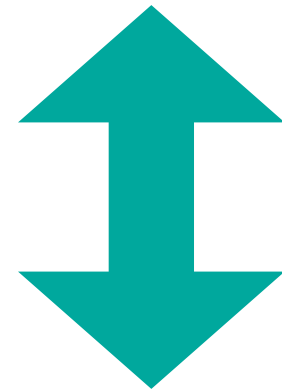


$$S \rightarrow (\mathcal{D} S)^A$$

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



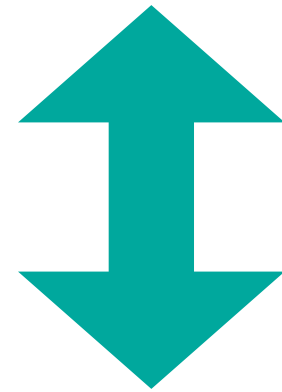
2. Approximate, ε -bisimulations and ε -bisimilarity

Two possible solutions:

Desharnais & Panangaden
van Breugel & Worrell

...

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



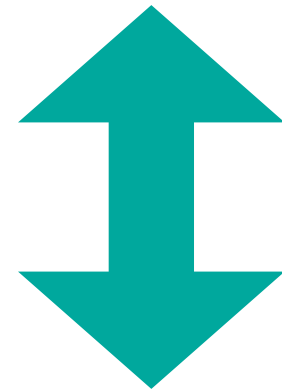
2. Approximate, ε -bisimulations and ε -bisimilarity

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

...

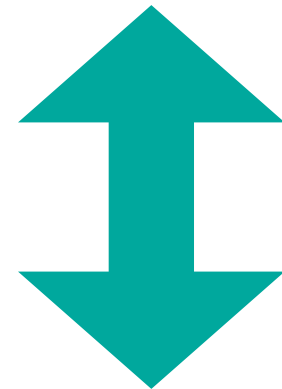
Two possible solutions:

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

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

...

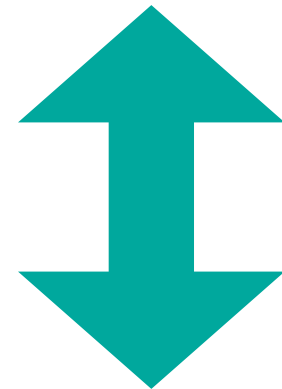
Two possible solutions:

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

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 $\Delta(d)(x,y) = \sup_a \delta_K(\tau_a(x), \tau_a(y))$
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.

van Breugel & Worrell

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.

van Breugel & Worrell

Baldan, Bonchi, König

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.

van Breugel & Worrell

Baldan, Bonchi, König

Mio, Sarkis, Vignudelli

The well known d_K

Kantorovich
distance on
distributions

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

van Breugel & Worrell

Baldan, Bonchi, König

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 $x R y \Rightarrow \tau(x)(X) \leq \tau(y)(R(X)) + \epsilon$ for any subset of states X .
2. It is an ϵ -bisimulation if it is a symmetric ϵ -simulation.
3. $s \sim_{\epsilon} t$ 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 $x R y \Rightarrow \tau(x)(X) \leq \tau(y)(R(X)) + \epsilon$ for any subset of states X .
2. It is an ϵ -bisimulation if it is a symmetric ϵ -simulation.
3. $s \sim_\epsilon t$ 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 $x R y \Rightarrow \tau(x)(X) \leq \tau(y)(R(X)) + \epsilon$ for any subset of states X .
2. It is an ϵ -bisimulation if it is a symmetric ϵ -simulation.
3. $s \sim_\epsilon t$ if they are related by some ϵ -bisimulation.

ϵ -bisimilarity, is a uniformity

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

ϵ -Bisimulation, ϵ -distance (one label)

...
Desharnais & Laviolette & Tracol
...

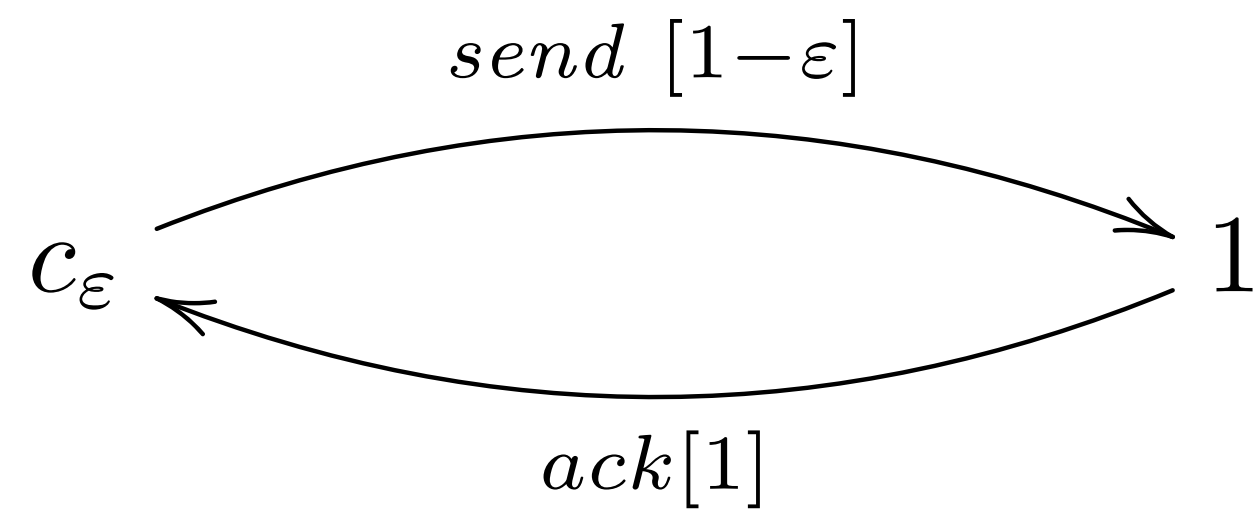
1. A relation R is an ϵ -simulation iff $x R y \Rightarrow \tau(x)(X) \leq \tau(y)(R(X)) + \epsilon$ for any subset of states X .
2. It is an ϵ -bisimulation if it is a symmetric ϵ -simulation.
3. $s \sim_\epsilon t$ if they are related by some ϵ -bisimulation.

ϵ -bisimilarity, is a uniformity

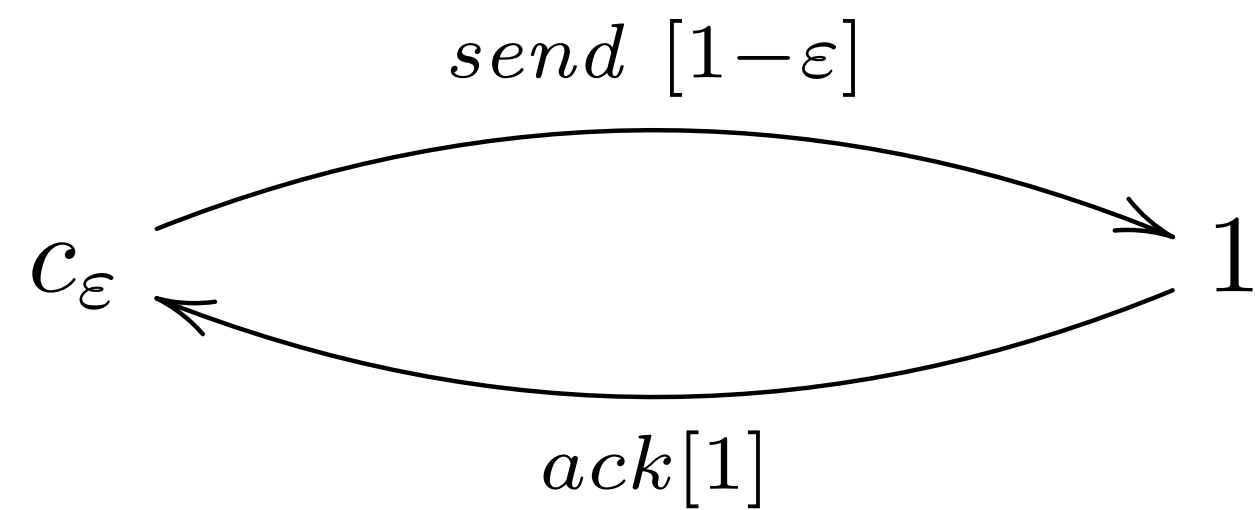
The ϵ -distance is defined by $d_\epsilon(s, t) = \inf \{ \epsilon \mid s \sim_\epsilon t \}$

Has no fixpoint characterisation, or other general properties ?

ε -simulation, ε -bisimulation example

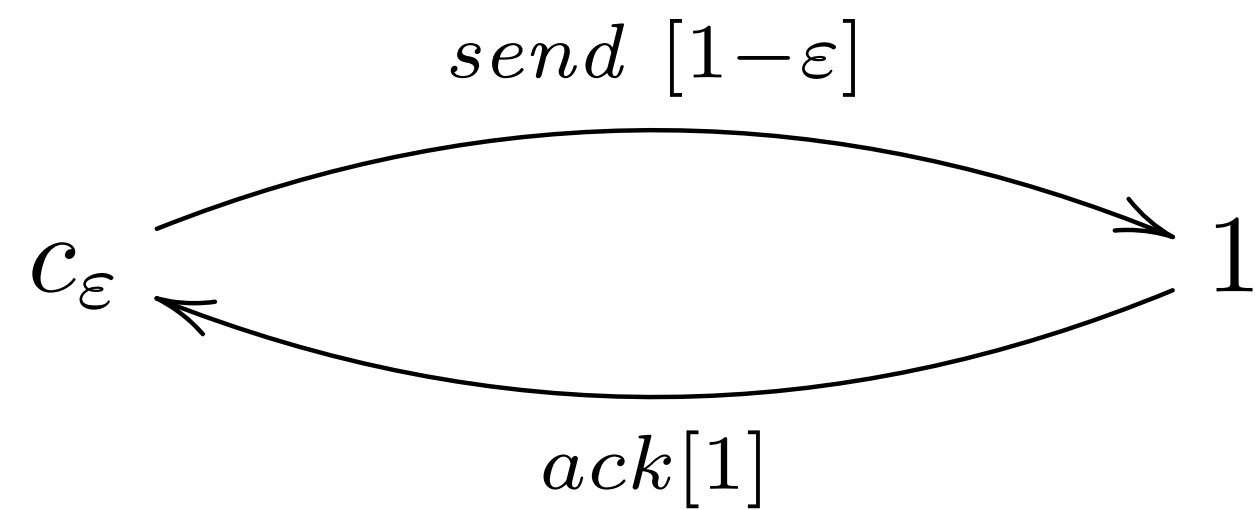


ε -simulation, ε -bisimulation example



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

ε -simulation, ε -bisimulation example

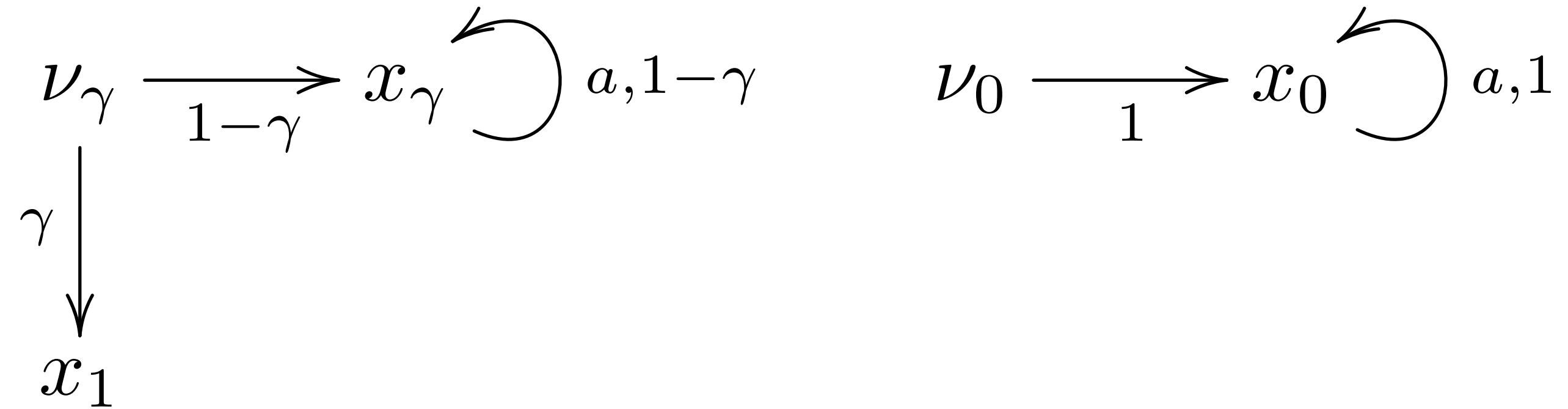


The ε -distance between correct and approximate channel is indeed ε

The K-distance between correct and approximate channel is 1

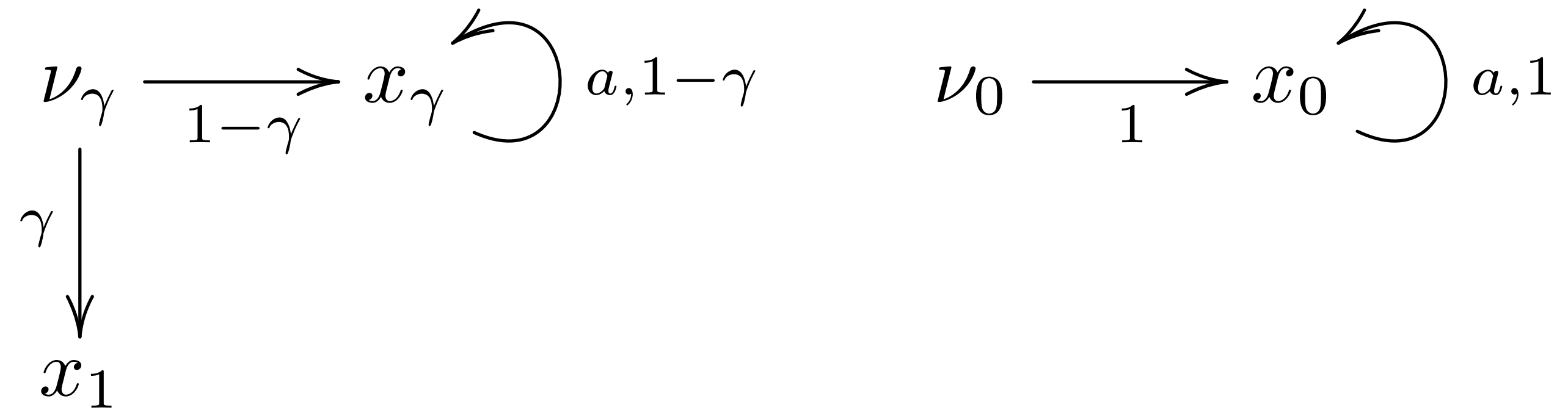
ε -simulation, ε -bisimulation example

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



ε -simulation, ε -bisimulation example

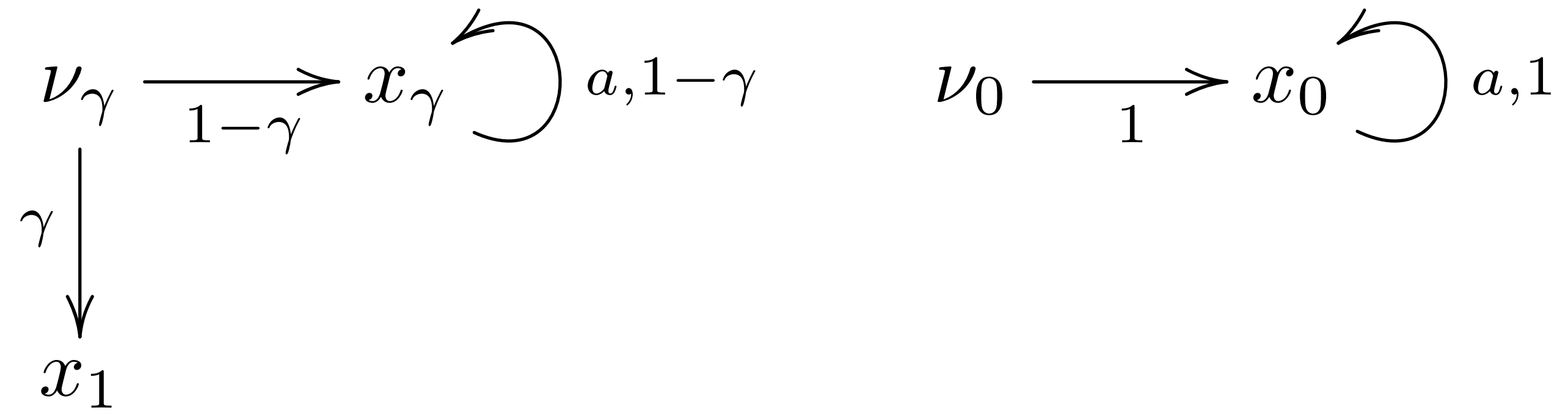
$$\gamma \in [0, 1/2)$$



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

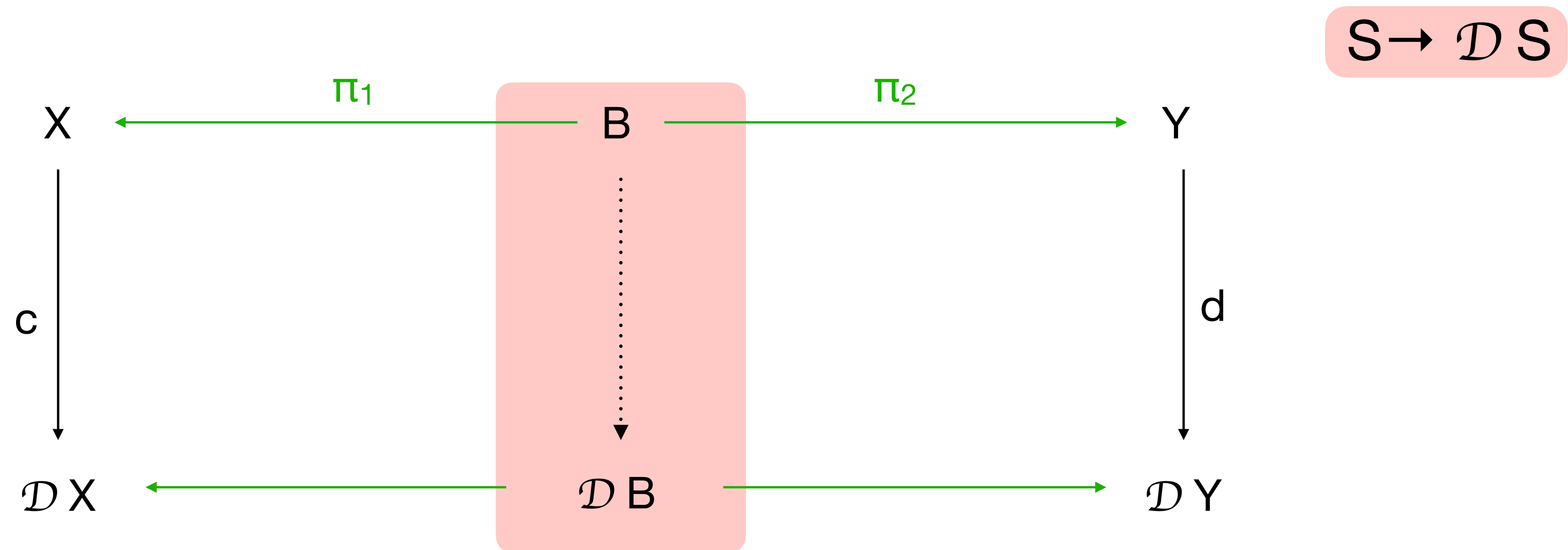
$$\delta_{LP}^d(\mu_0, \mu_1) = \inf\{\varepsilon \mid \forall X \subseteq S : \mu_i(X) \leq \mu_{1-i}(X_\varepsilon^d) + \varepsilon, \text{ for } i = 0, 1\}$$

where

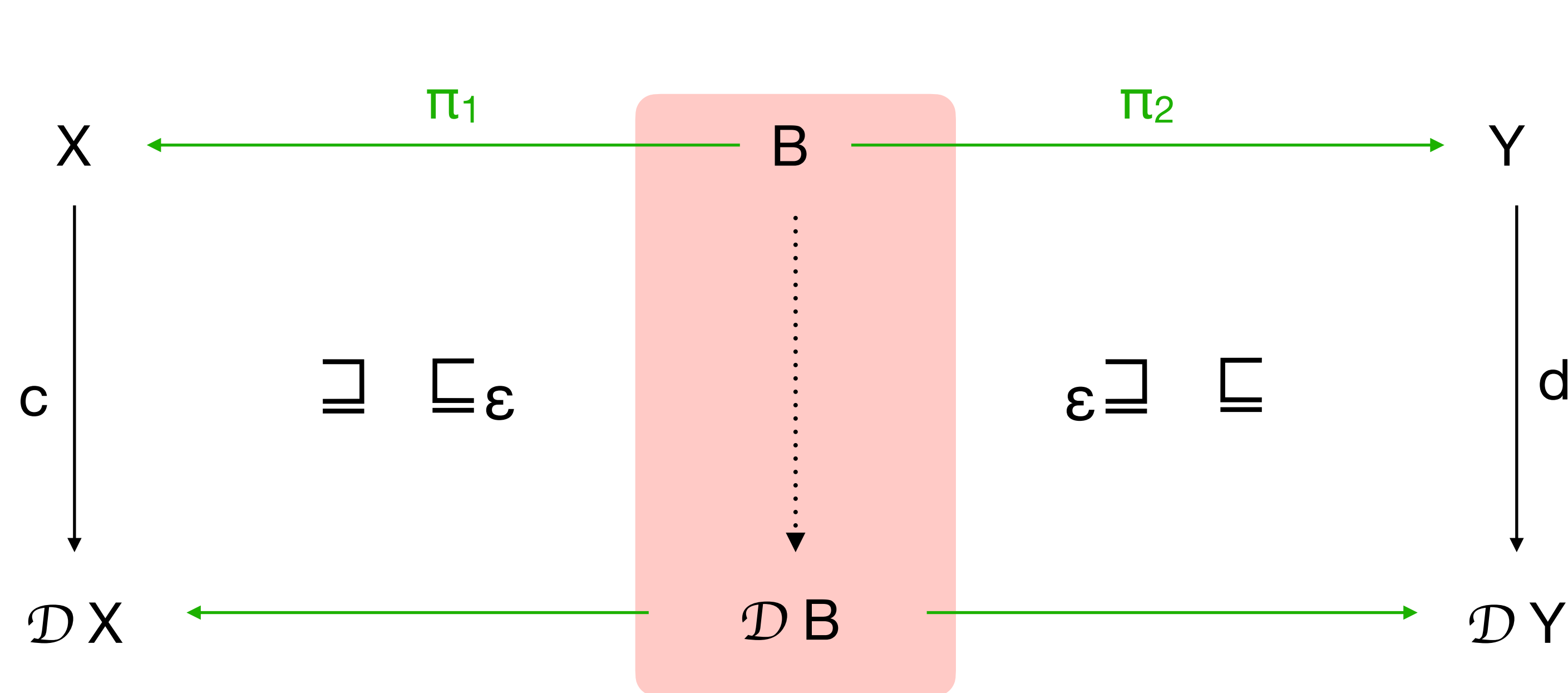
$$X_\varepsilon^d = \{y \mid \exists x \in X : d(x, y) < \varepsilon\}$$

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

Recall bisimulation for PTS (one label)



