Bisimulation and Precongruence

David Sprunger

October 23, 2017

Perspective

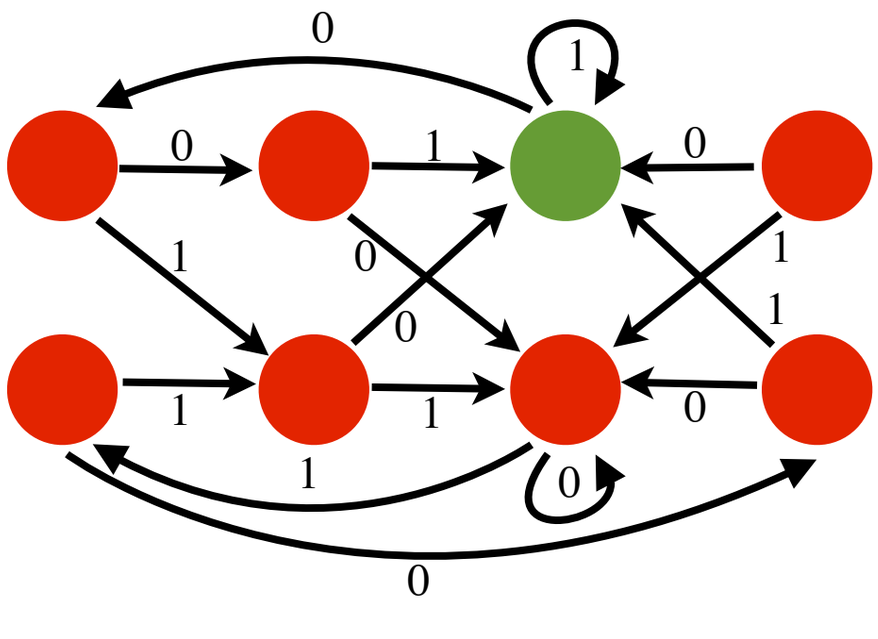
- Avoid category theory
- Automata-level bisimulation
- Weak pullback preservation & bisimulation
- Automata-level precongruence

Plan

1. Bisimulation for deterministic systems
2. Bisimulation for nondeterministic systems
3. Precongruences (for both)

Deterministic bisimulation

Deterministic Finite Automata

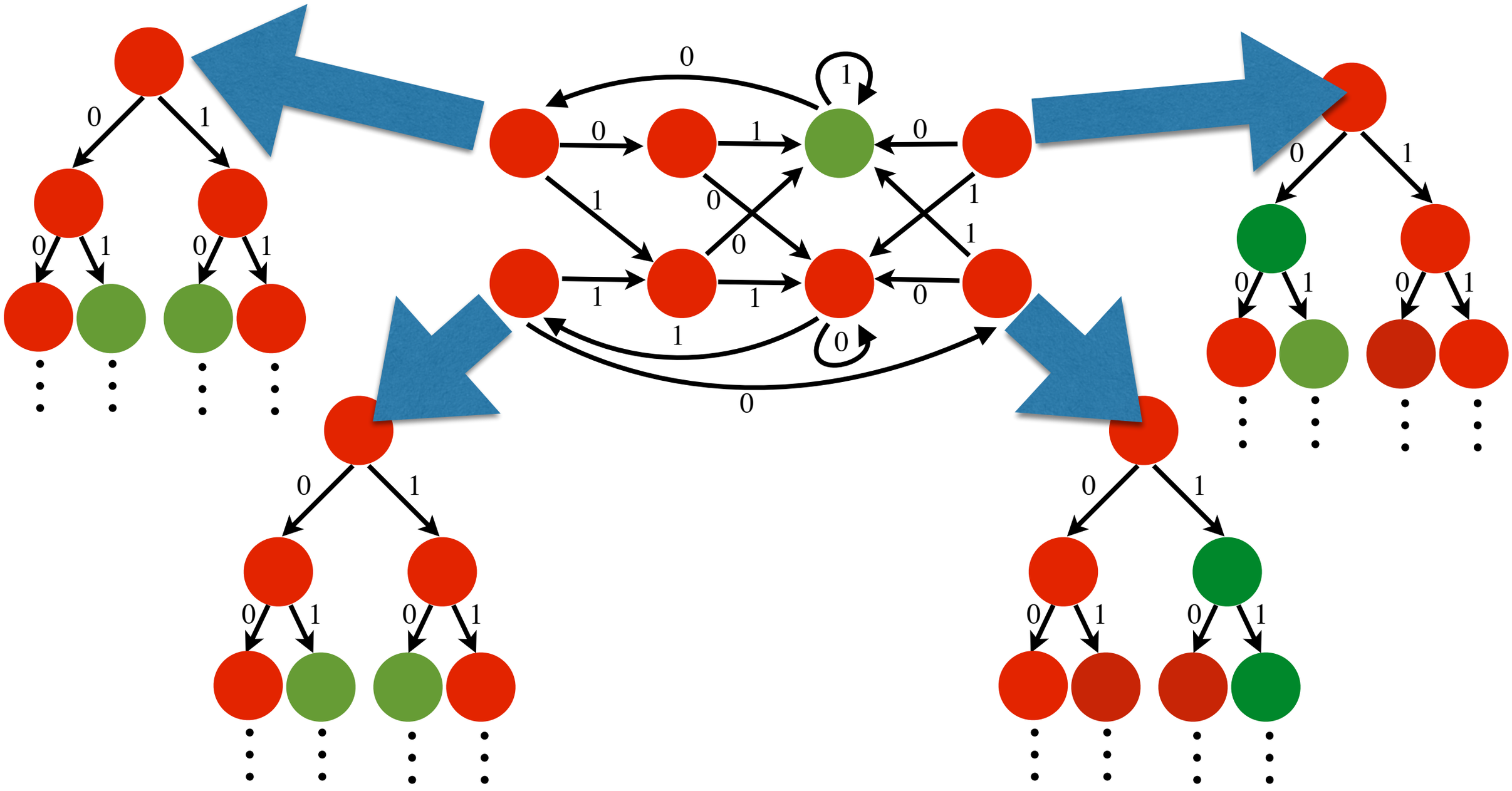


- Each state has a type
- Finitely many transition types
- Exactly one transition of each type leaves each state

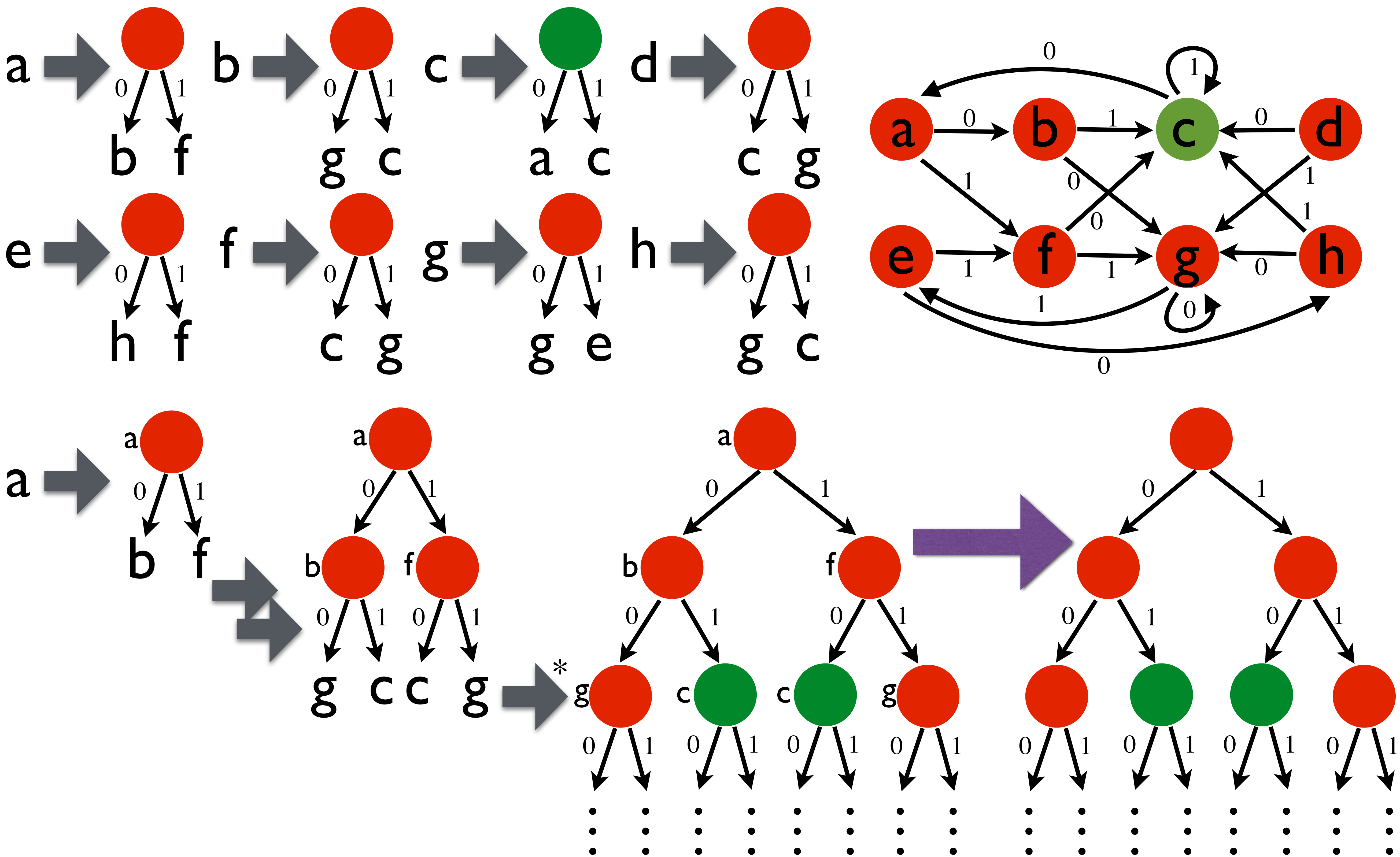
Def'n: the <u>output</u> of a state s on a word w is the state type we reach from s following transitions in w

Def'n: the <u>behavior</u> of a state is the map from words to outputs starting from s