ε -Bisimulation in a diagram

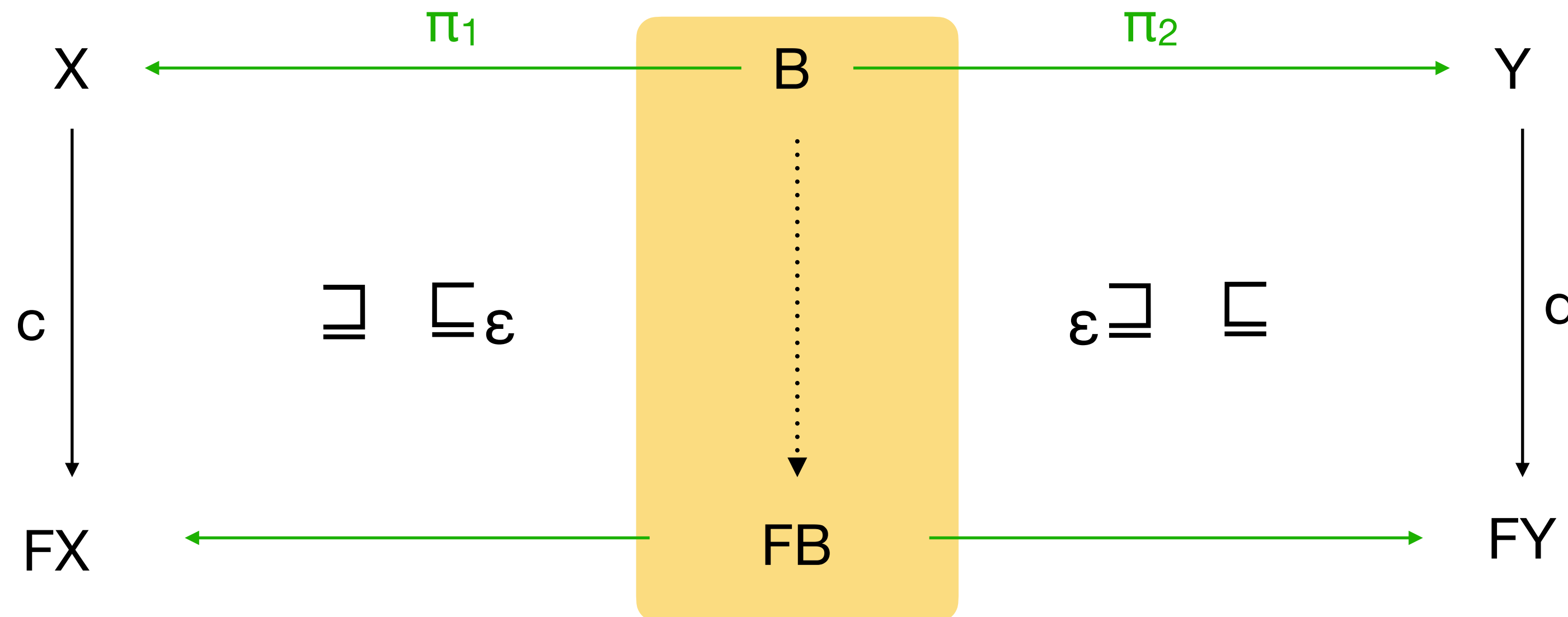


$$S \rightarrow \mathcal{D} S$$

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

ε -Bisimulation for F-coalgebras

$S \rightarrow FS$



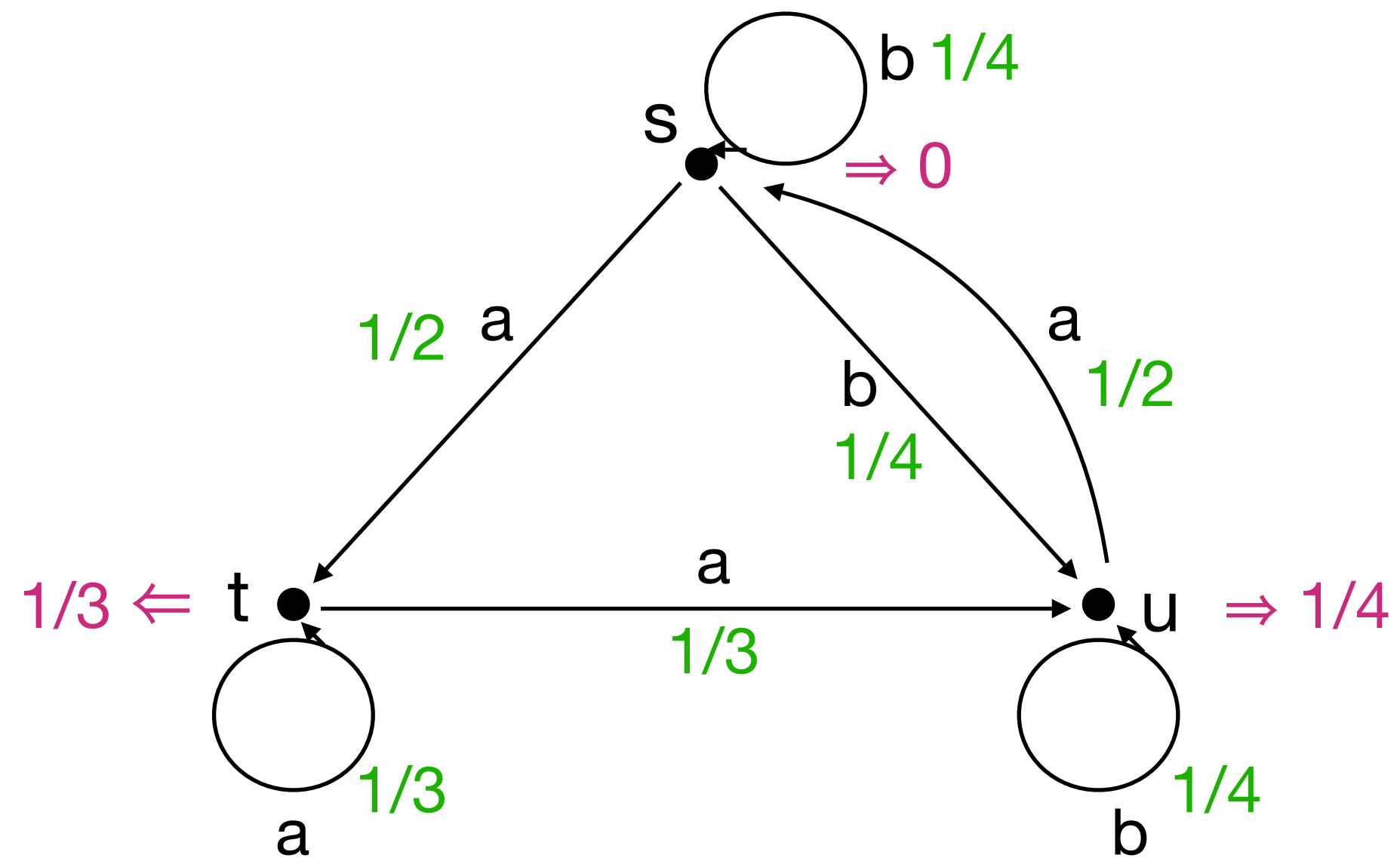
as long as F comes with a suitable order

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

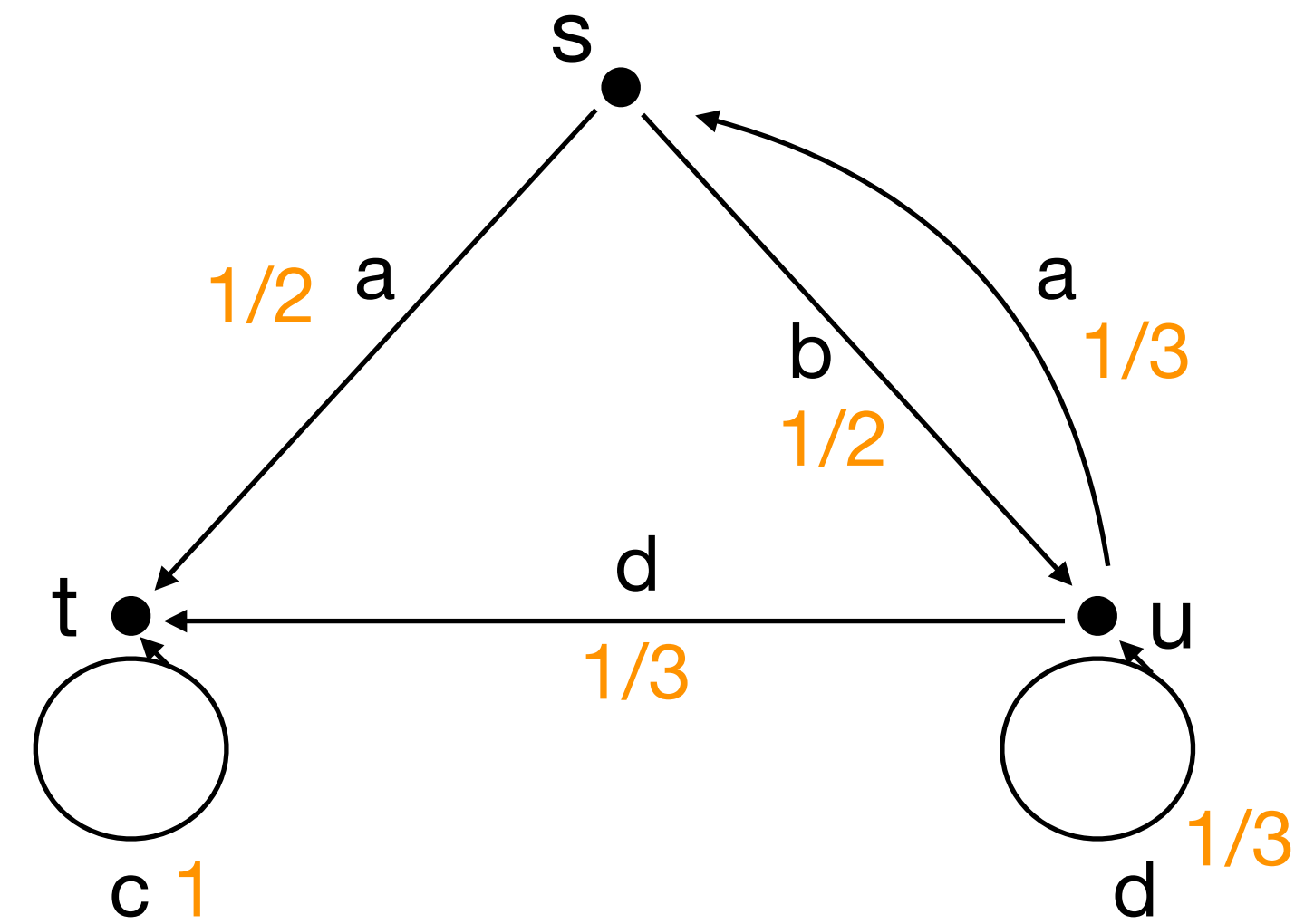
Trace Semantics = Bisimilarity Elsewhere



Finite Traces

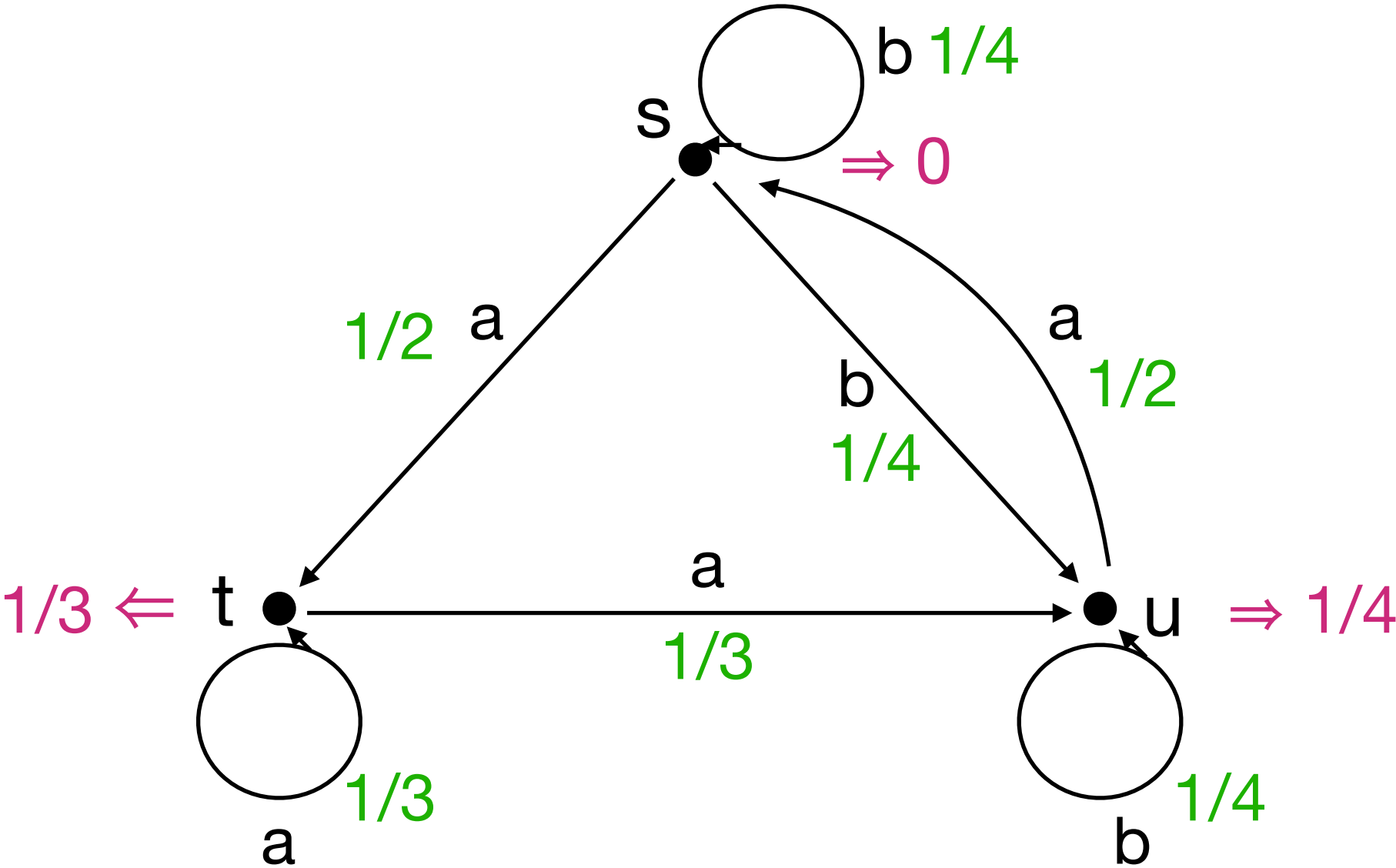


Infinite Traces

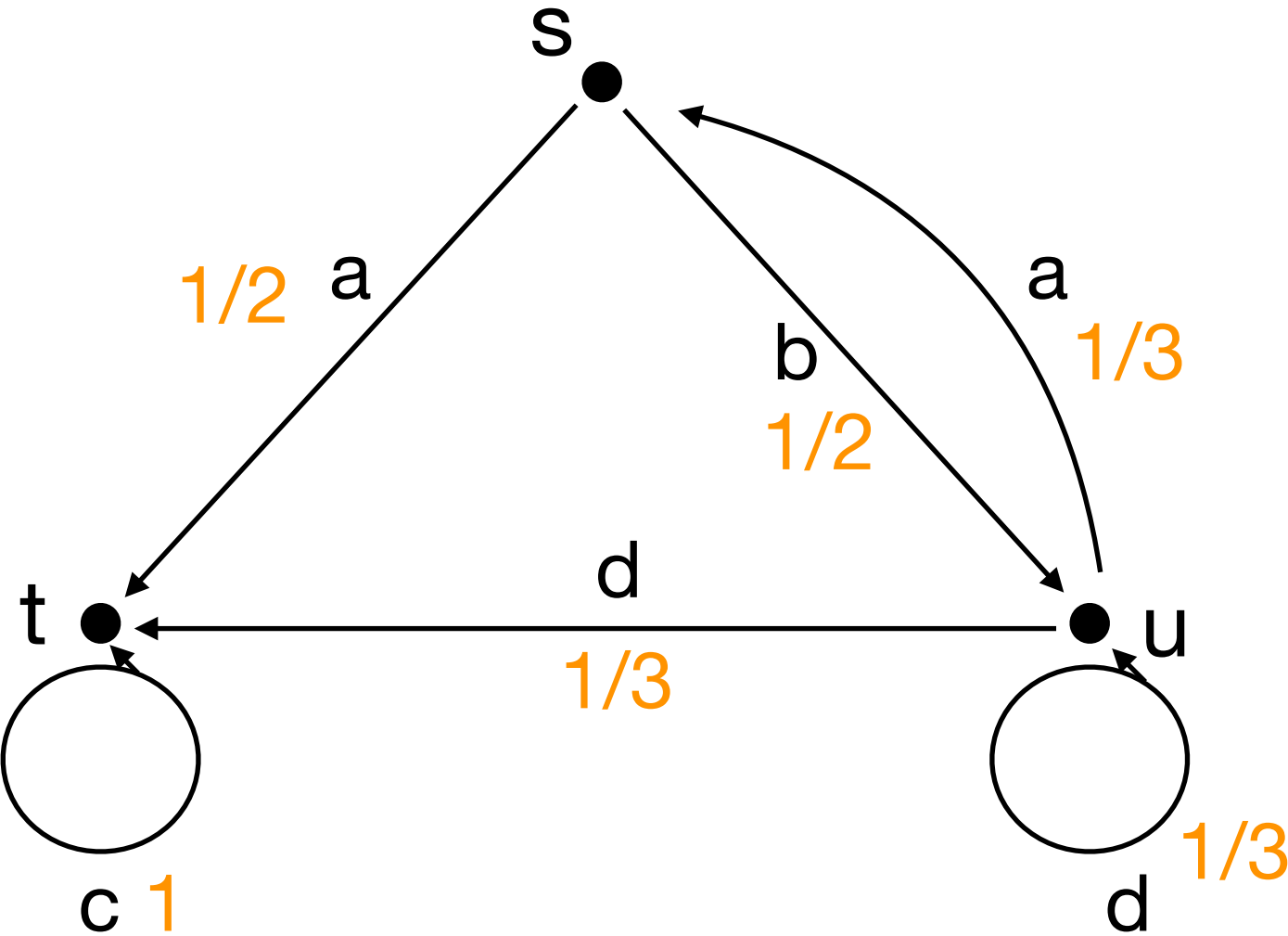


Finite Traces

$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$

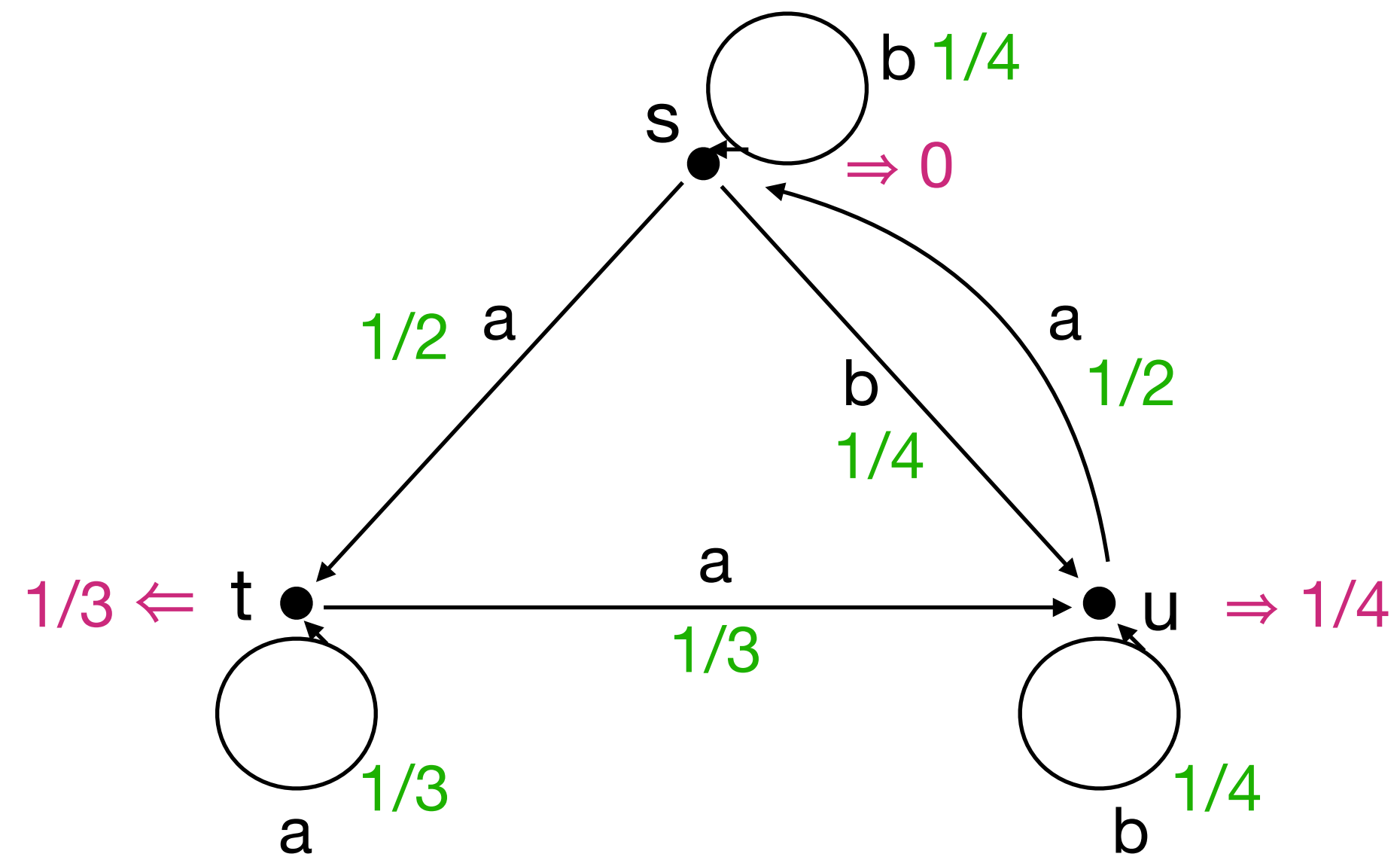


Infinite Traces



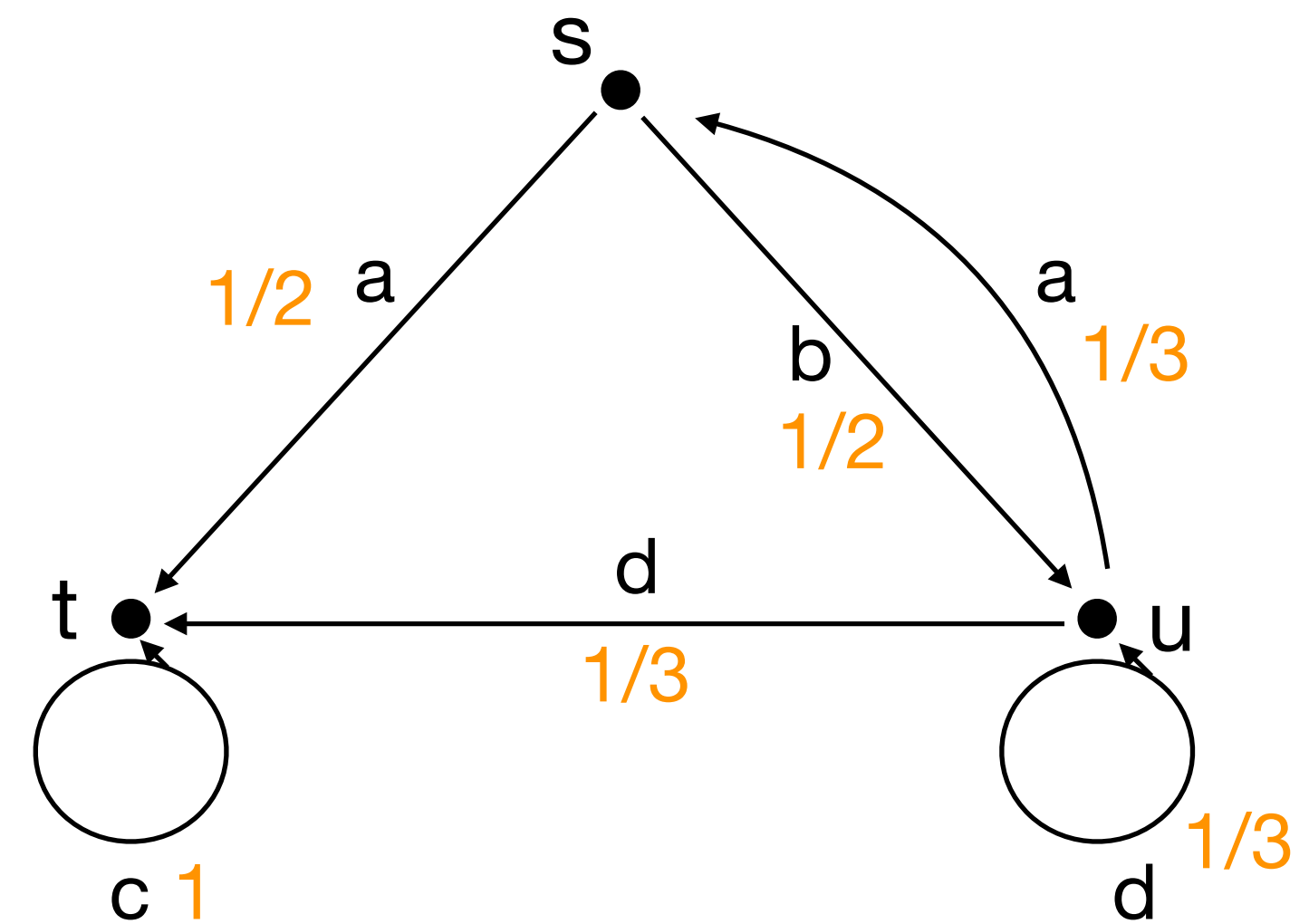
Finite Traces

$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



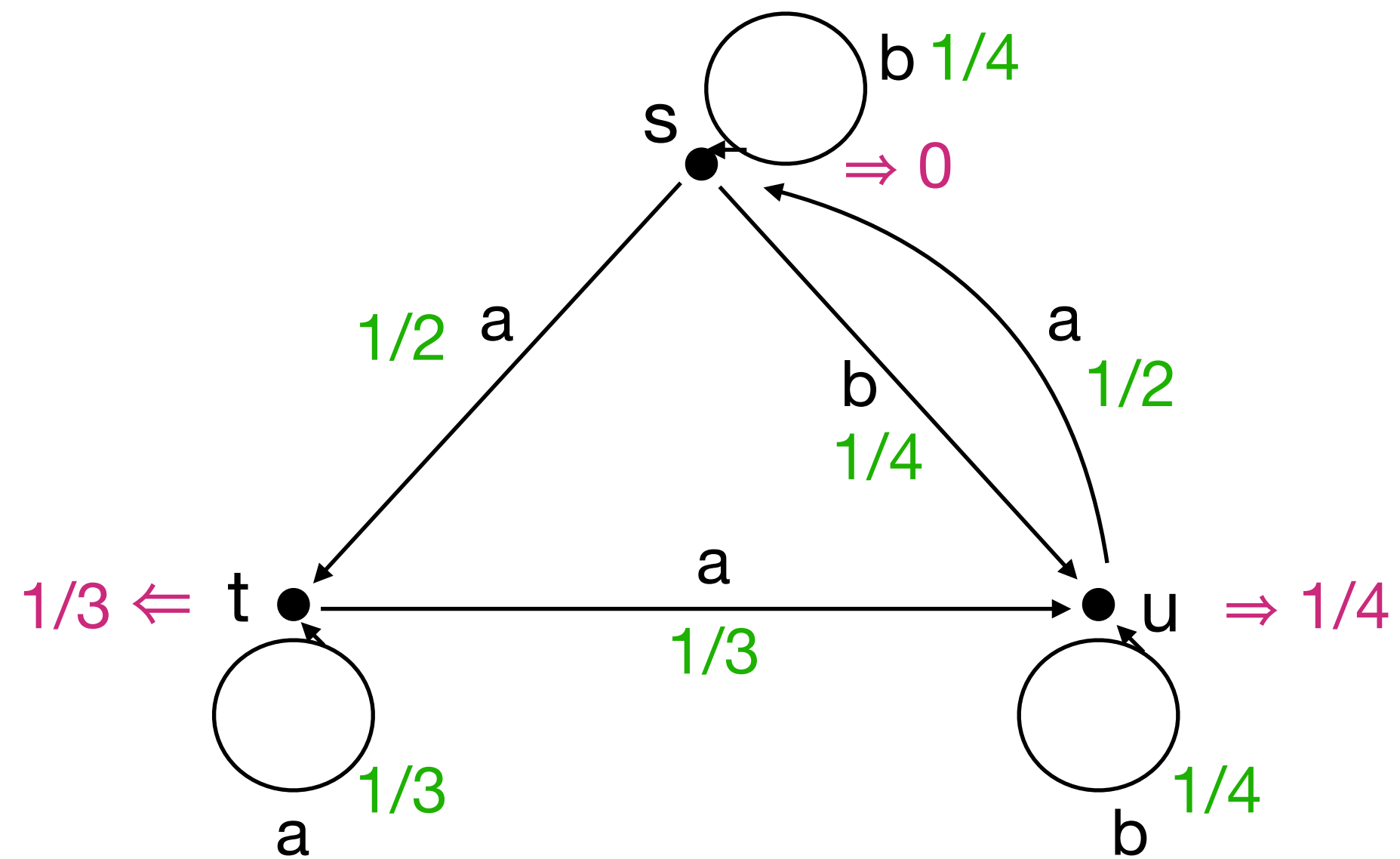
Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$