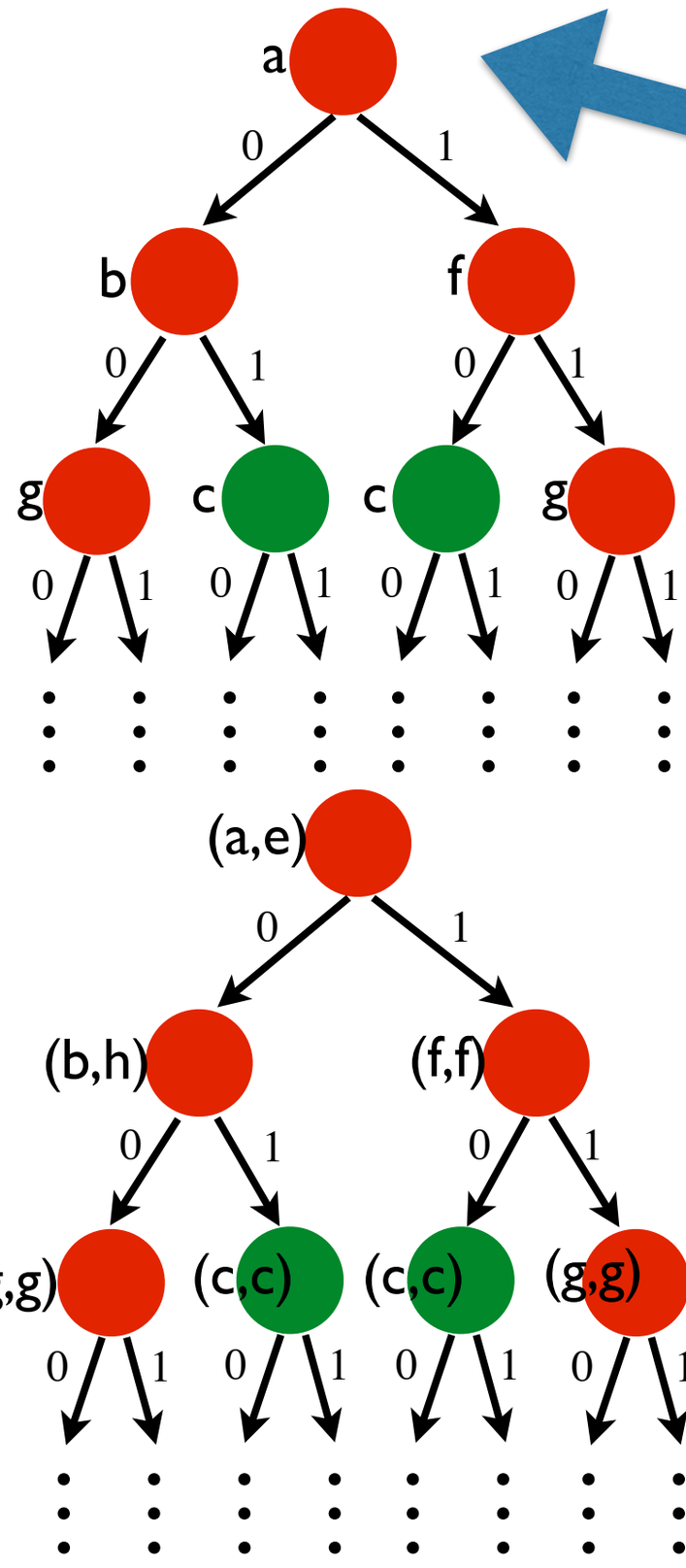
Tree unfoldings are behavior

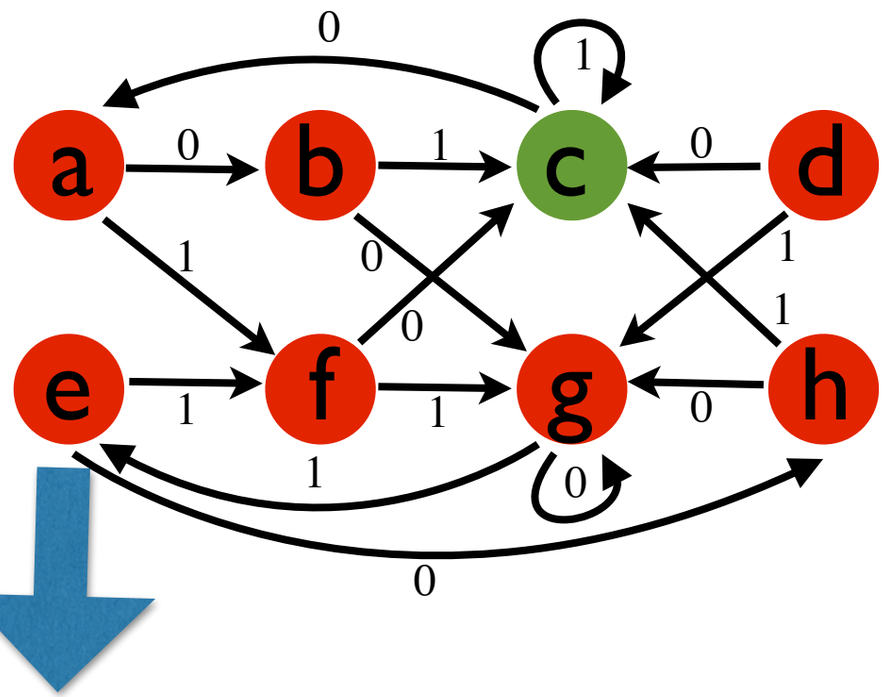


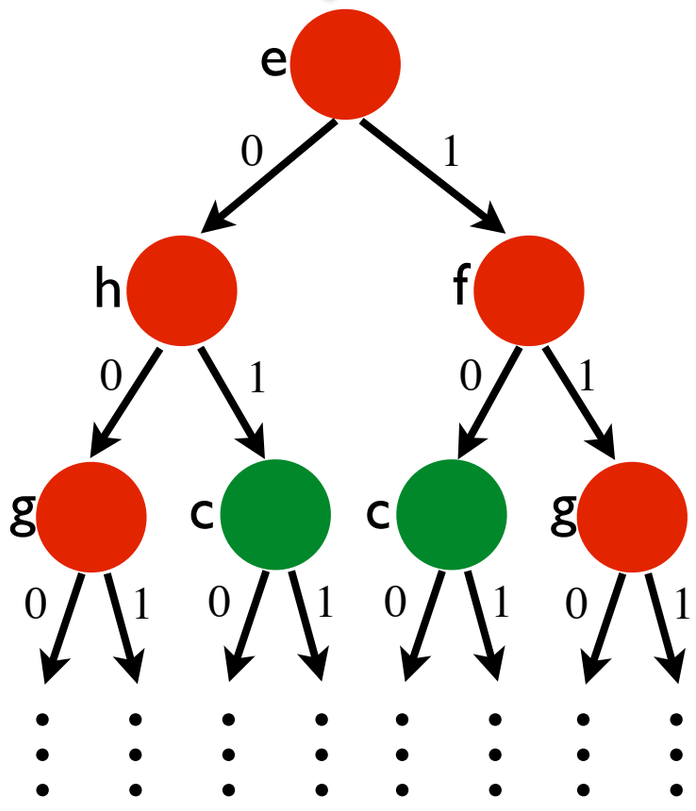
Annotated tree unfolding



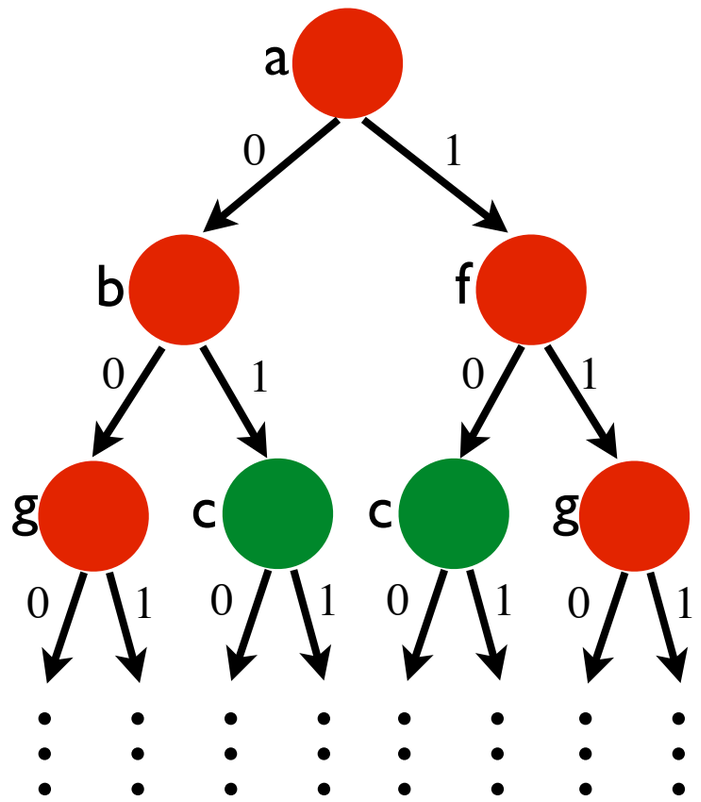
Detecting equal unfoldings





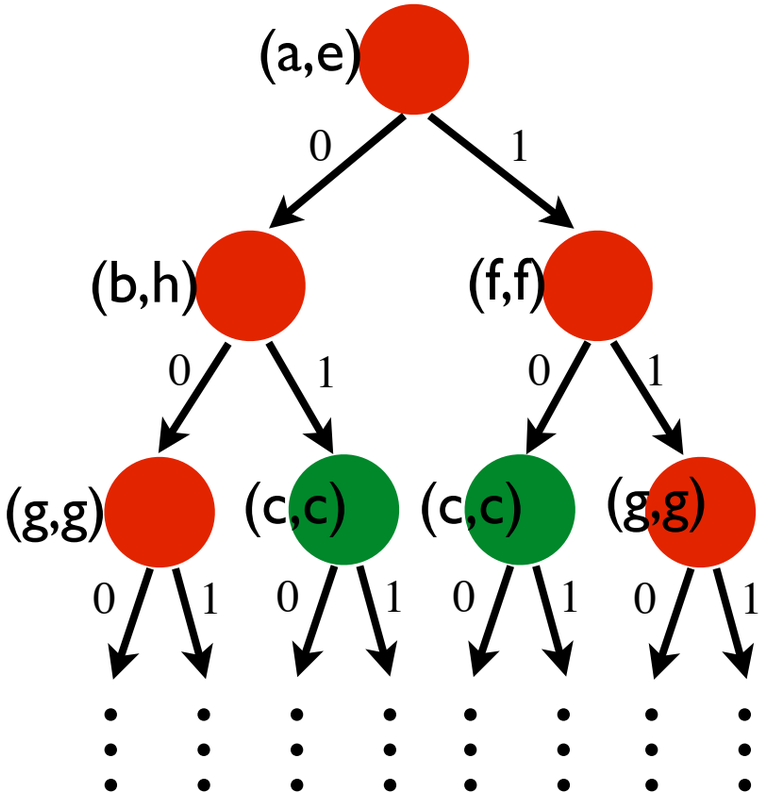