Finite Traces

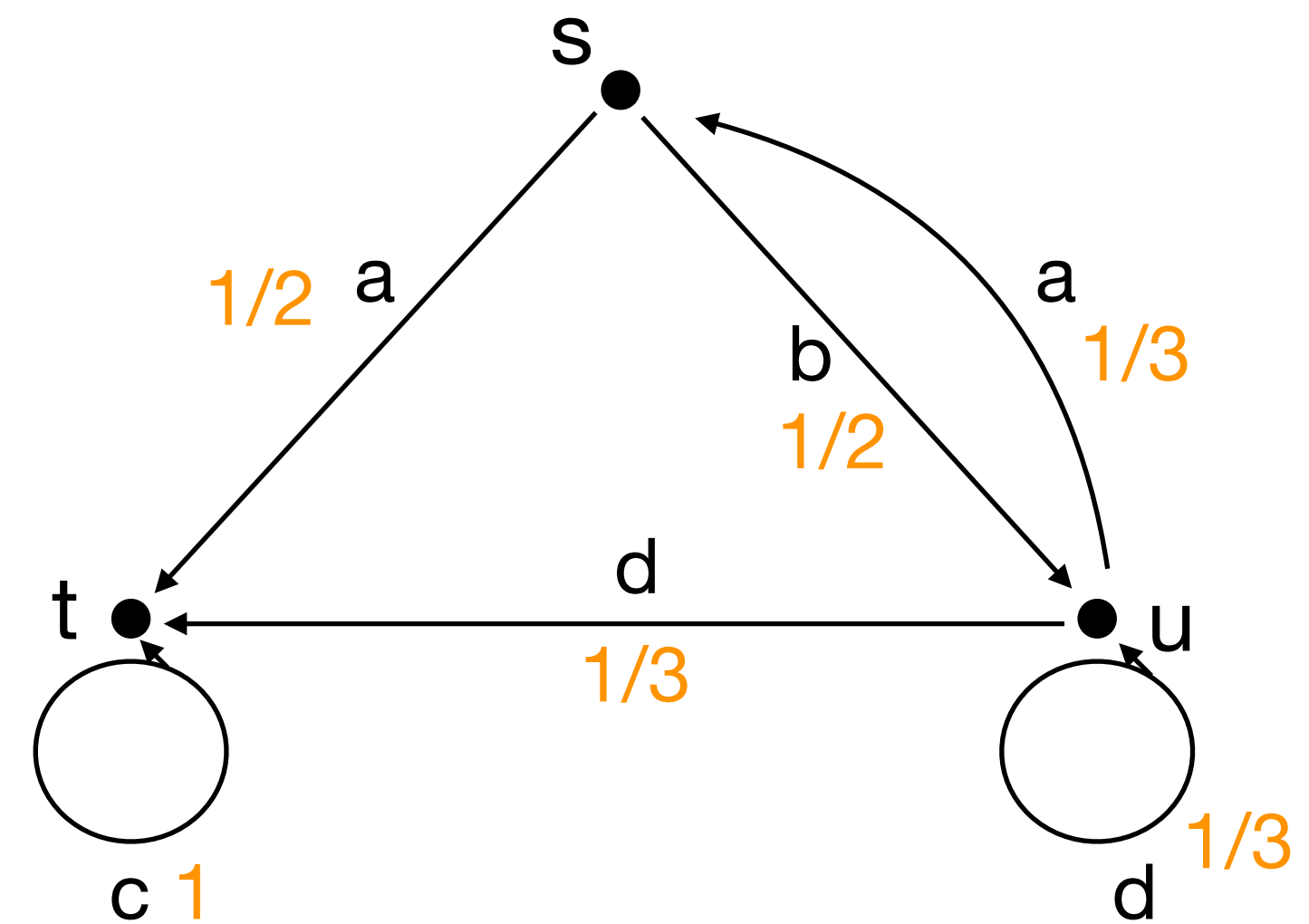
$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



$\text{trace}(s)(aaa) = \text{Prob}(sttt, sttu, stus)$

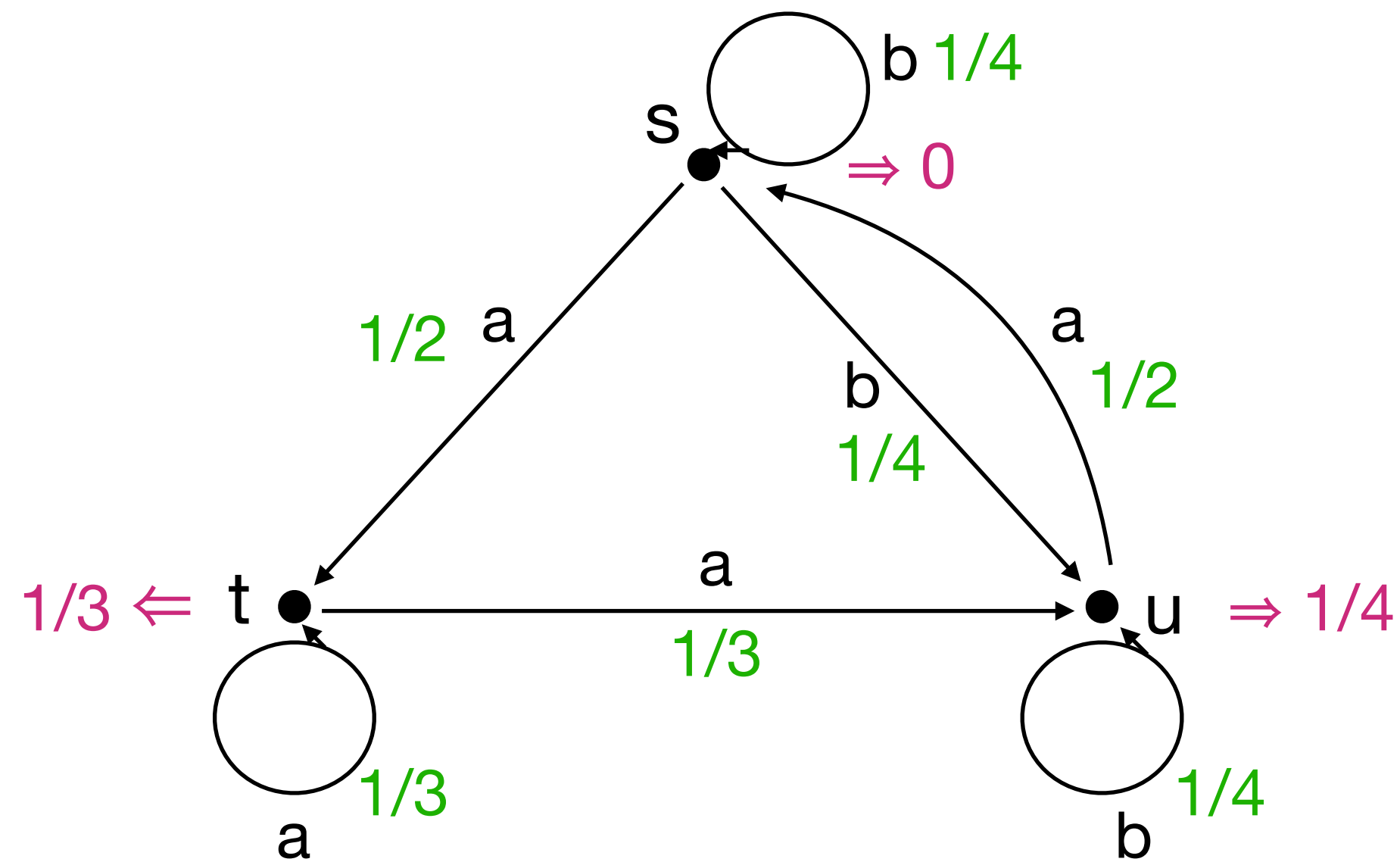
Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$



Finite Traces

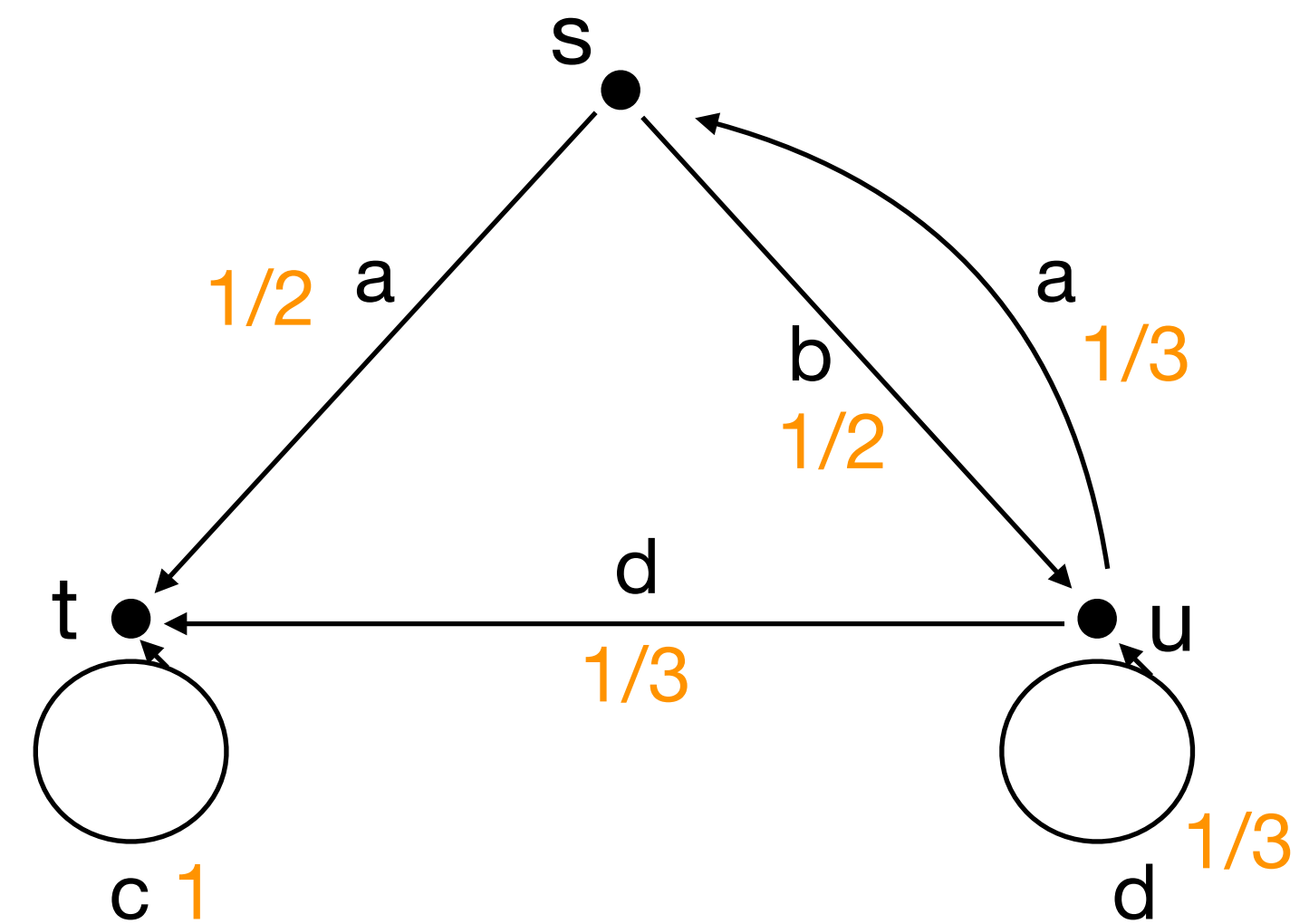
$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



$$\begin{aligned} \text{trace}(s)(aaa) &= \text{Prob}(\underline{\text{sttt}}, \text{sttu}, \text{stus}) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \end{aligned}$$

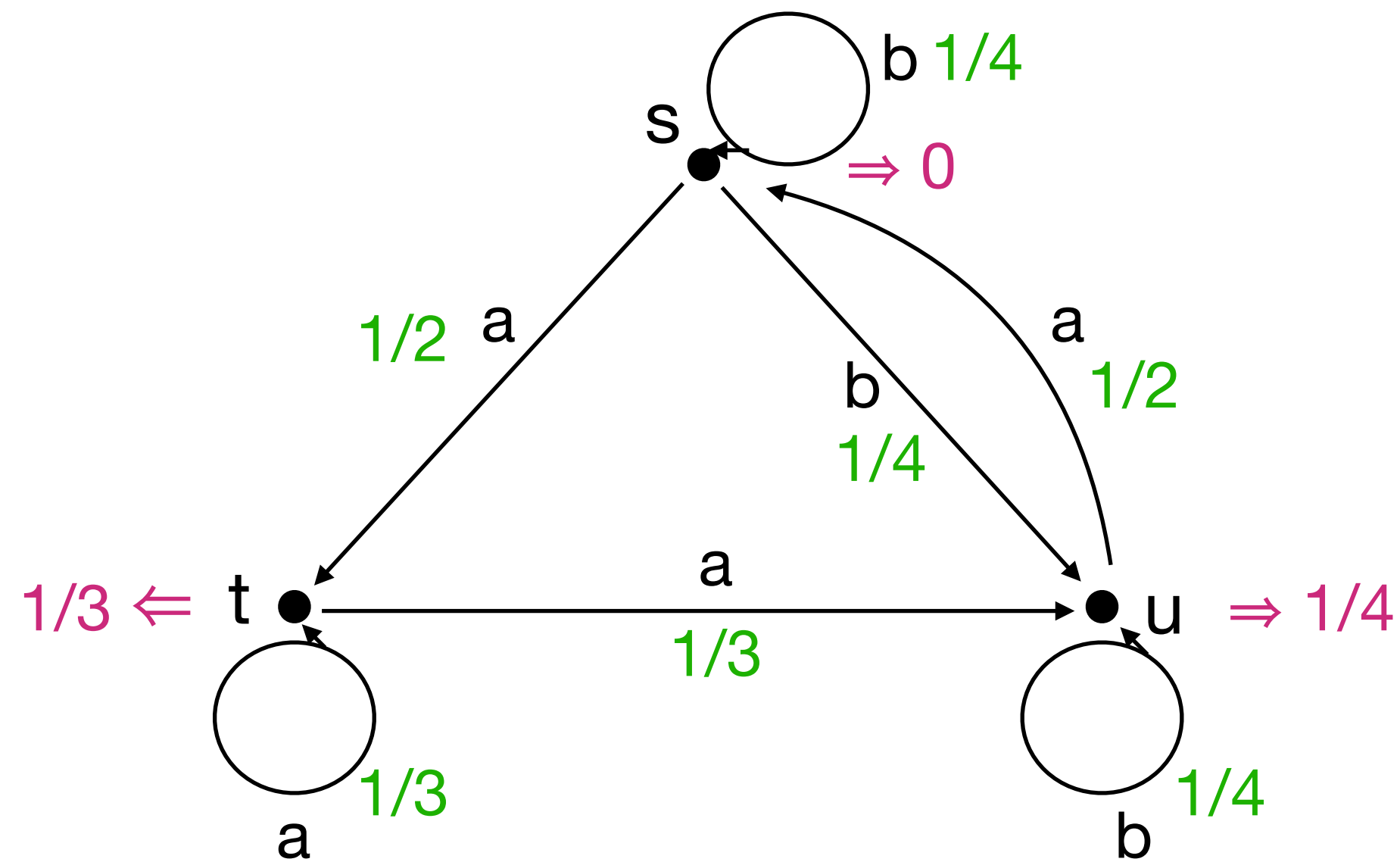
Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$



Finite Traces

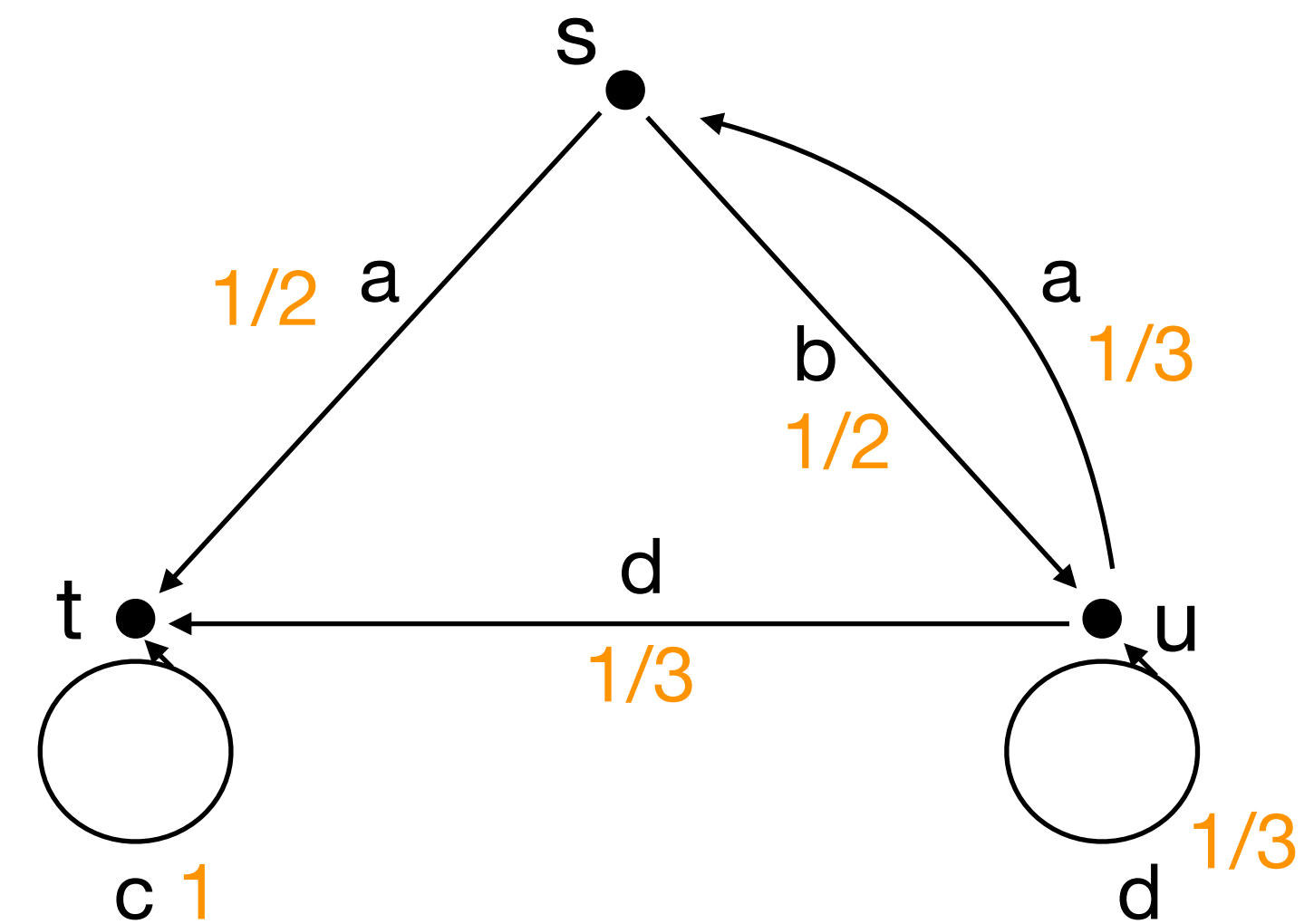
$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



$$\begin{aligned} \text{trace}(s)(aaa) &= \text{Prob}(\underline{\text{sttt}}, \underline{\text{sttu}}, \text{stus}) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{4} \end{aligned}$$

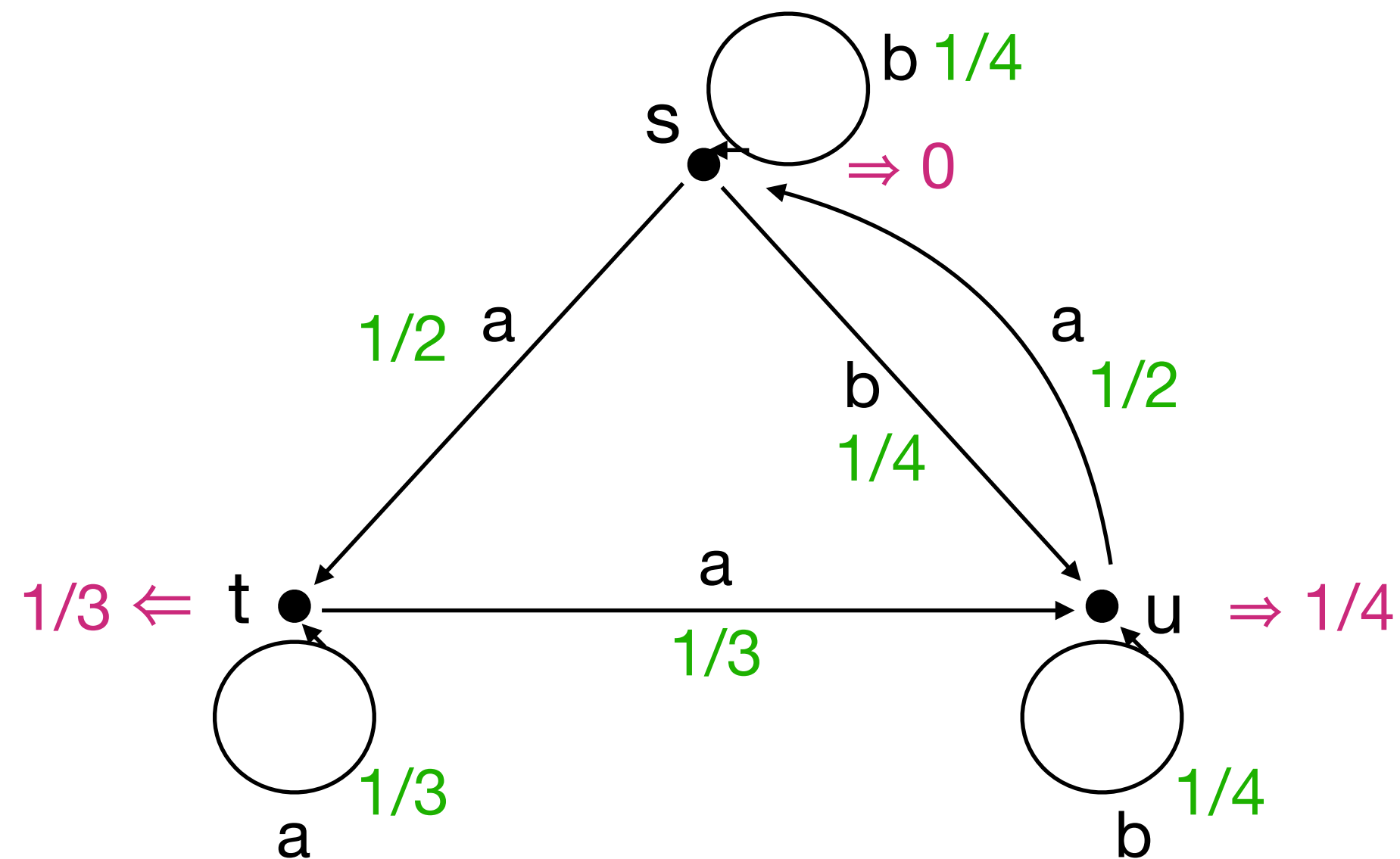
Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$



Finite Traces

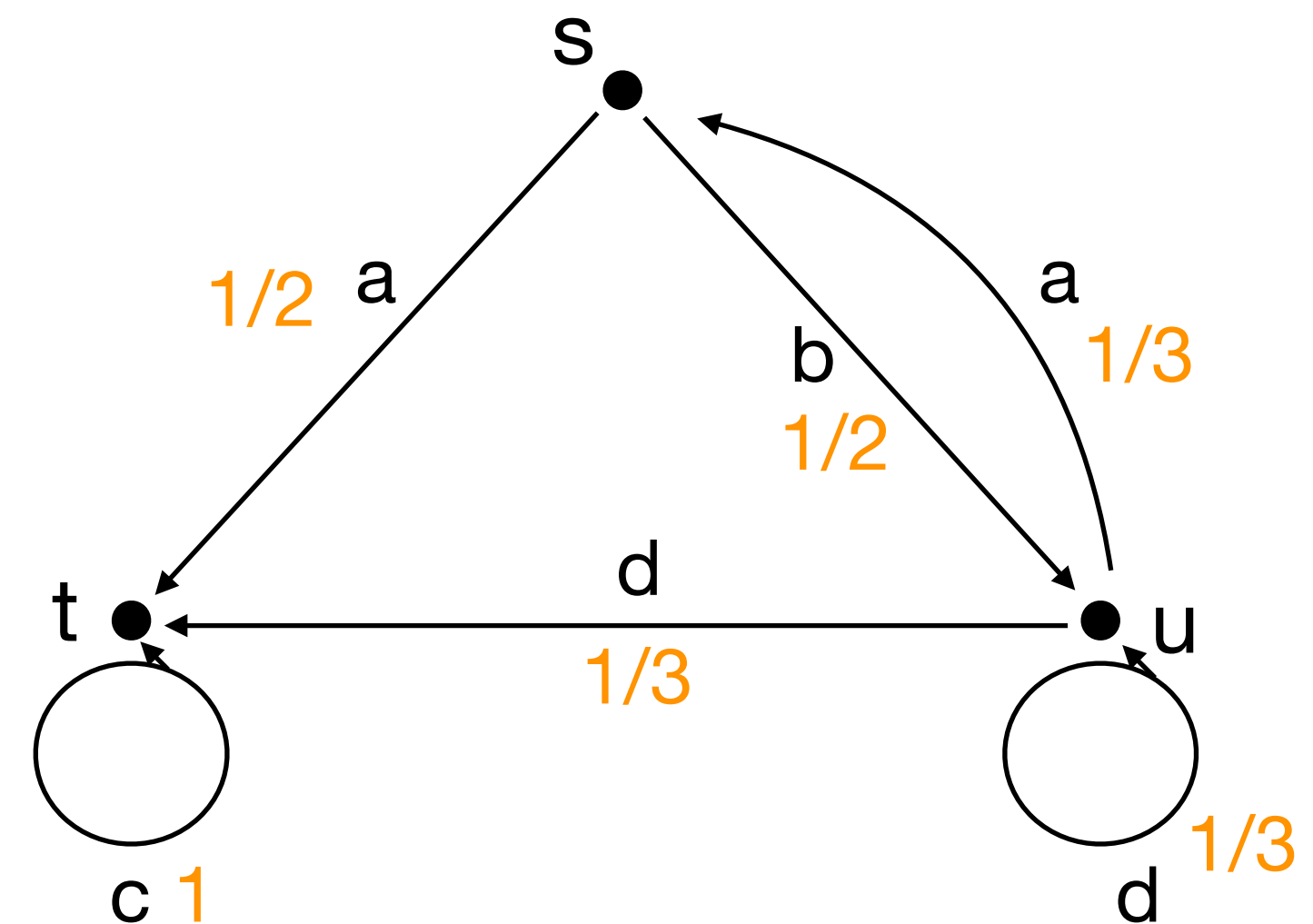
$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



$$\begin{aligned} \text{trace}(s)(aaa) &= \text{Prob}(\underline{\text{sttt}}, \underline{\text{sttu}}, \underline{\text{stus}}) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{4} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{2} \cdot 0 \end{aligned}$$

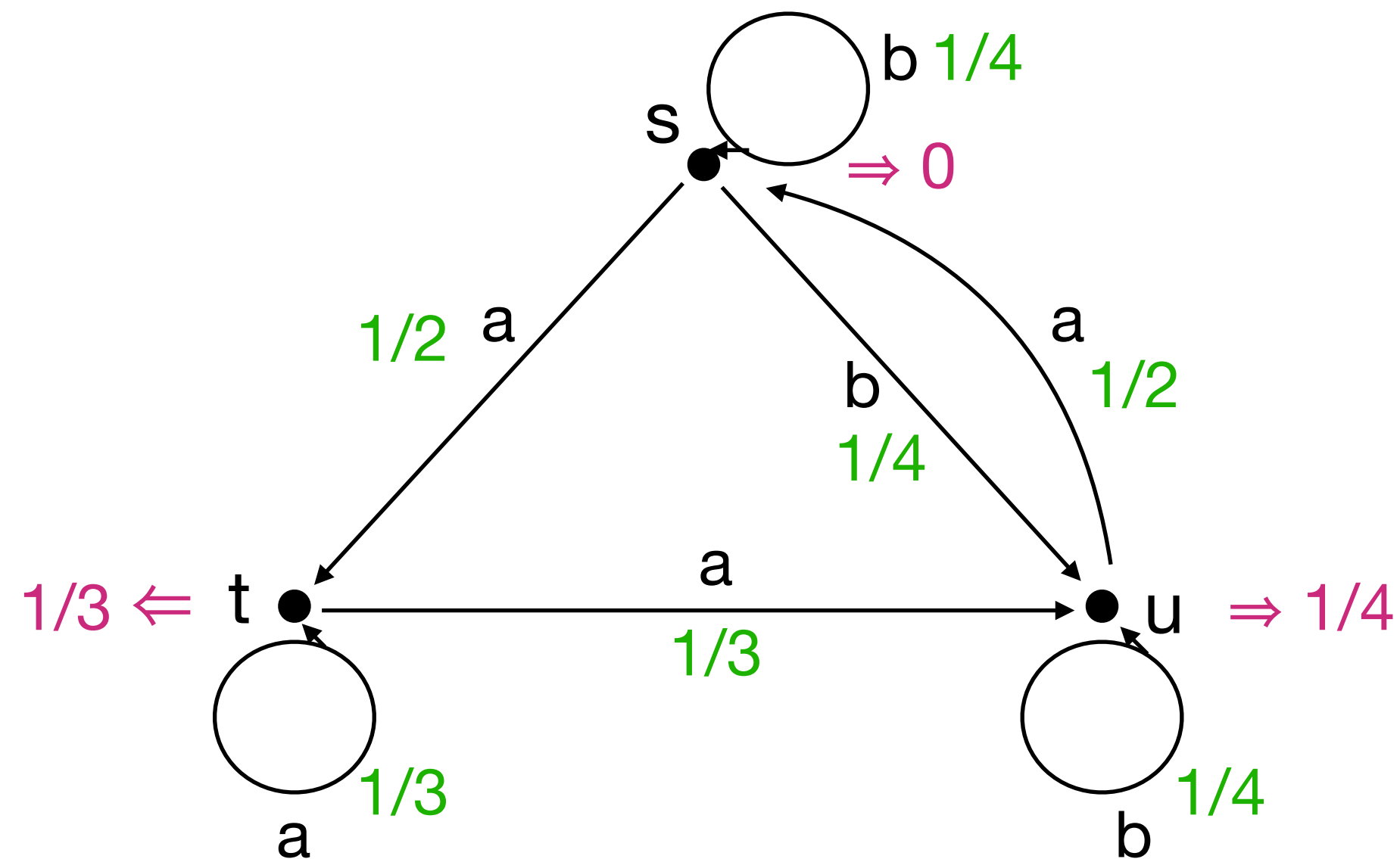
Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$



Finite Traces

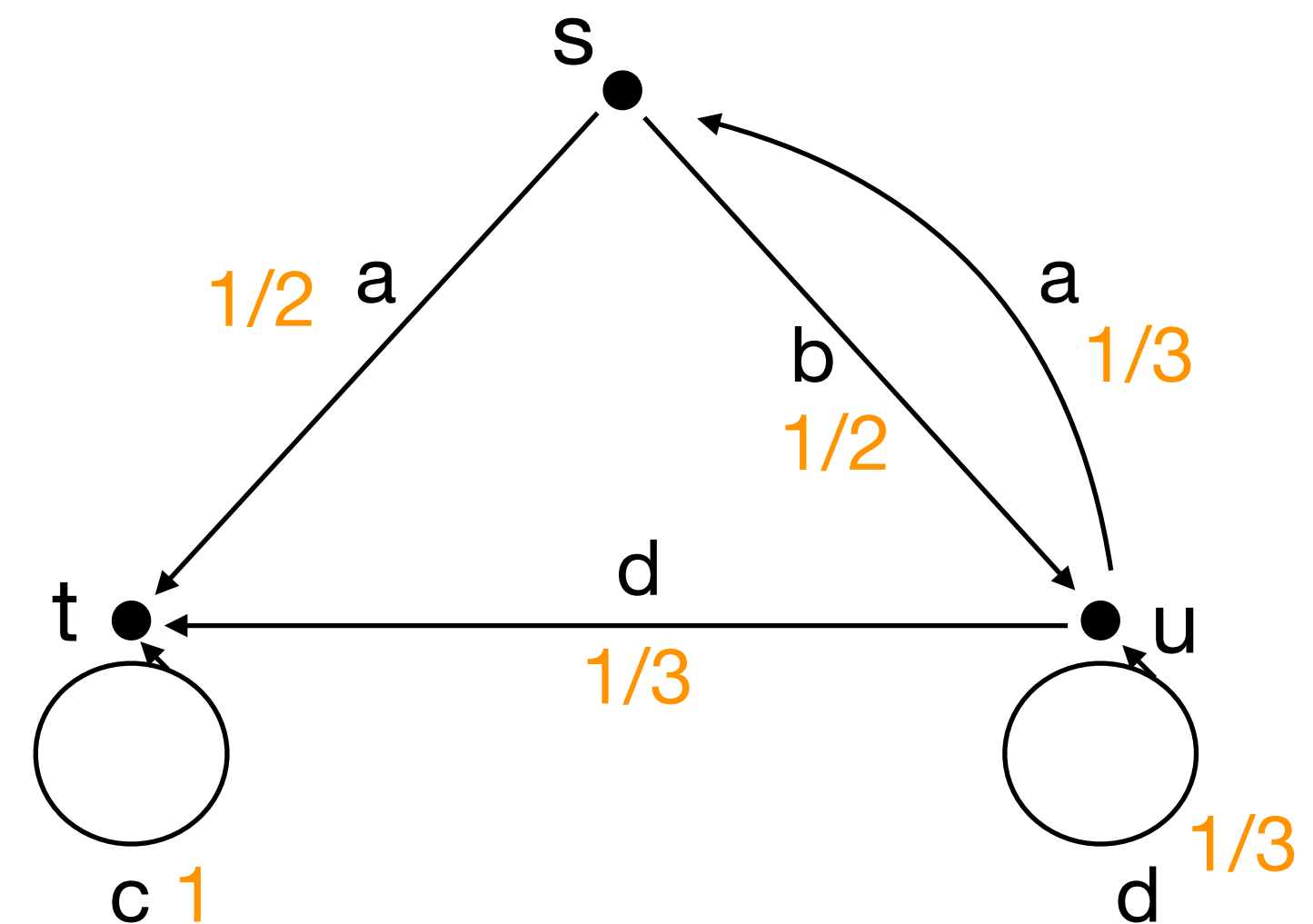
$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



$$\begin{aligned} \text{trace}(s)(aaa) &= \text{Prob}(\underline{\text{sttt}}, \underline{\text{sttu}}, \underline{\text{stus}}) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{4} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{2} \cdot 0 \end{aligned}$$

Infinite Traces

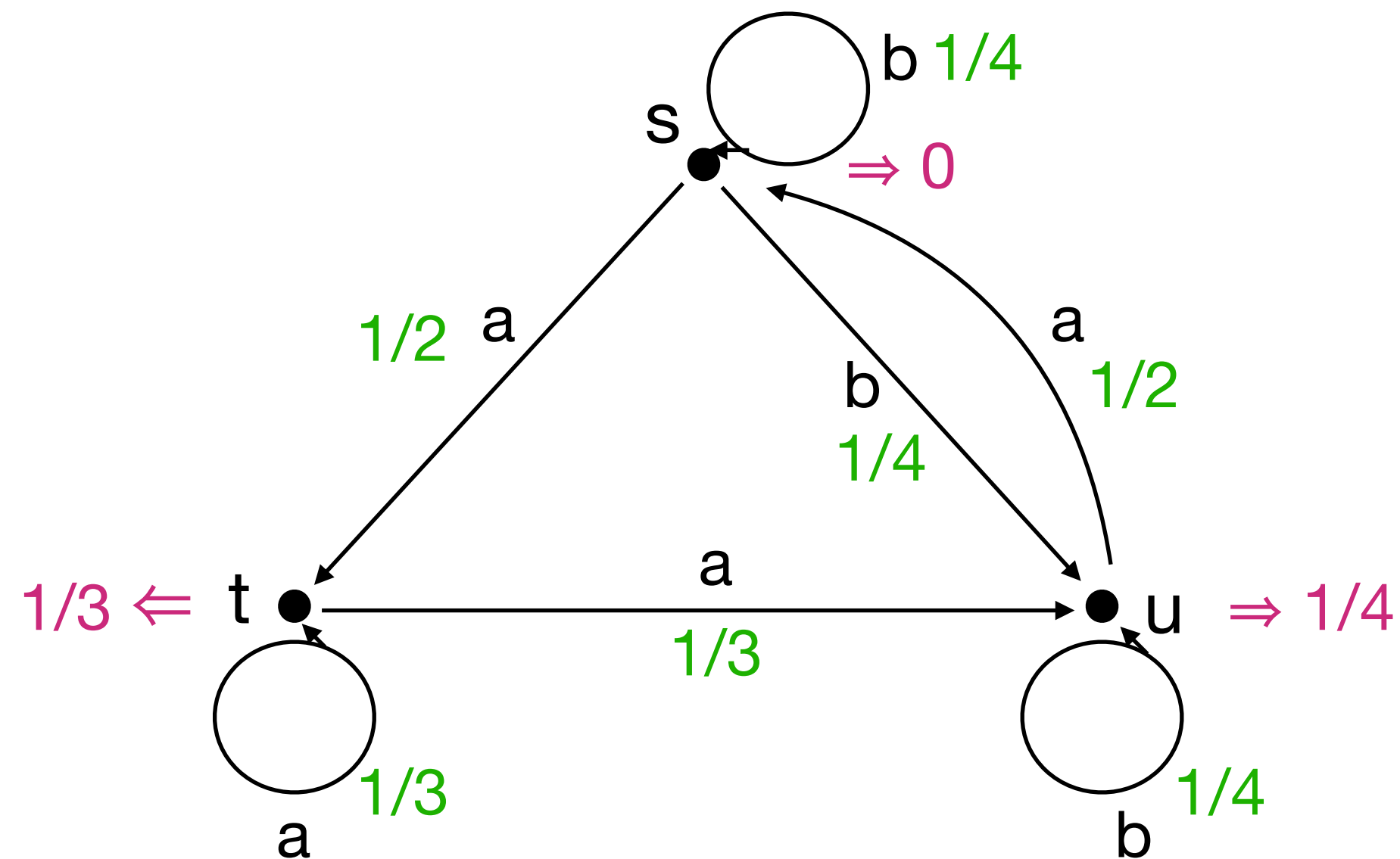
$$c: S \rightarrow \mathcal{D}(A \times S)$$



$$\text{trace}(s)(ac^\omega) = \frac{1}{2}$$

Finite Traces

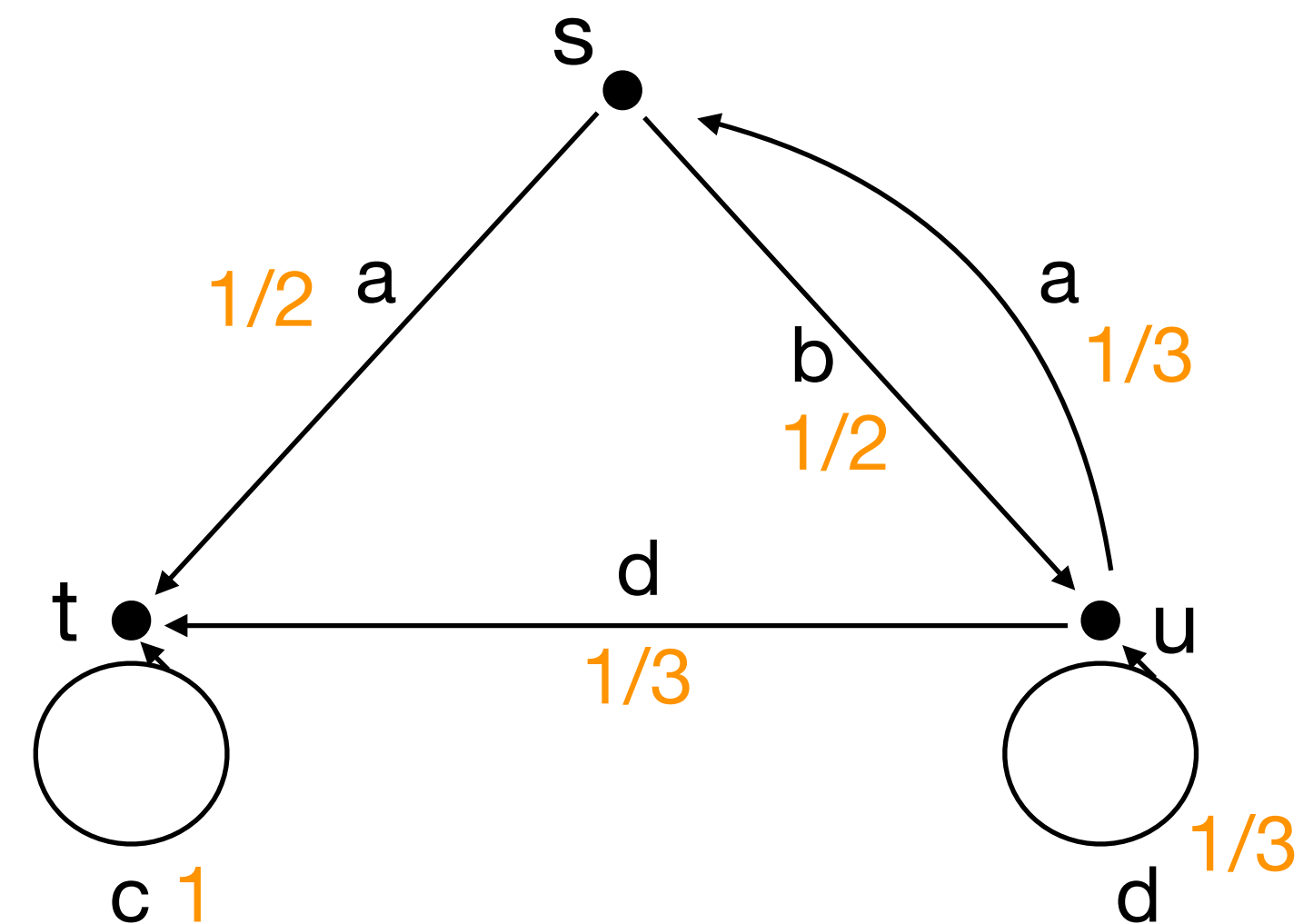
$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



$$\begin{aligned} \text{trace}(s)(aaa) &= \text{Prob}(\underline{\text{sttt}}, \underline{\text{sttu}}, \underline{\text{stus}}) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{4} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{2} \cdot 0 \end{aligned}$$

Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$

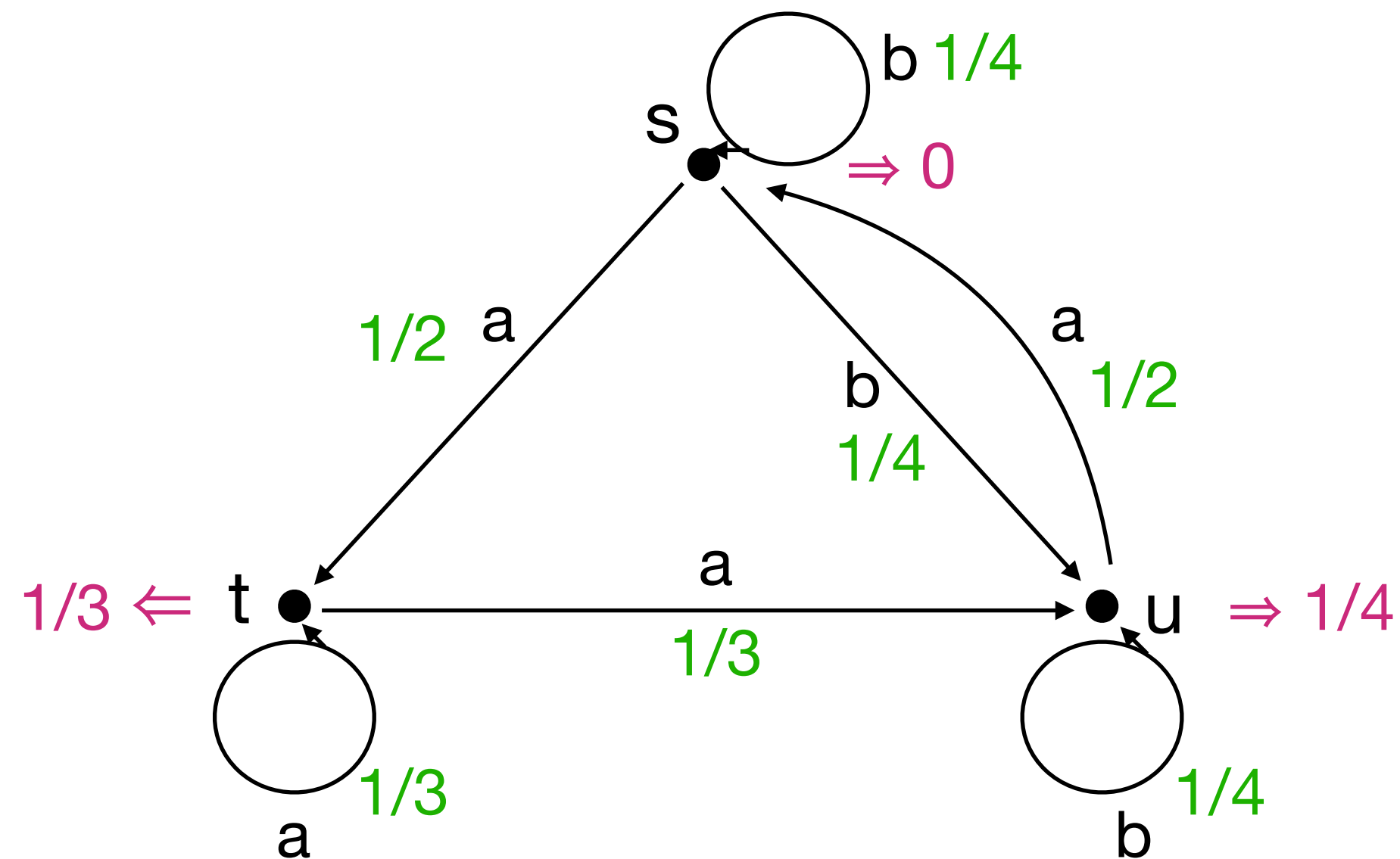


$$\text{trace}(s)(ac^\omega) = \frac{1}{2}$$

$$\text{trace}(s)(bd^\omega) = \lim_{n \rightarrow \infty} \frac{1}{2} \cdot \left(\frac{1}{3}\right)^n = 0$$

Finite Traces

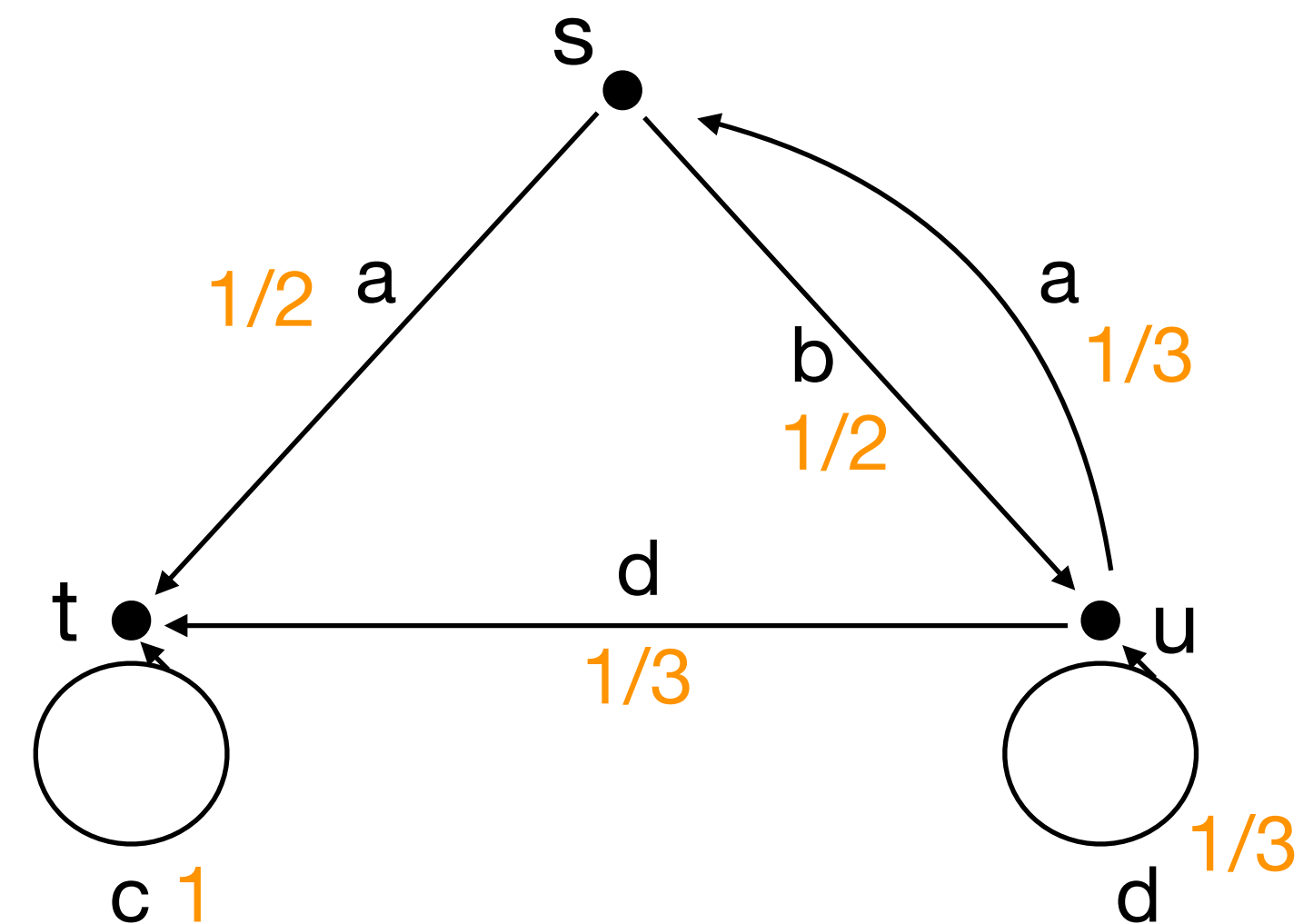
$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



$$\begin{aligned} \text{trace}(s)(aaa) &= \text{Prob}(\underline{\text{sttt}}, \underline{\text{sttu}}, \underline{\text{stus}}) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{4} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{2} \cdot 0 \end{aligned}$$

Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$



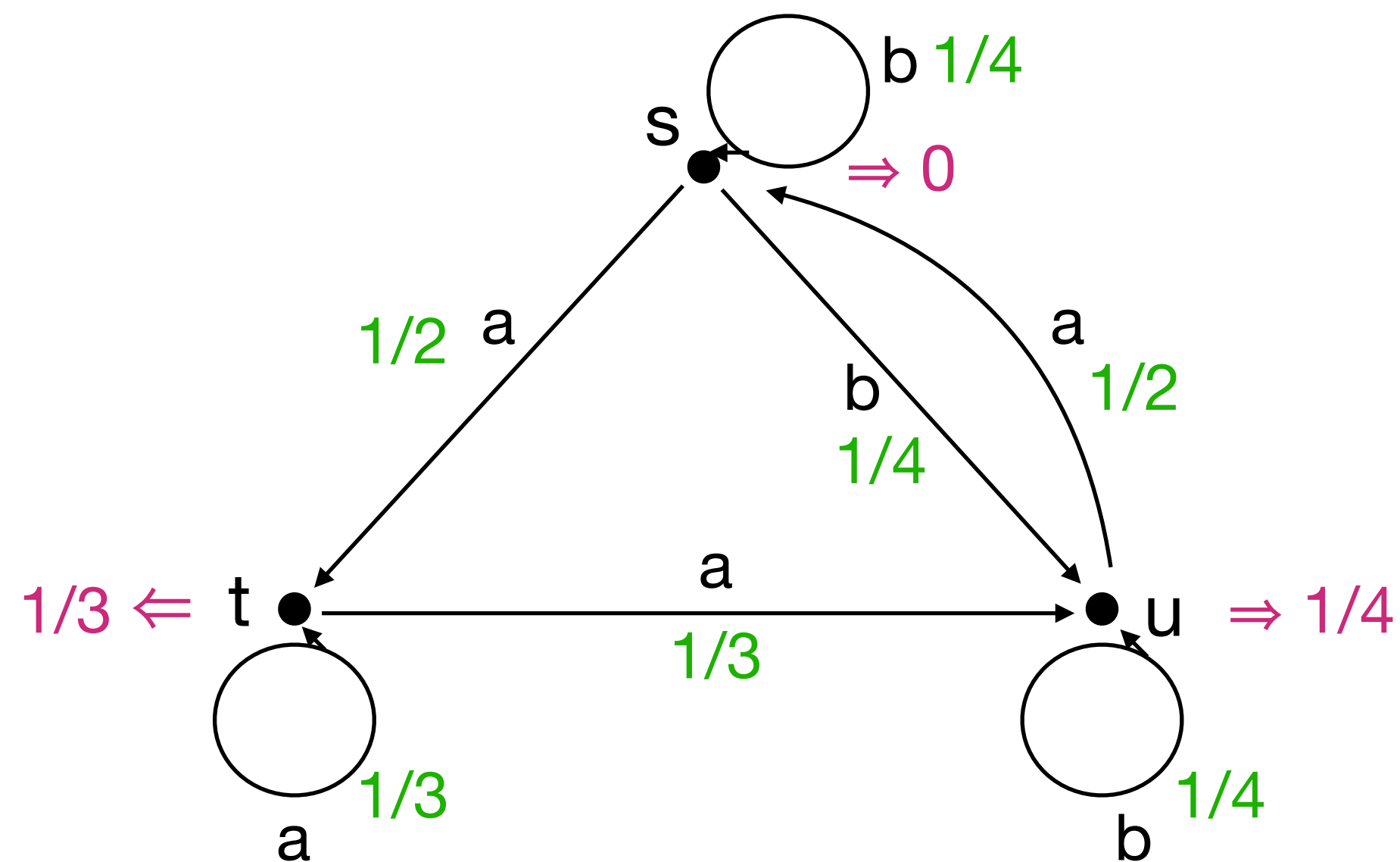
$$\text{trace}(s)(ac^{\omega}) = \frac{1}{2}$$

$$\text{trace}(s)(bd^{\omega}) = \lim_{n \rightarrow \infty} \frac{1}{2} \cdot \left(\frac{1}{3}\right)^n = 0$$

Therefore, measurable spaces !

Finite Traces

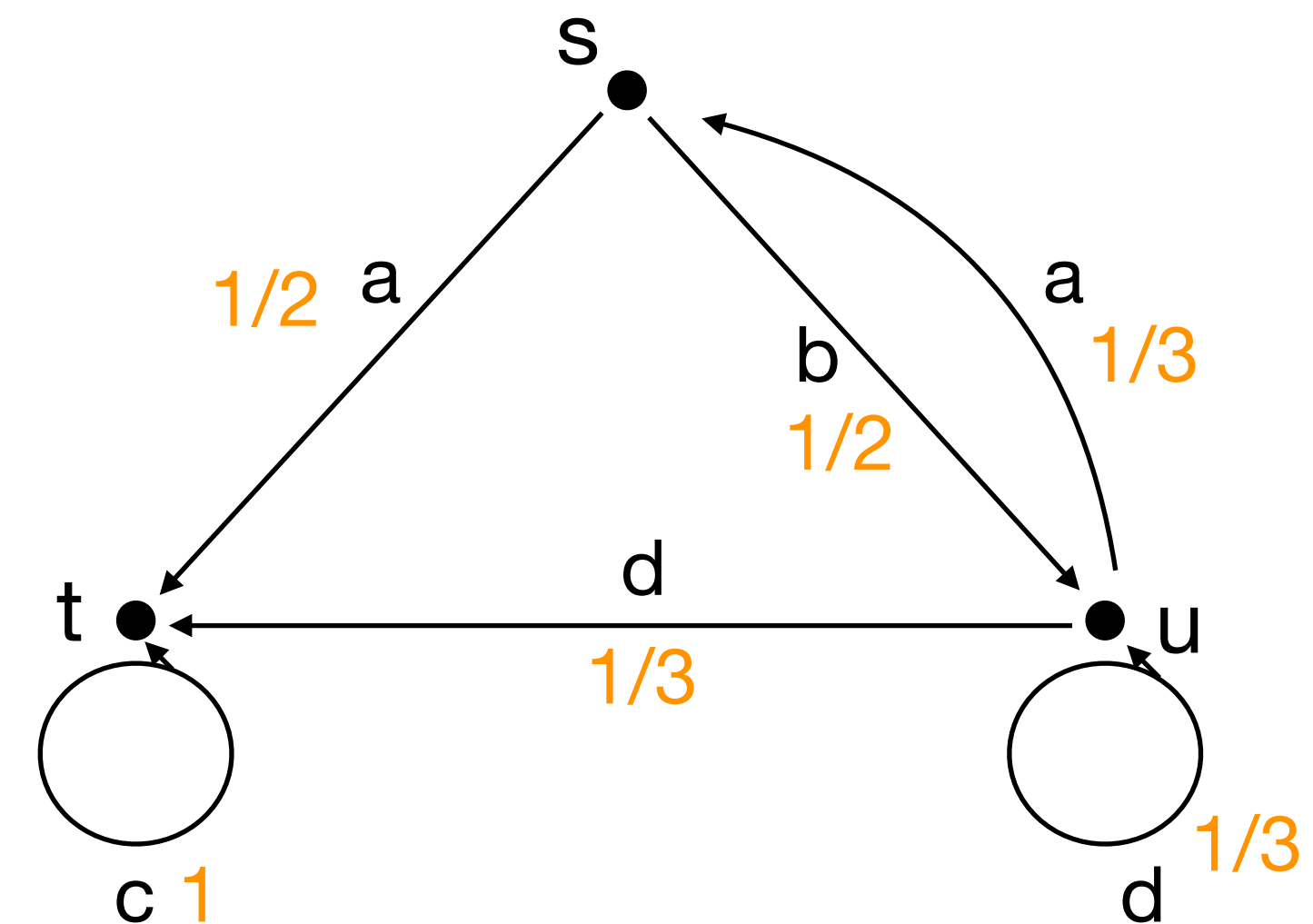
$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



$$\begin{aligned} \text{trace}(s)(aaa) &= \text{Prob}(\underline{\text{sttt}}, \underline{\text{sttu}}, \underline{\text{stus}}) \\ &= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{4} \\ &\quad + \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{2} \cdot 0 \end{aligned}$$

Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$

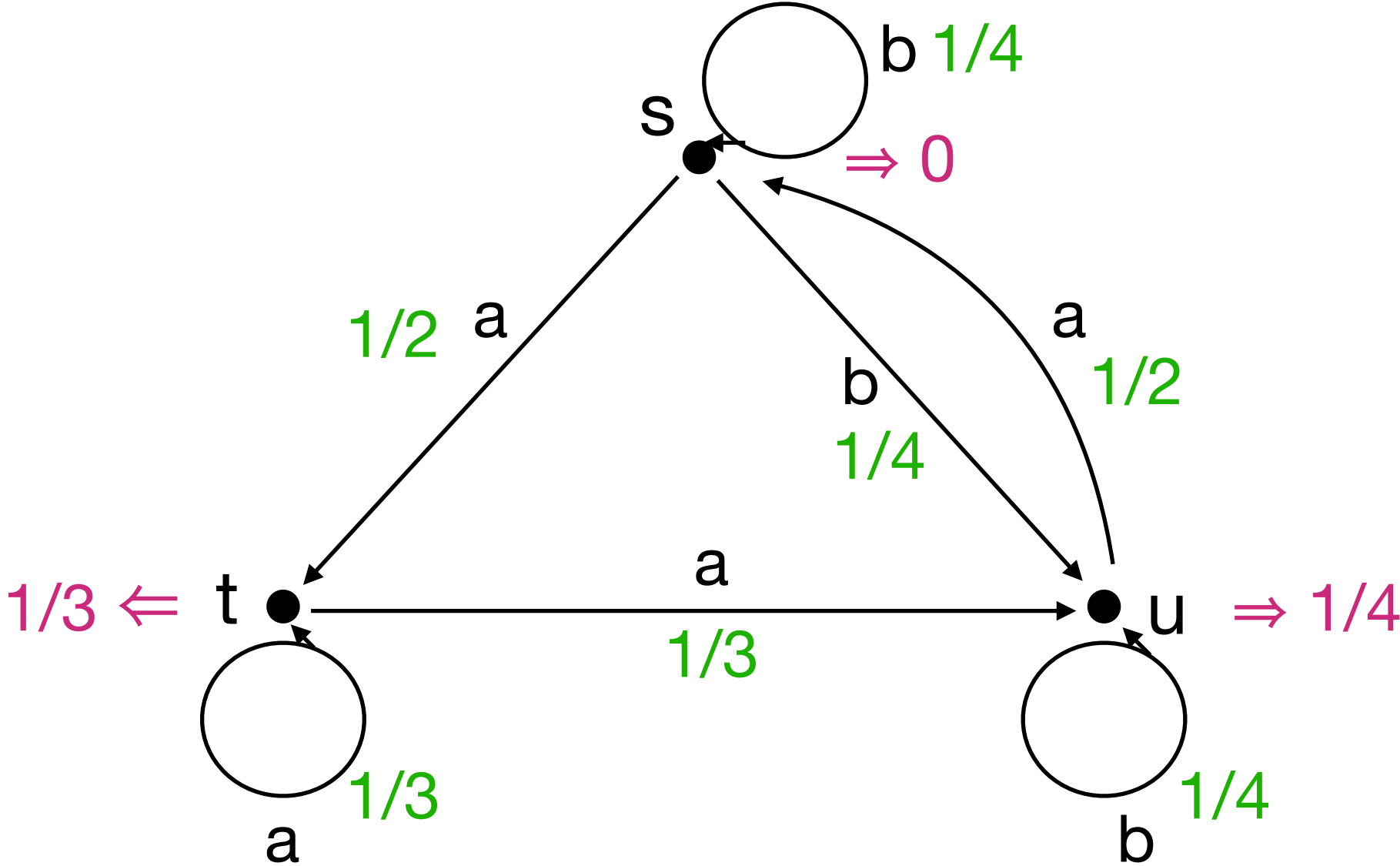


Each state defines a Borel probability measure on A^ω , $\text{trace}(s) \in \text{Prob}(A^\omega)$

with the σ -algebra generated by the cylinders $B_w = \{ \alpha \in A^\omega \mid w \text{ is a prefix of } \alpha \}$

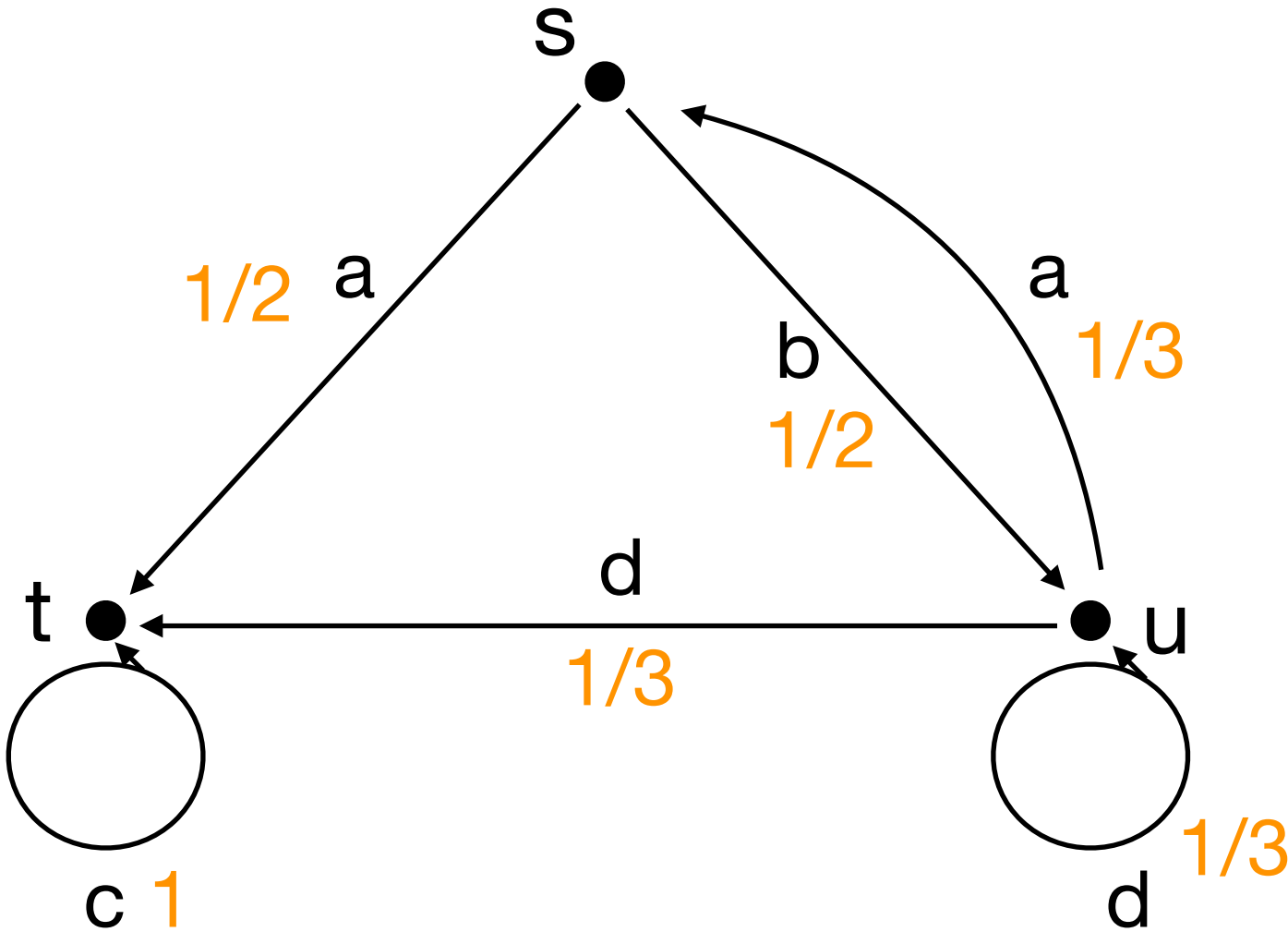
Finite Traces

$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



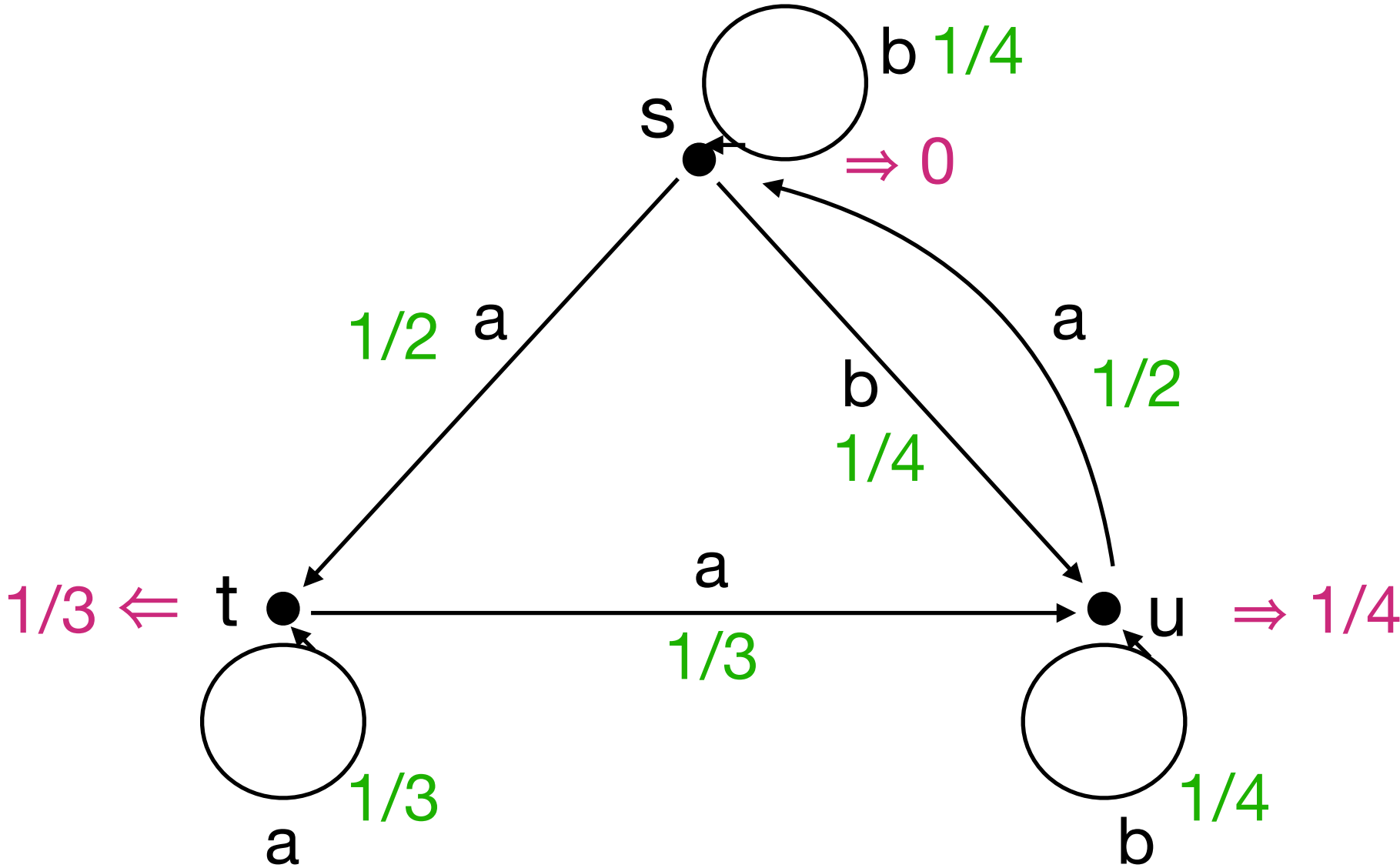
Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$



Finite Traces

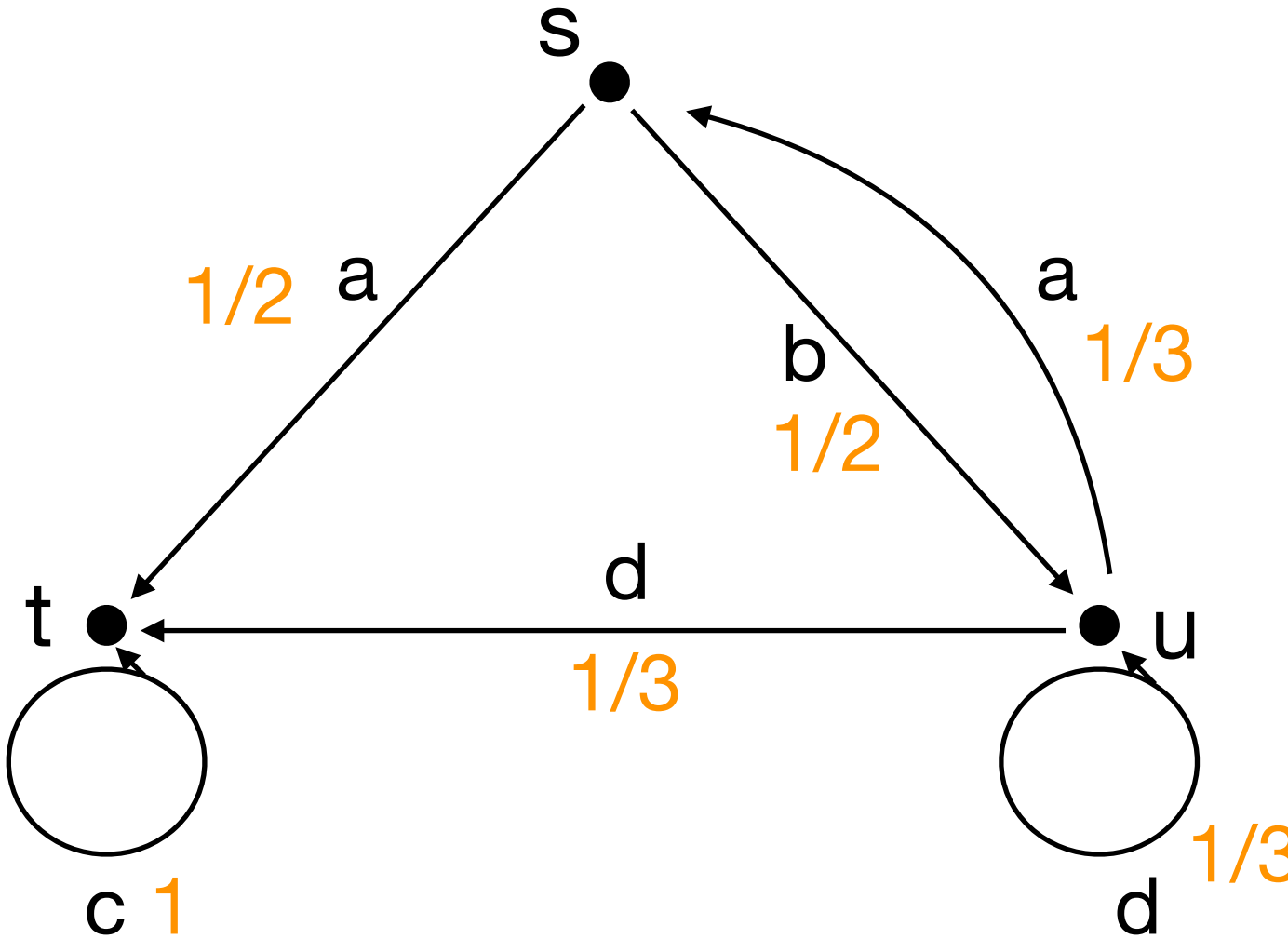
$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$



$$\text{trace}(s)(\epsilon) = c(s)(*) = \text{output-at-}s$$

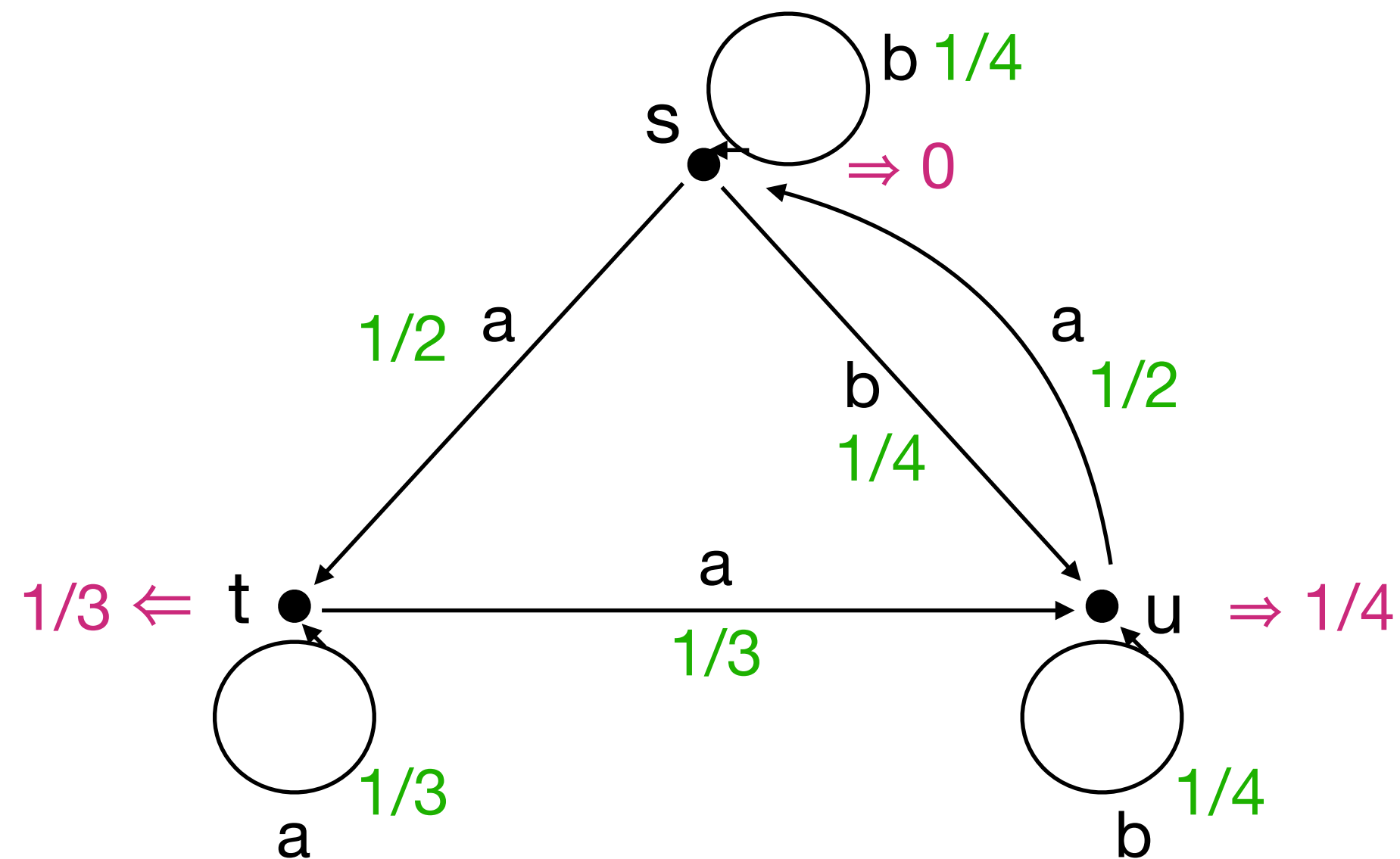
Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$