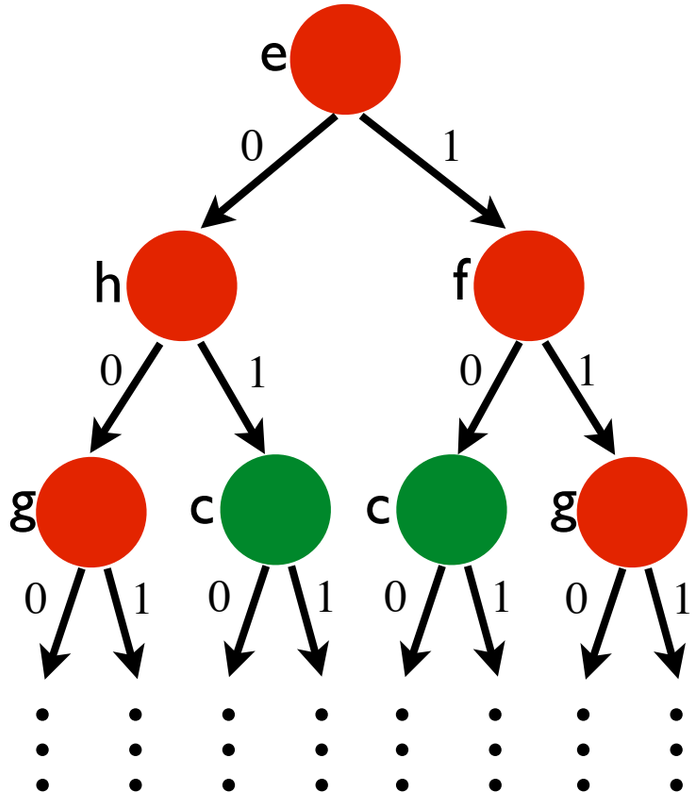
Detecting equal unfoldings

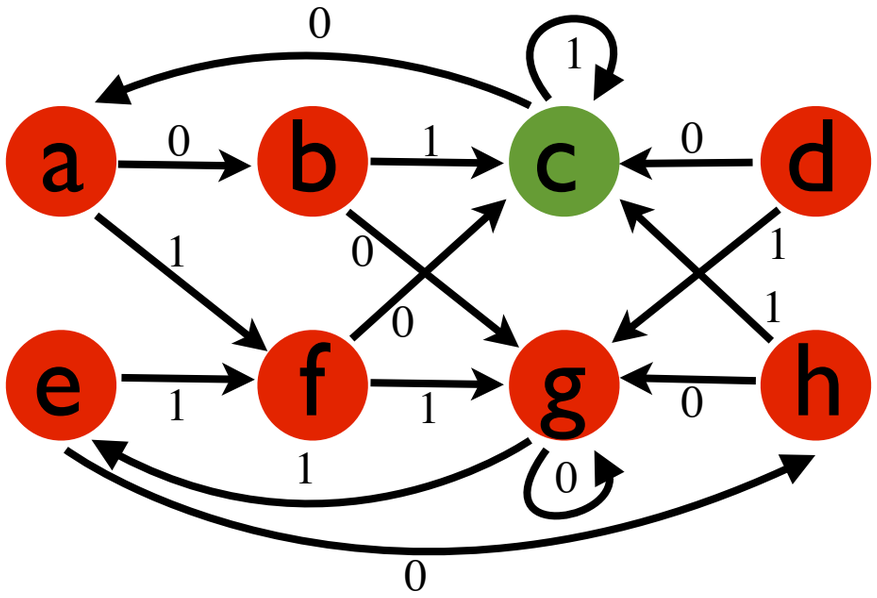


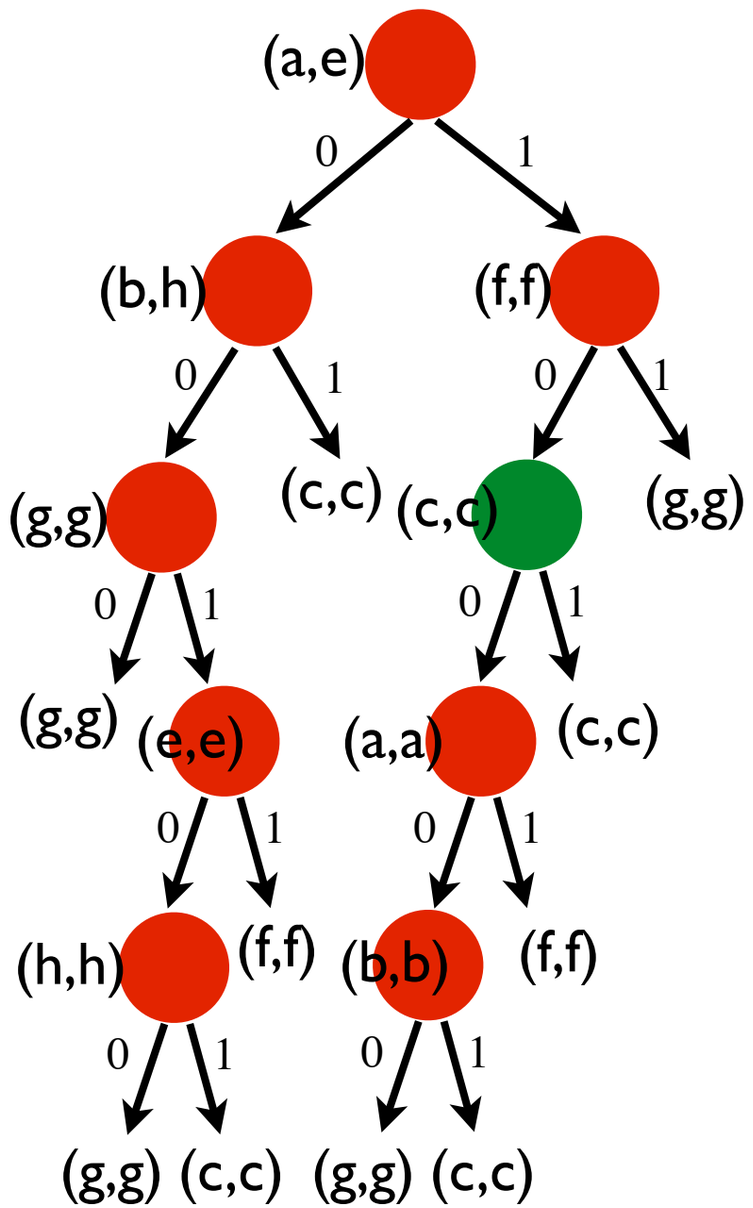
Good properties:

1. The type of (r, s) is the common type of r and s.
2. If $\delta(r, \lambda) = r'$ and $\delta(s, \lambda) = s'$, then $\delta((r, s), \lambda) = (r', s')$









Bisimulation

That this tree satisfies…

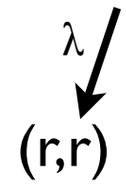
1. The type of (r, s) is the common type of r and s,

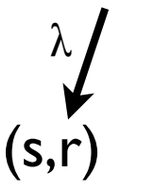
2. If $\delta(r, \lambda) = r'$ and $\delta(s, \lambda) = s'$, then $\delta((r, s), \lambda) = (r', s')$,

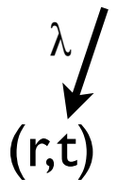
3. All unexpanded pairs are expanded elsewhere

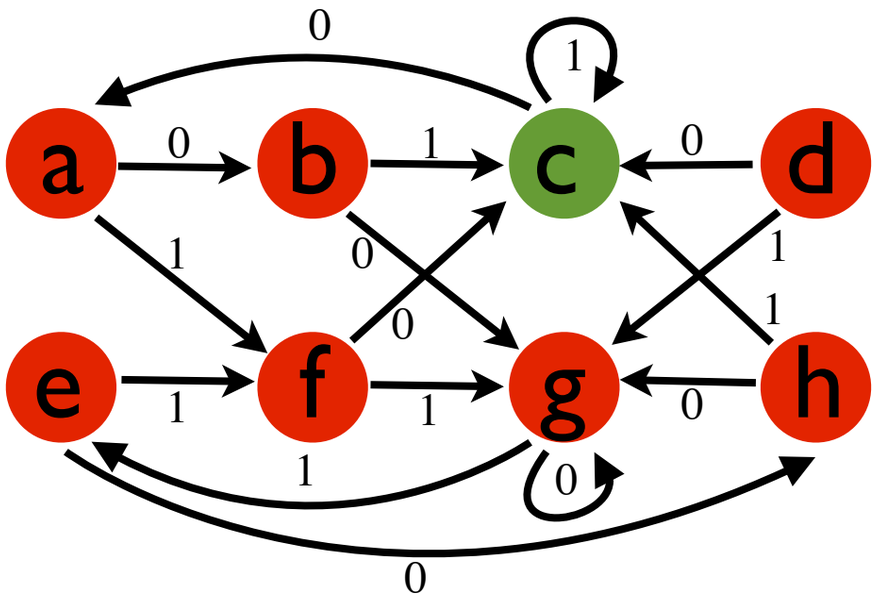
…is enough to guarantee the infinite tree matches.

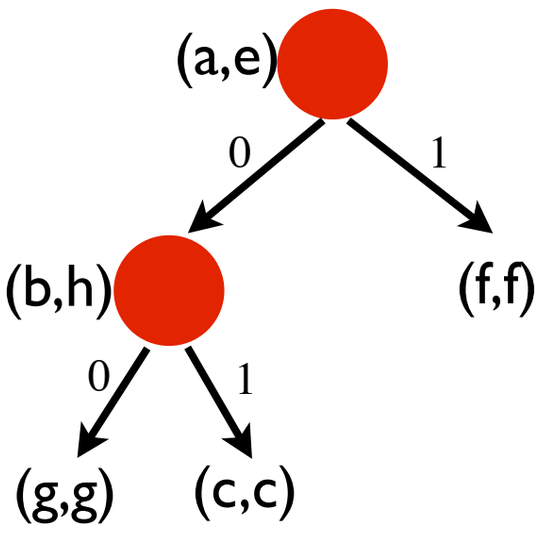
Shortcut rules II

R) If we discover a pair like  we don't need to expand (r, r).

S) If we discover a pair like  and we already expanded (r, s), we don't need to expand (s, r).

T) If we discover a pair like  and there is an s such that we already expanded (r, s) and (s, t), we don't need to expand (r, t).





Bisimulation up-to

That this tree satisfies…

1. The type of (r, s) is the common type of r and s,

2. If $\delta(r, \lambda) = r'$ and $\delta(s, \lambda) = s'$, then $\delta((r, s), \lambda) = (r', s')$,

3. All unexpanded pairs are *in the equivalence closure of those* expanded elsewhere

…is enough to guarantee the infinite tree matches.