Finite Traces

$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$

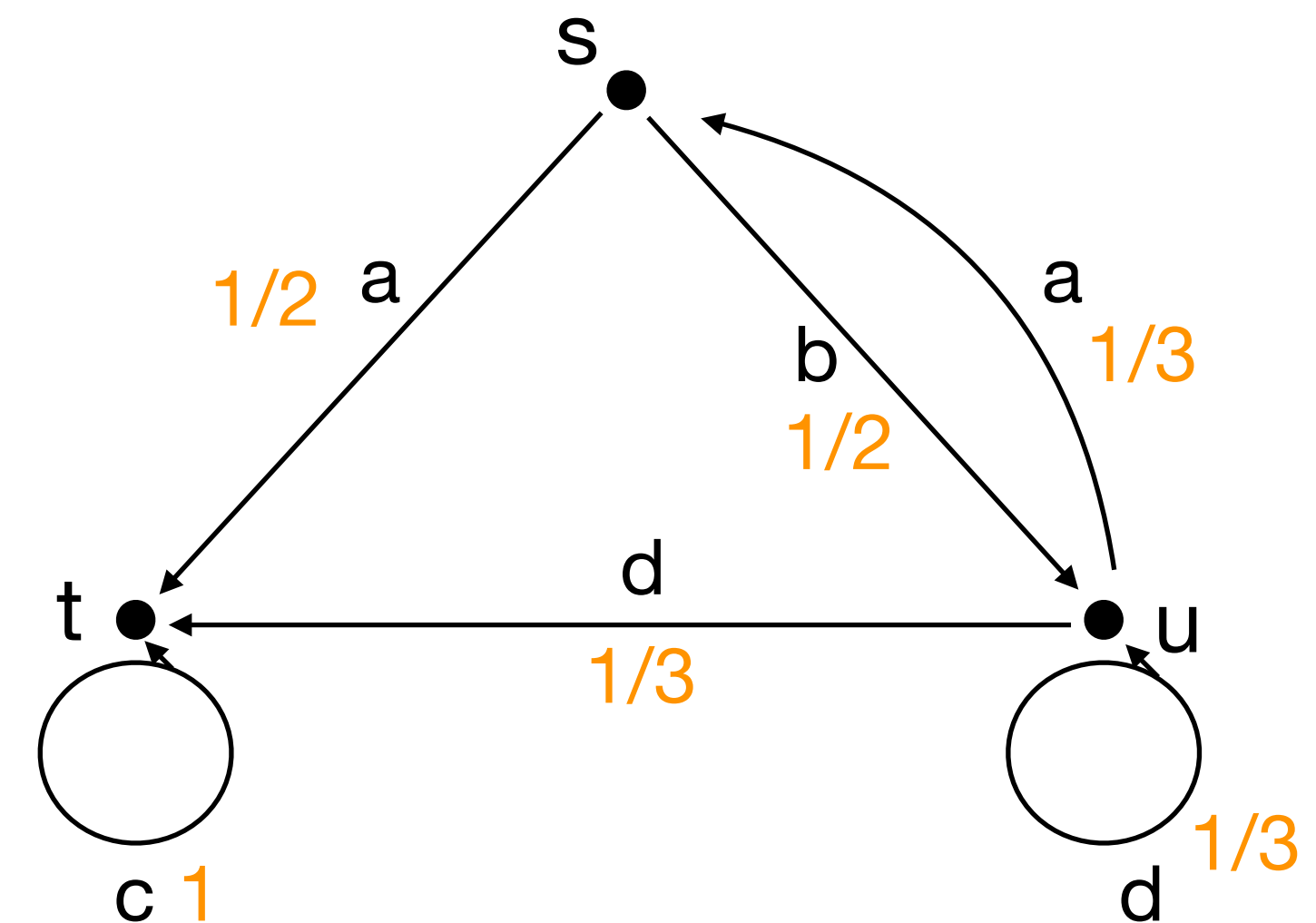


$$\text{trace}(s)(\epsilon) = c(s)(*) = \text{output-at-}s$$

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

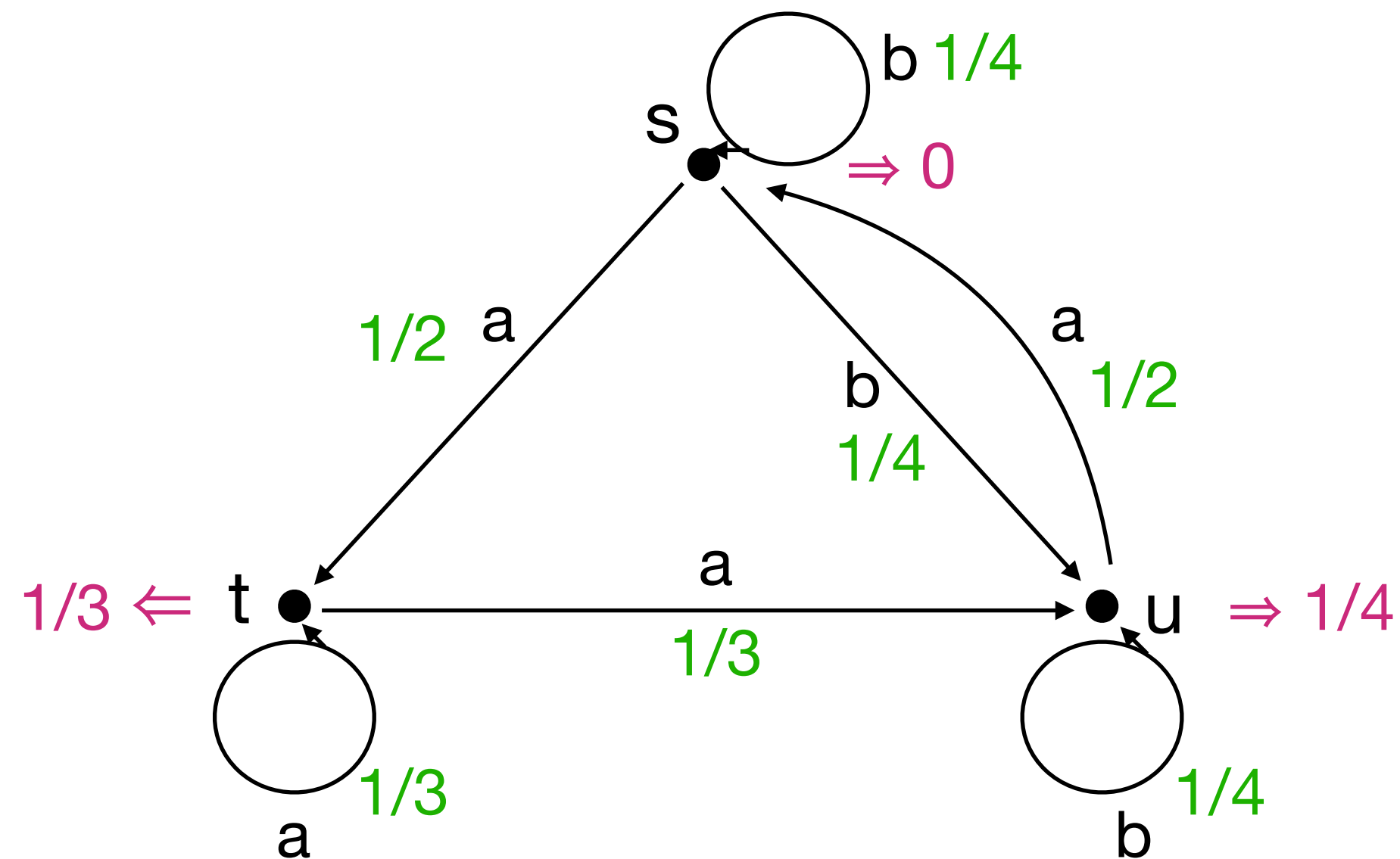
Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$



Finite Traces

$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$

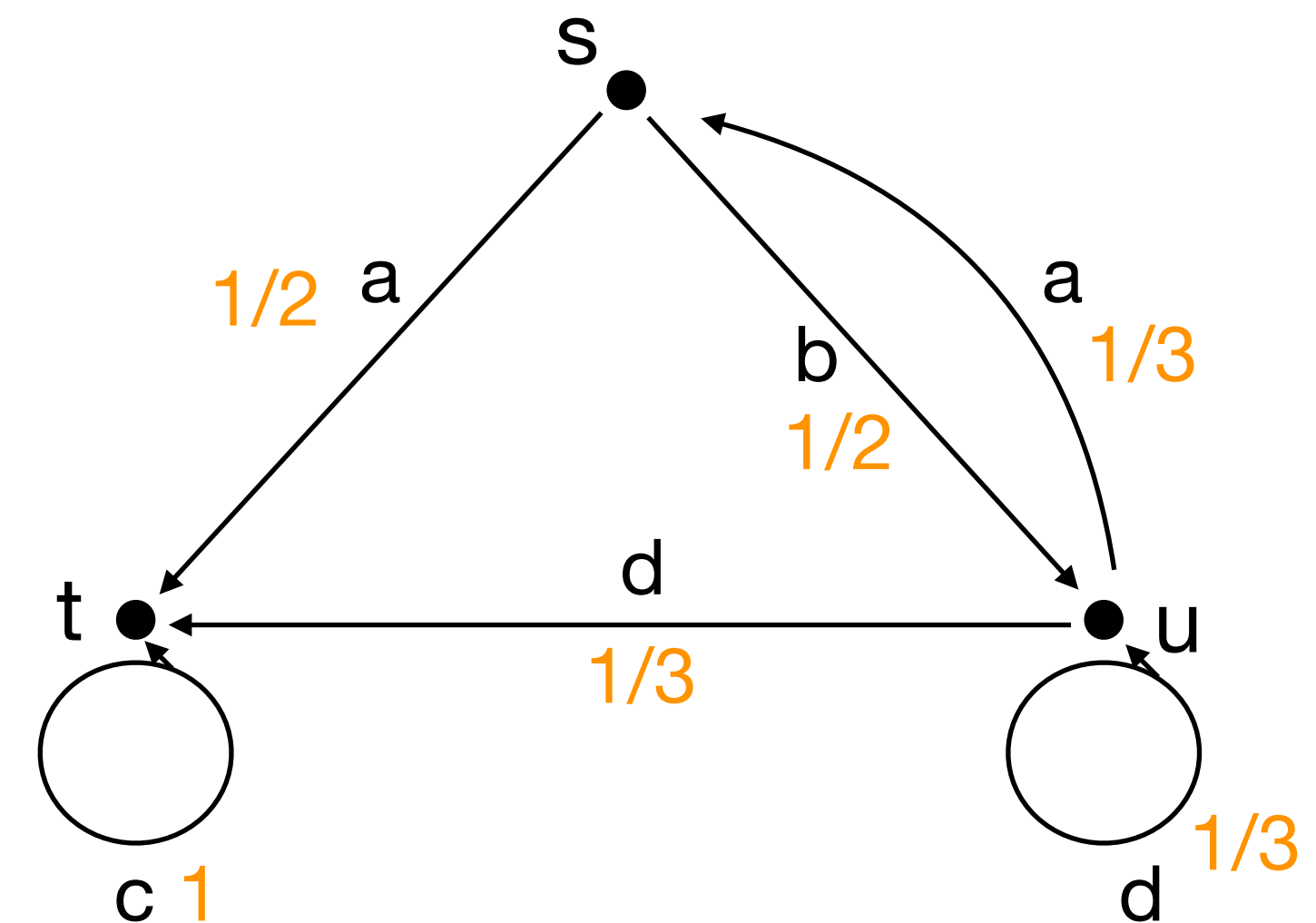


$$\text{trace}(s)(\varepsilon) = c(s)(*) = \text{output-at-}s$$

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

Infinite Traces

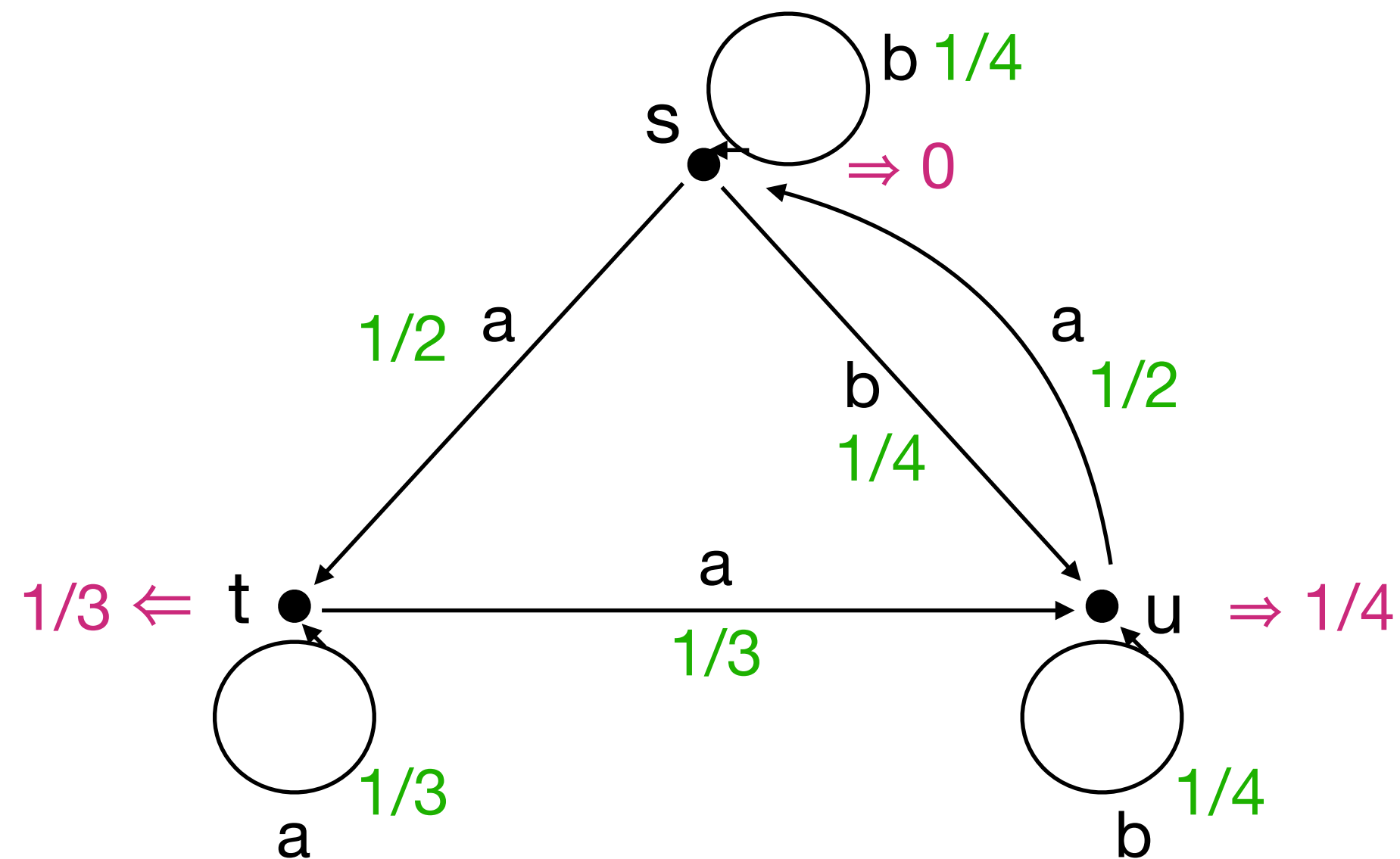
$$c: S \rightarrow \mathcal{D}(A \times S)$$



$$\text{trace}(s)(B_{\varepsilon}) = 1$$

Finite Traces

$$c: S \rightarrow \mathcal{D}_{\leq}(A \times S)$$

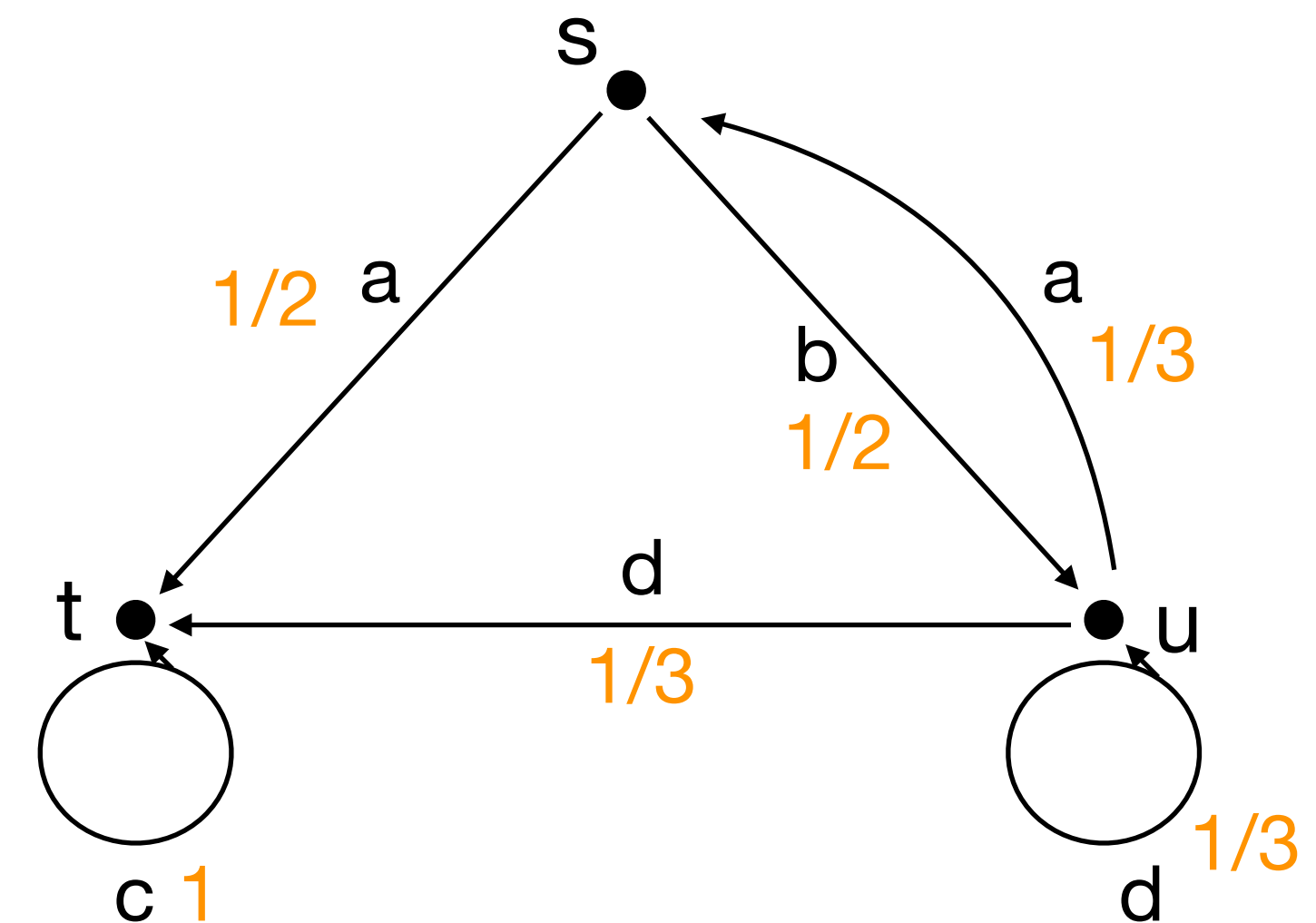


$$\text{trace}(s)(\varepsilon) = c(s)(*) = \text{output-at-}s$$

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

Infinite Traces

$$c: S \rightarrow \mathcal{D}(A \times S)$$



$$\text{trace}(s)(B_{\varepsilon}) = 1$$

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

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:

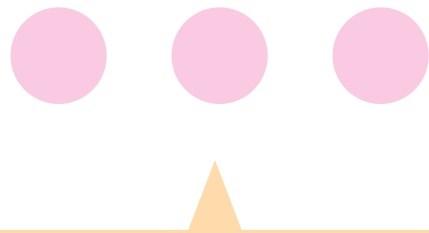
$$s =_{tr} t \iff \text{trace}(s) = \text{trace}(t)$$

Trace Equivalence and History

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

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

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 $s =_{\text{tr}} t$, if and only if they have the same trace:

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

Previous work on infinite trace equivalence (coalgebraically):

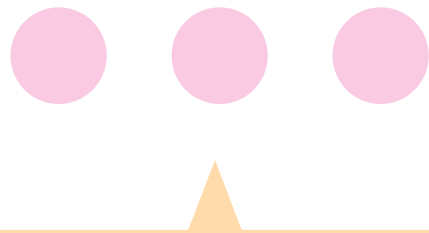
Jacobs '04

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 $s =_{\text{tr}} t$, if and only if they have the same trace:

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

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):

Trace Equivalence and History

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

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

Previous work on infinite trace equivalence (coalgebraically):

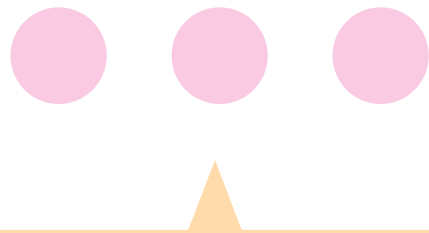
Jacobs '04

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

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

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

Expressions

Expressions for labelled Markov chains: Stark & Smolka '00

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

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

Expressions

Expressions for labelled Markov chains: Stark & Smolka '00

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

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

Process terms are expressions that are guarded and closed.

Expressions

Expressions for labelled Markov chains: Stark & Smolka '00

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

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

Process terms are expressions that are guarded and closed.

every v in
scope of $a.(-)$

Expressions

Expressions for labelled Markov chains: Stark & Smolka '00

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

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 $\text{fix } v.(-)$

Expressions

Expressions for labelled Markov chains: Stark & Smolka '00

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

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 $\text{fix } v.(-)$

Process terms behave:

$$\gamma : \text{PTerms} \rightarrow \mathcal{D}(A \times \text{PTerms})$$

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 $\text{fix } v.(-)$

Process terms behave:

$$\gamma : \text{PTerms} \rightarrow \mathcal{D}(A \times \text{PTerms})$$

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 $\text{fix } v.(-)$

Process terms behave:

$$\gamma : \text{PTerms} \rightarrow \mathcal{D}(A \times \text{PTerms})$$

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

$$s =_{\text{tr}} t_s$$

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

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

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

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

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

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

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

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

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

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

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

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

fixpoints

Theorem (Soundness&Completeness):

Axioms

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

$$\begin{aligned} e \oplus_1 f &= e \\ e \oplus_p e &= e \\ e \oplus_p f &= f \oplus_{1-p} e \\ e \oplus_p (f \oplus_q g) &= (e \oplus_{pq} f) \oplus_{q(1-p)/(1-pq)} g \end{aligned}$$

convex algebras

Basically the same from
Silva & S. '11

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

distributivity

$$\begin{aligned} \text{fix } v.e &= e[v := \text{fix } v.e] \\ g = e[v := g] &\Rightarrow g = \text{fix } v.e \end{aligned}$$

fixpoints

Theorem (Soundness&Completeness):

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

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

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

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

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

fixpoints

Theorem (Soundness & Completeness):

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

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

The Role of Algebras

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

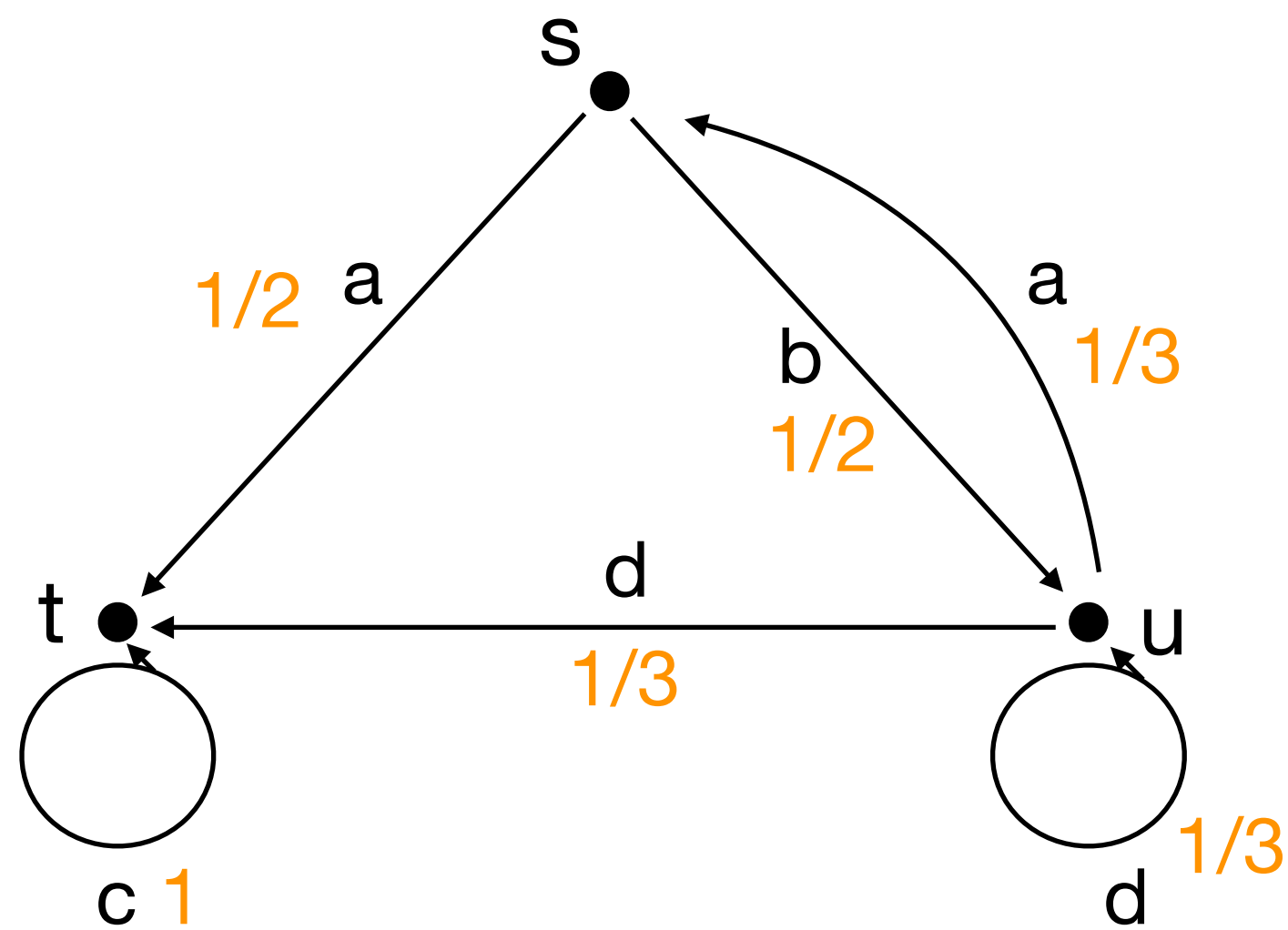
Eilenberg-Moore
algebras for $\mathcal{D}$

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



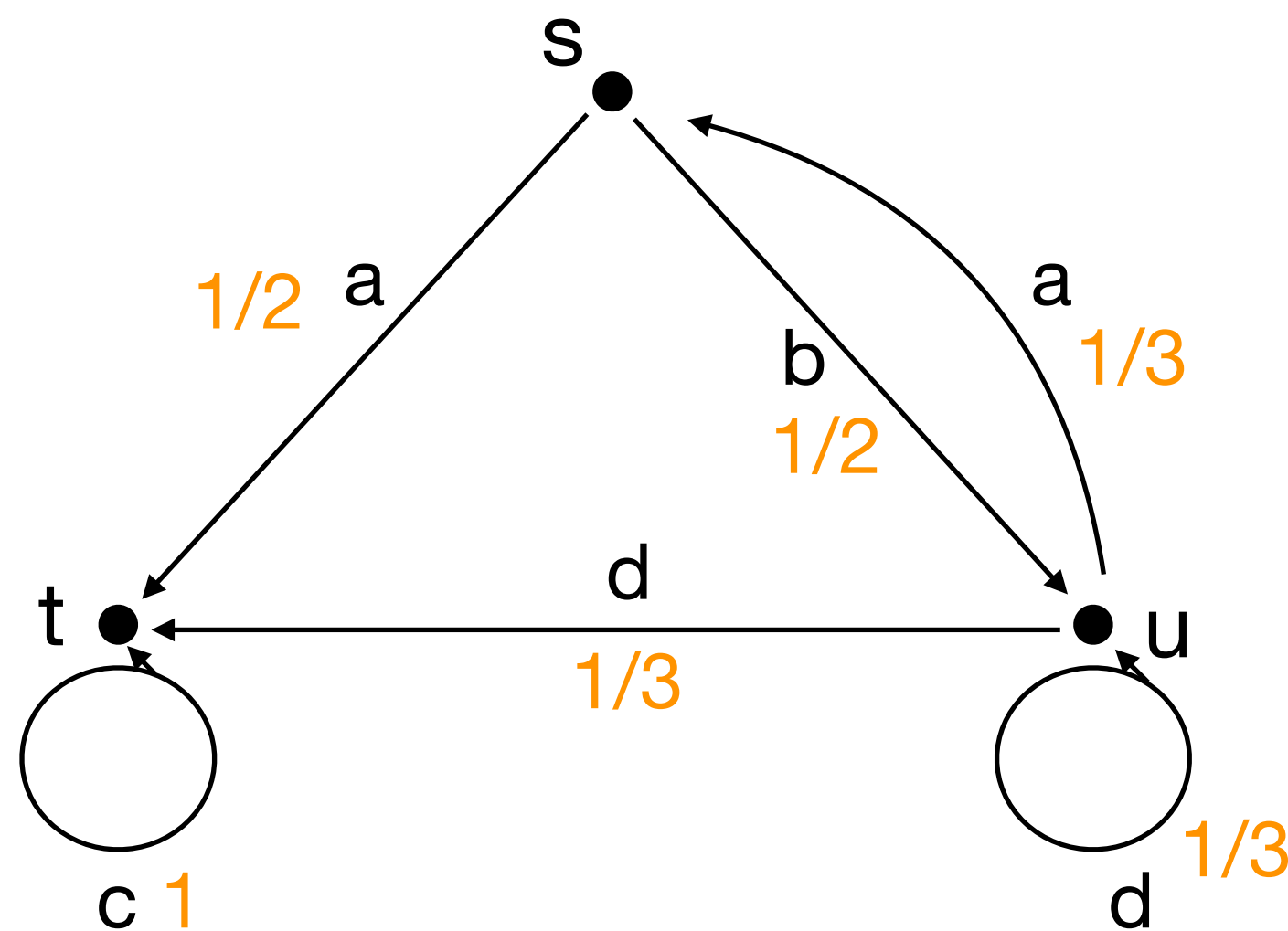
The Role of Algebras

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

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

“Determinization” of

Freely generated by:

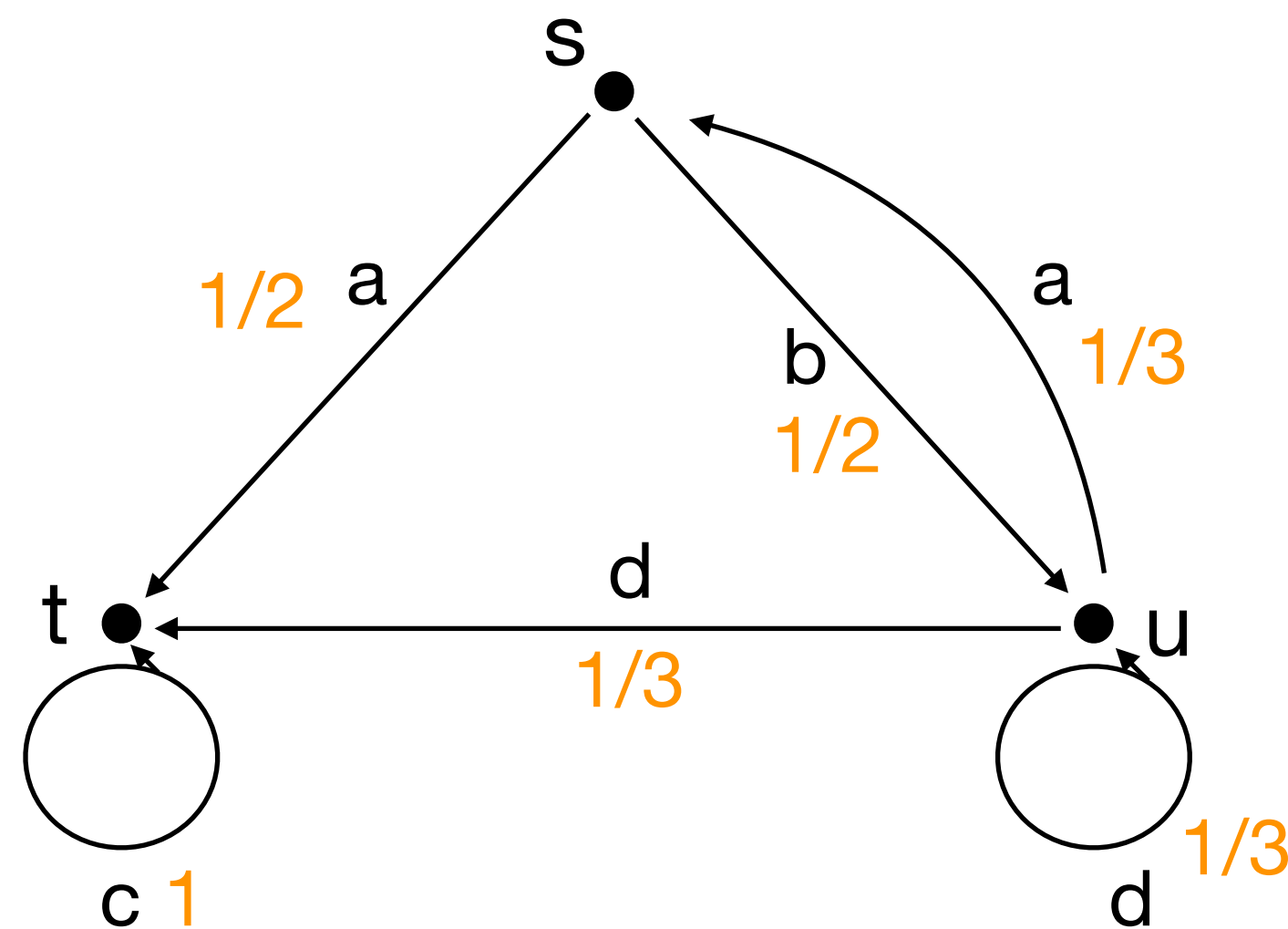


The Role of Algebras

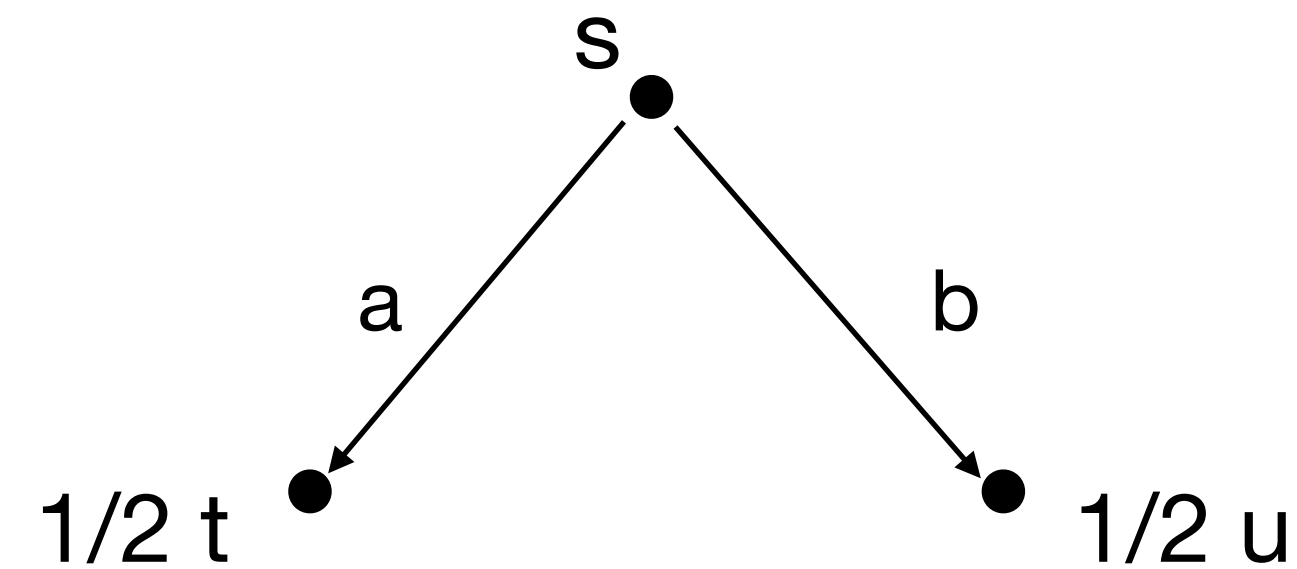
A lot of work happens in categories of algebras, for LMC - convex algebras

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

“Determinization” of



Freely generated by:

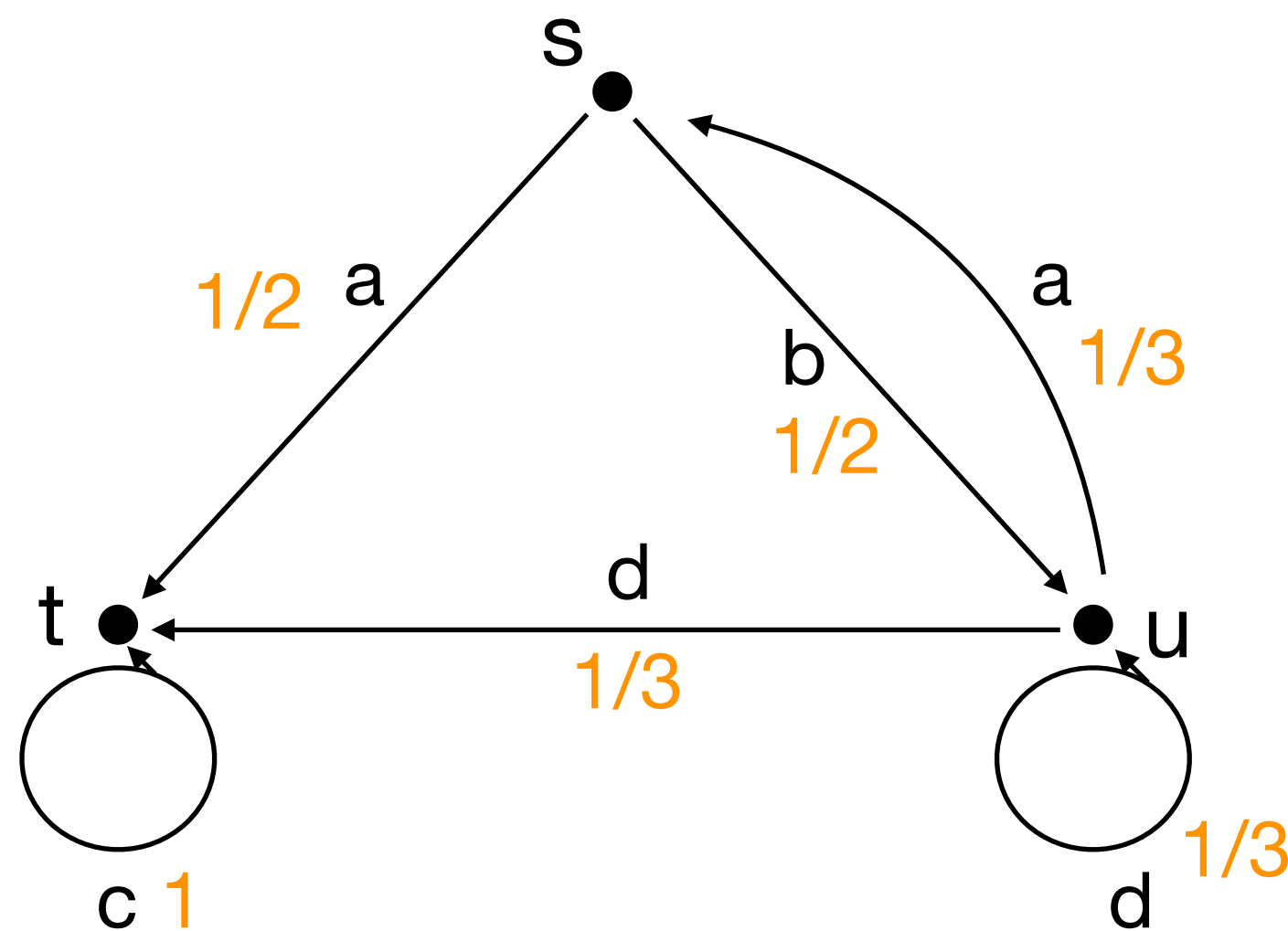


The Role of Algebras

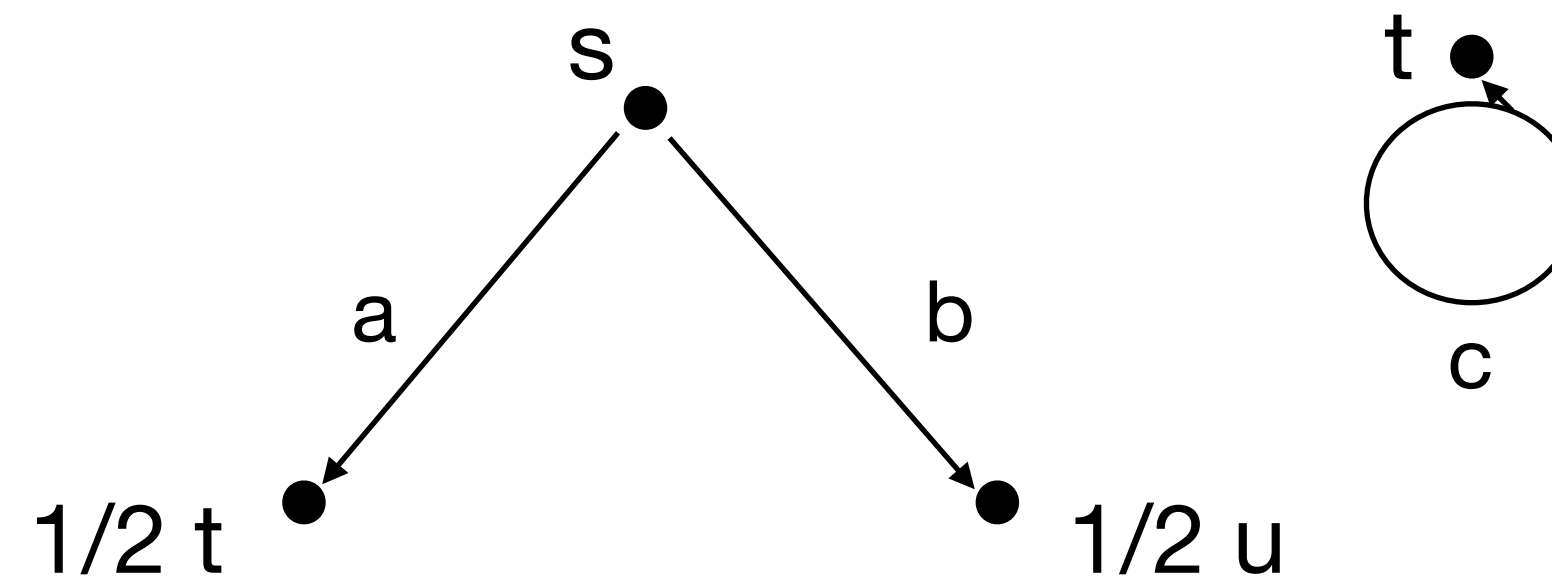
A lot of work happens in categories of algebras, for LMC - convex algebras

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

“Determinization” of



Freely generated by:



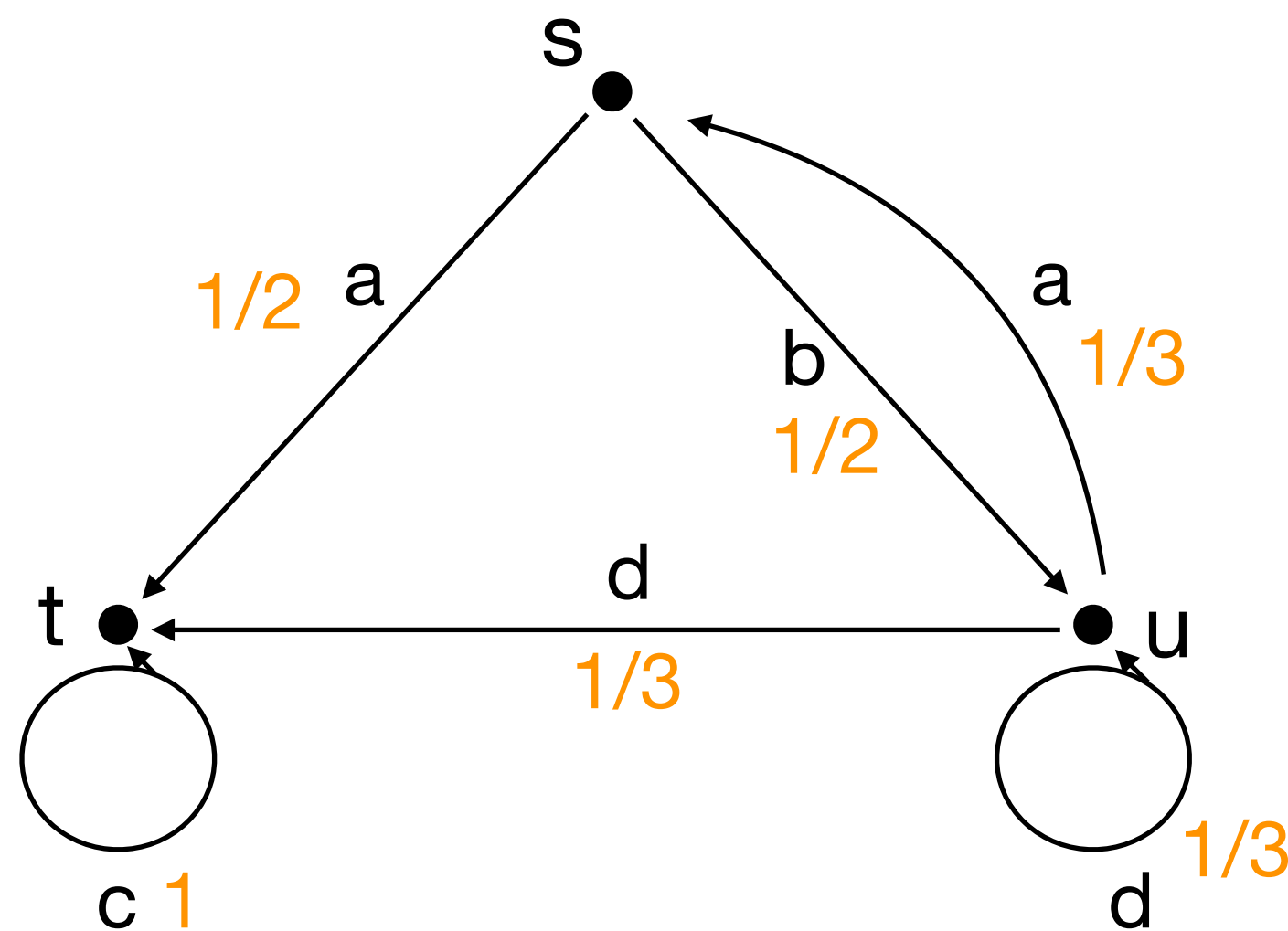
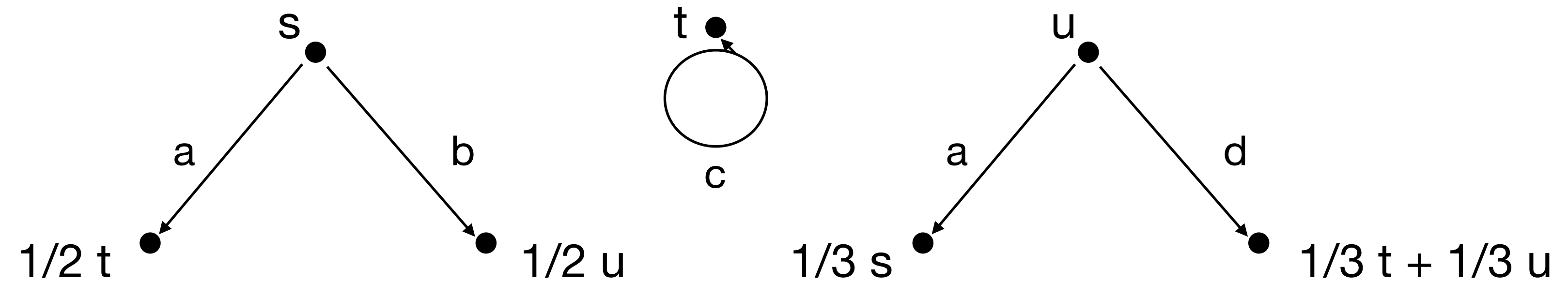
The Role of Algebras