Easy Generalization

We need:

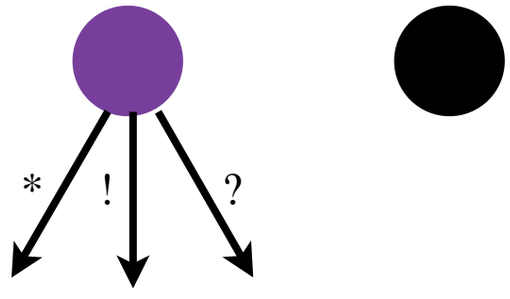
- All states of a type have the same transitions
- There are finitely many transitions for each state type
- Each transition gives a unique next state

We don’t need:

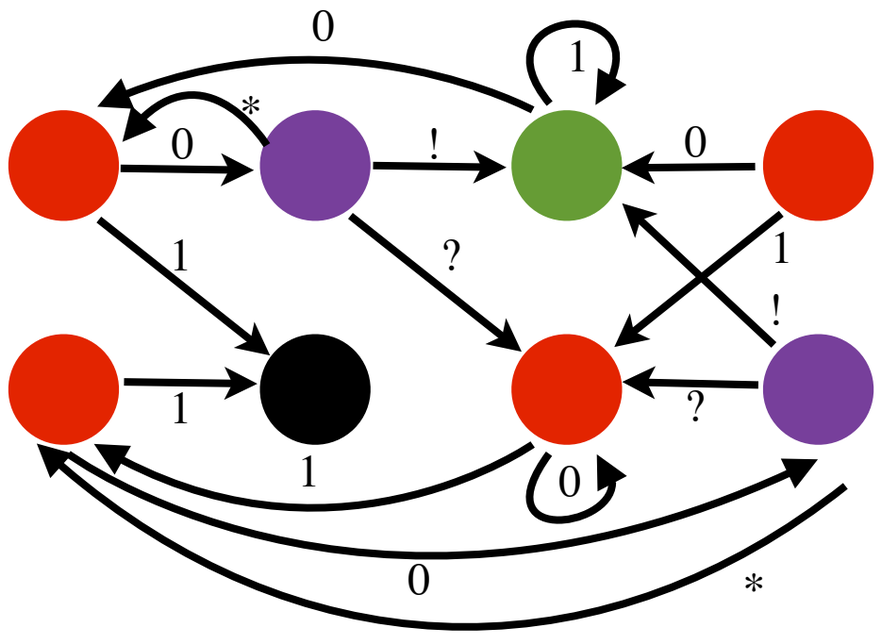
- All states have the same transitions

Easy Generalization

We could add any number of new types



And still reason about



Hard Generalization

We need:

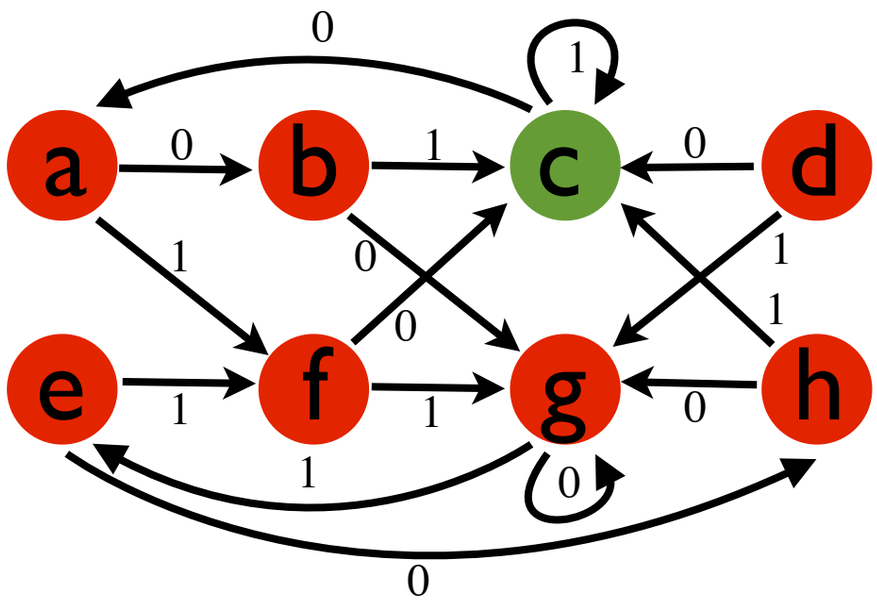
- All states of a type have the same transitions
- There are finitely many transitions for each state type
- Each transition gives a unique next state