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

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

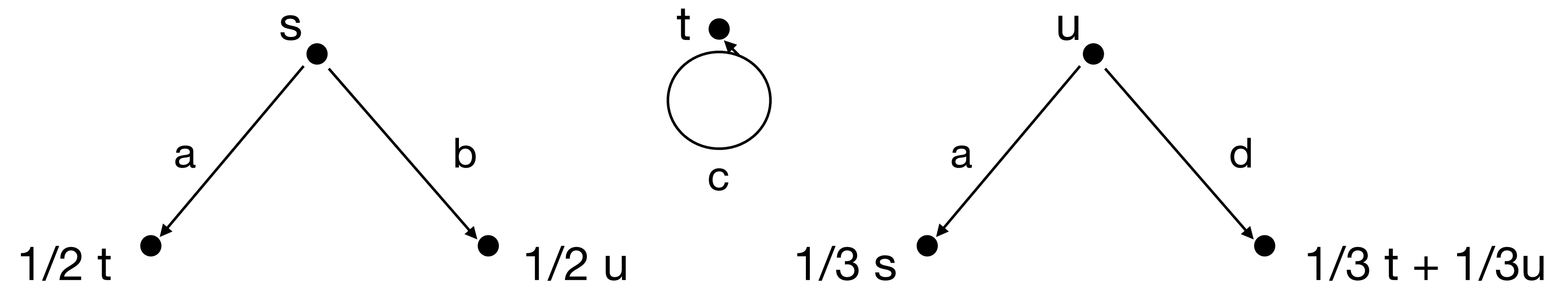
“Determinization” of

Freely generated by:



Determinization

Freely generated by:



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

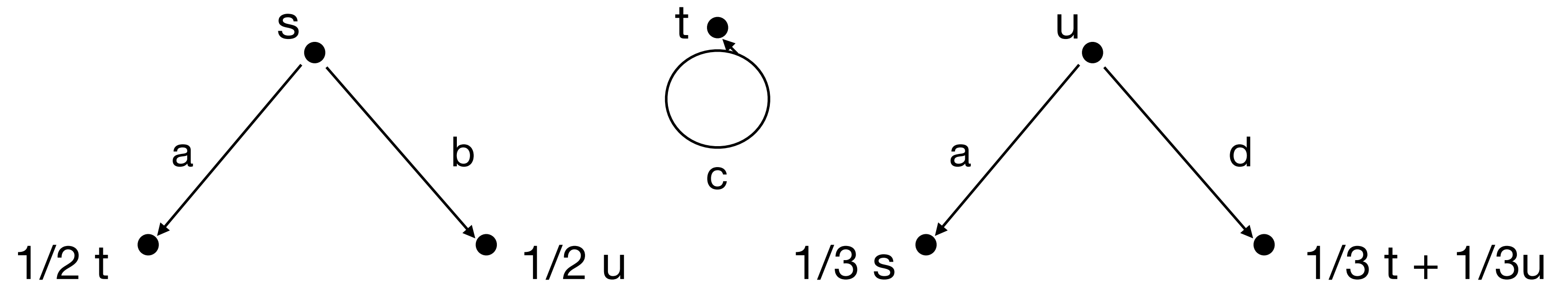
The role of G

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

Freely generated by:

2. $\text{Prob}(A^\omega)$ carries the final G-coalgebra

$\text{Prob}(A^\omega) \rightarrow G \text{Prob}(A^\omega)$



3. Infinite trace equivalence is the final coalgebra semantics

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

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:

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

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:

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

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:

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

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:

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

Proof obligations:

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

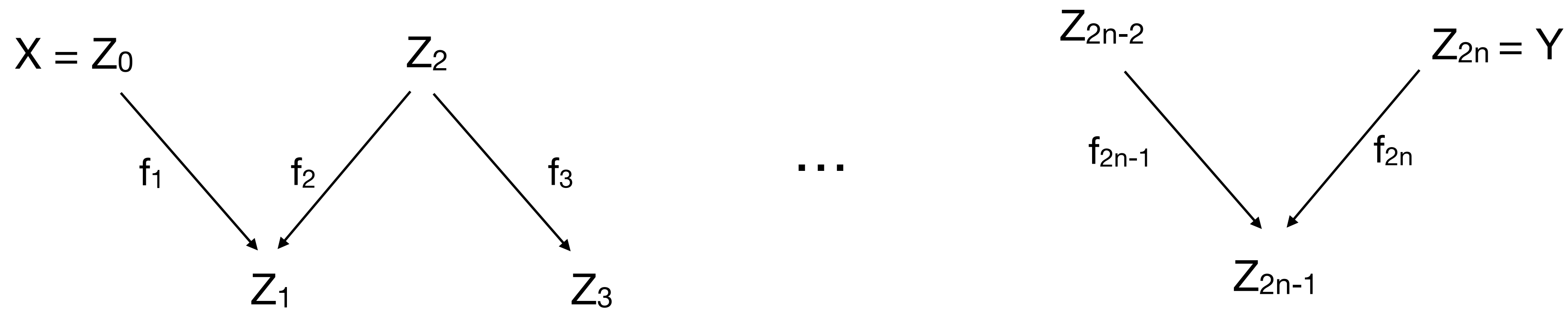
Properness

Properness

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

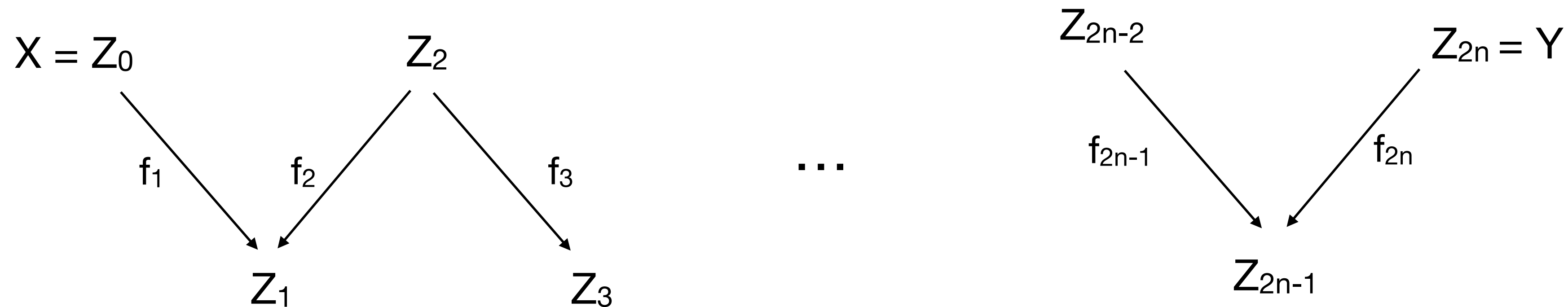
Properness

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



Properness

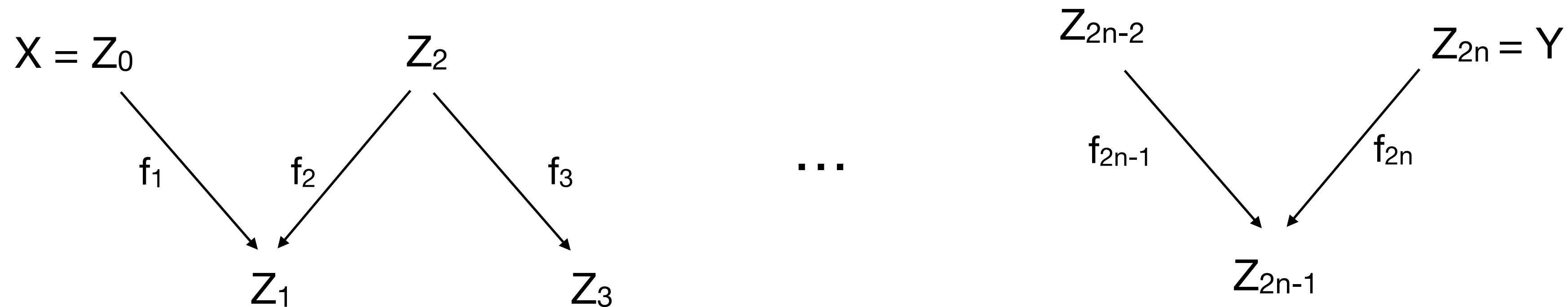
A **zigzag** is a diagram in F-coalgebras over convex algebras ($\mathcal{D}$ - algebras) of the form:



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



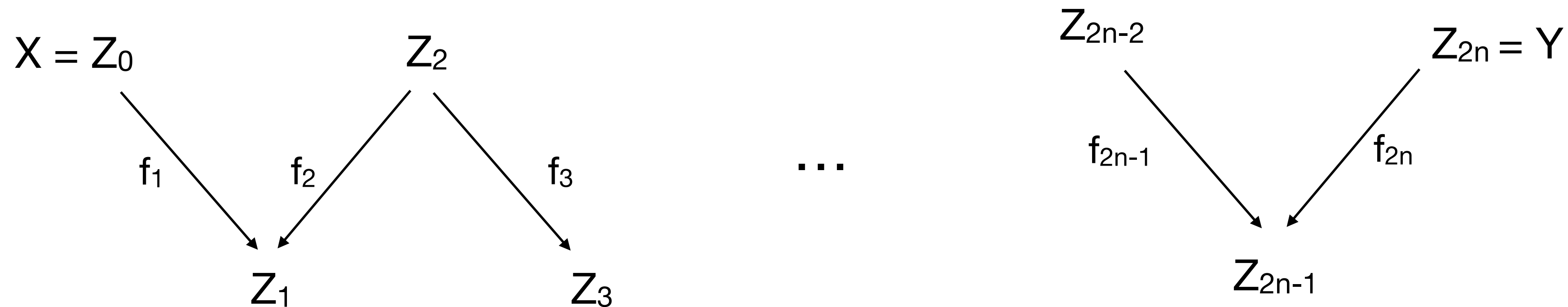
It relates $x \in X$ and $y \in Y$, denoted **$x \sim y$** , if there exist $z_{2k} \in Z_{2k}$, for $k = 1, \dots, n-1$ with $z_0 = x$, $z_{2n} = y$,

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

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.

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 $k = 1, \dots, n-1$ 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 

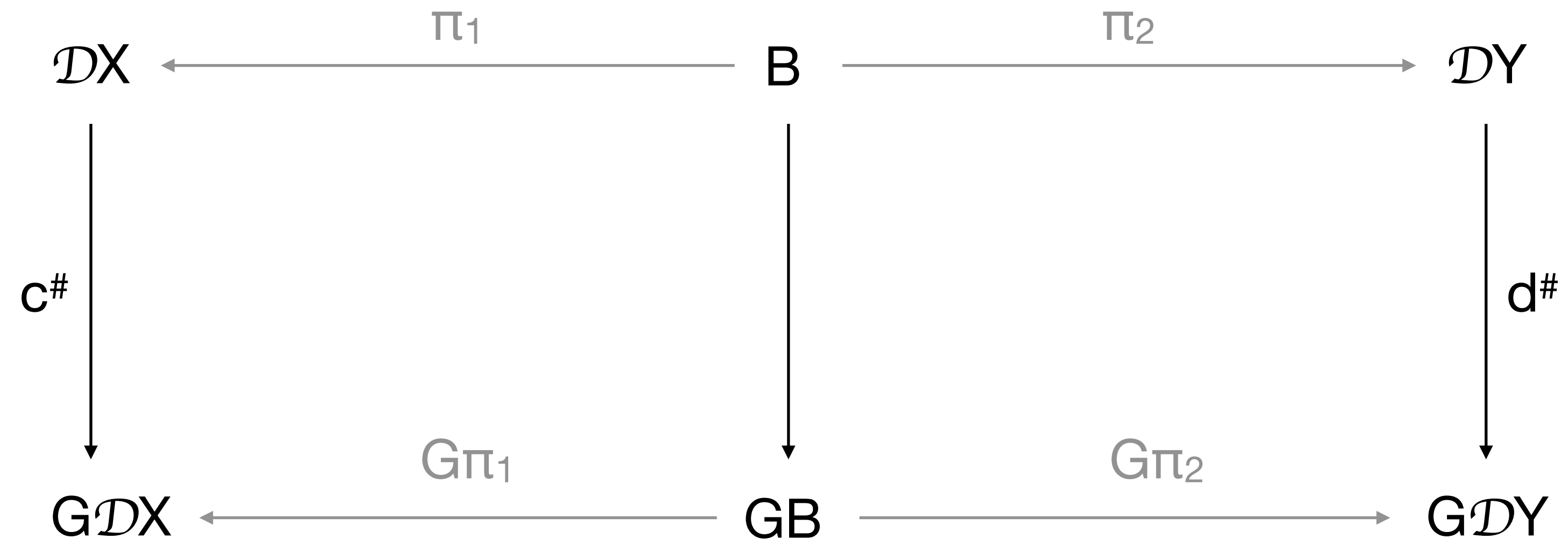
Milius '13/'17 If proper functor and , then “completeness”

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

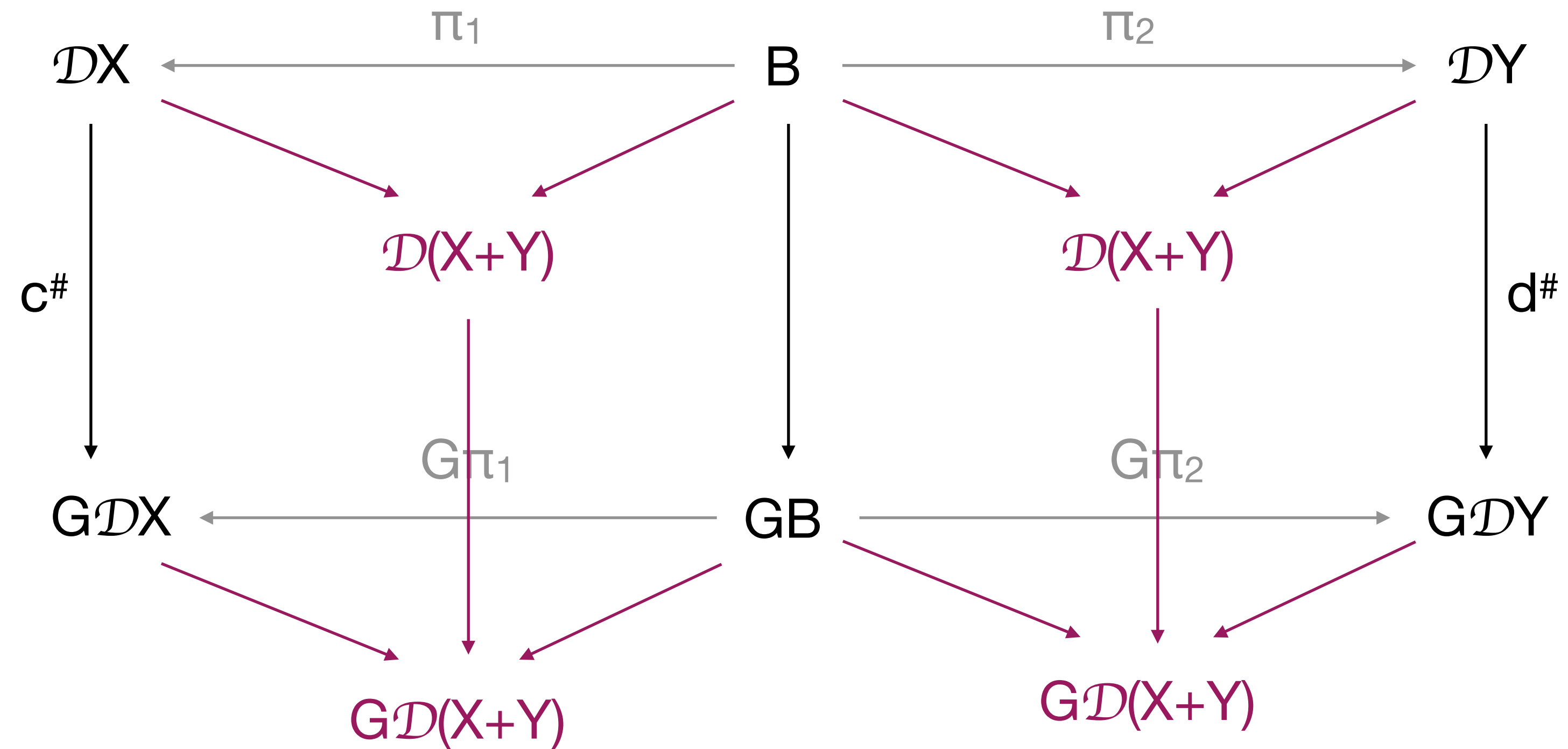
Now, for infinite traces: Remarkably beautiful and easy proof of properness for G

Bisimilarity gives the zigzag

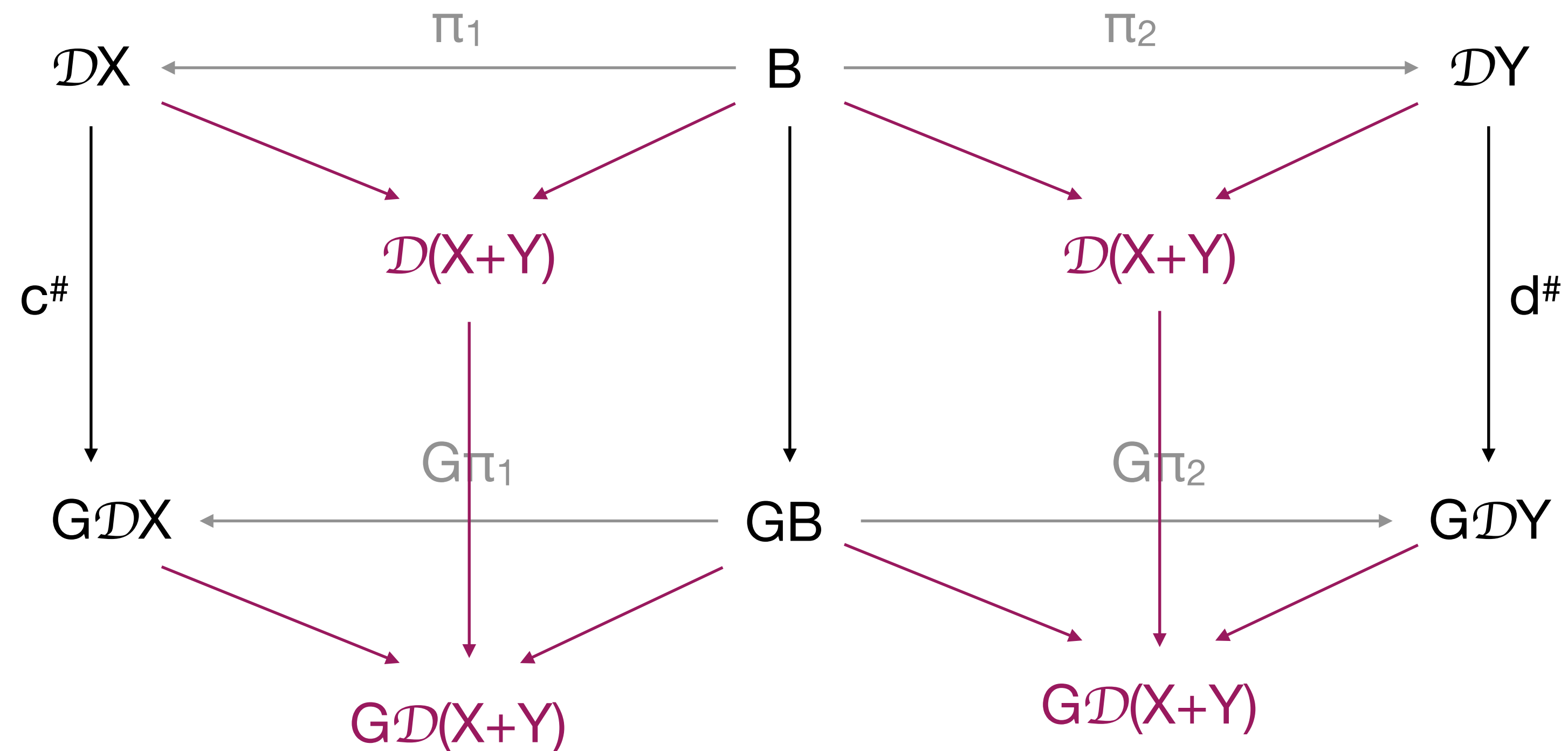
Bisimilarity Gives the Zigzag



Bisimilarity Gives the Zigzag

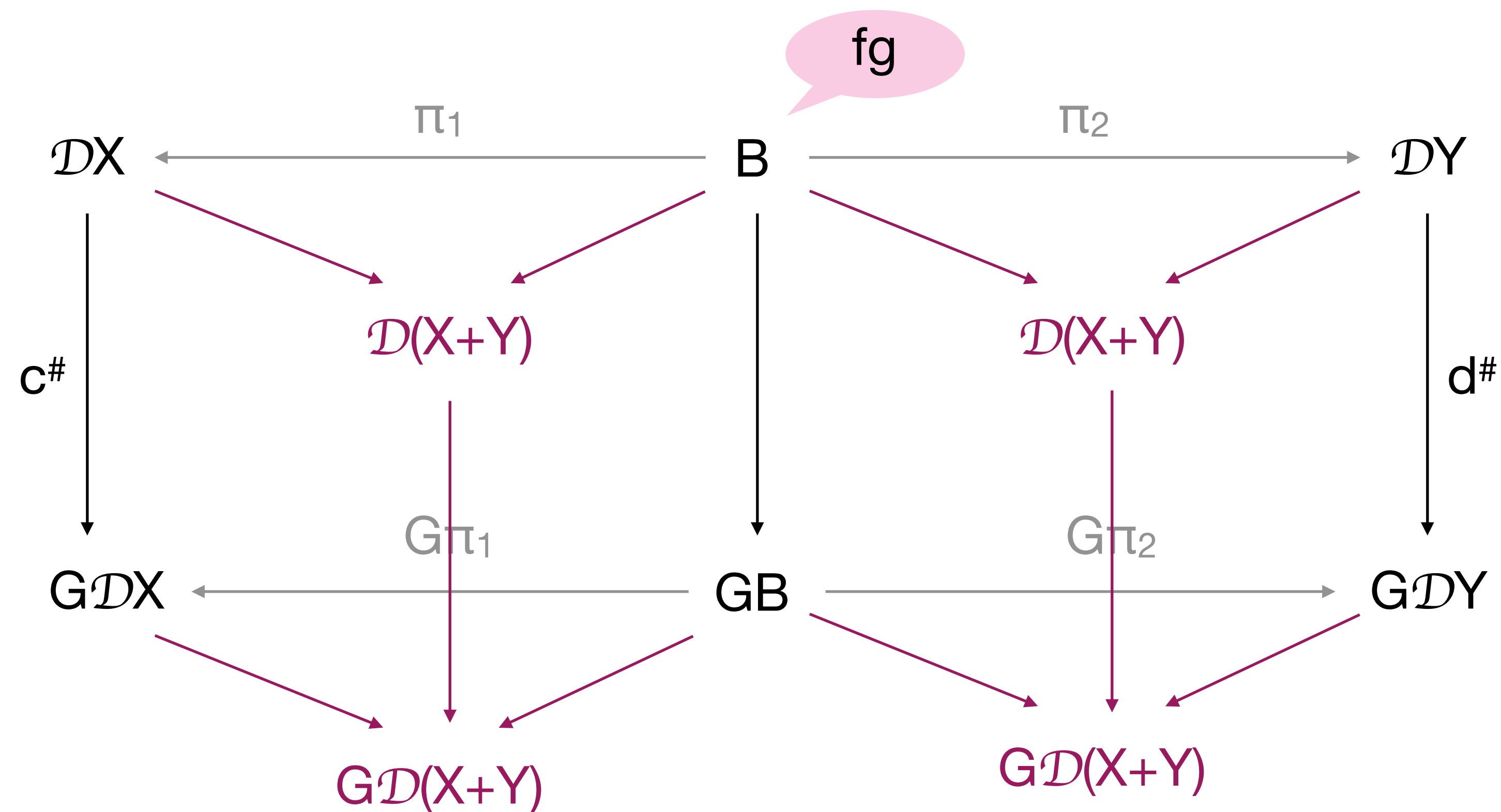


Bisimilarity Gives the Zigzag



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

Radboud University



Larry Moss



Tori Noquez



Todd Schmidt



Alexandra Silva



Valeria Vignudelli



Erik de Vink



Harald Woracek



Thank You !

Strong foundations enhance developments in formal methods.

Coalgebraic generalisations may give strong foundations.

Two recent cases in favour of bisimilarity.