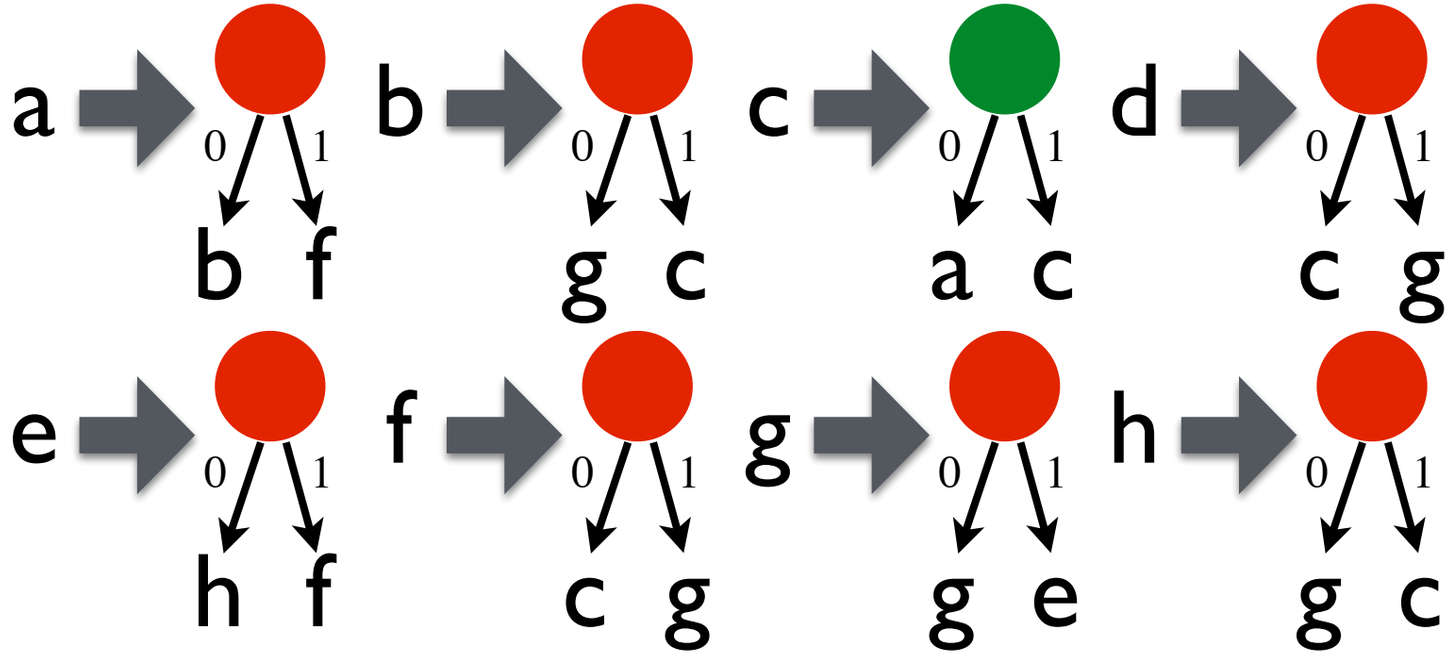
Removing the last criteria is possible, but delicate.

Interlude: categorical formulation

Transition structures are captured by a functor F:

- Given states X, transitions returning to X are FX
- Given f: X→Y, get substitution Ff: FX→FY



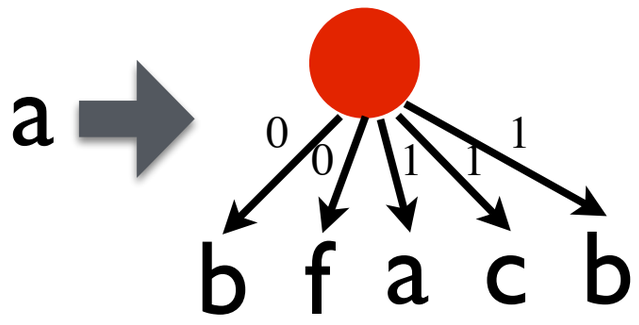


A transition system is just τ: X→FX

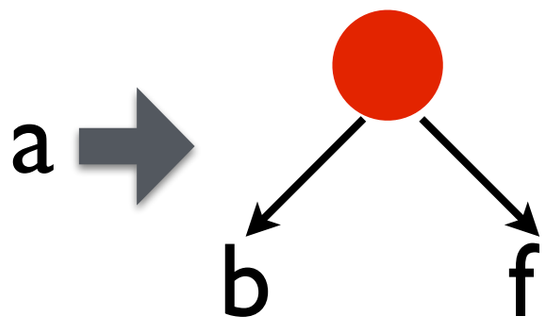
Nondeterministic bisimulation

Nondeterminstic automata

We want to reason about states with transitions like

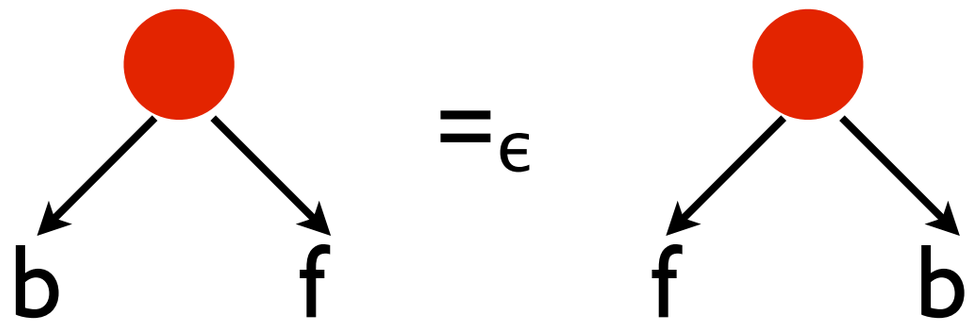


(We simplify by only considering one type)

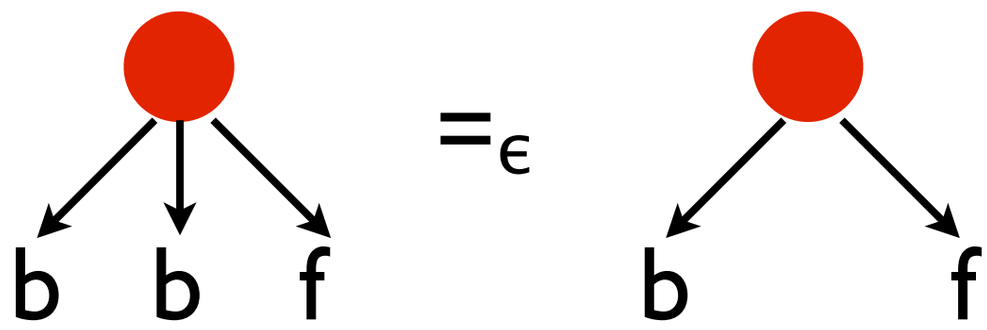


Axioms of nondeterminacy

Order doesn't matter, so these should be the same:



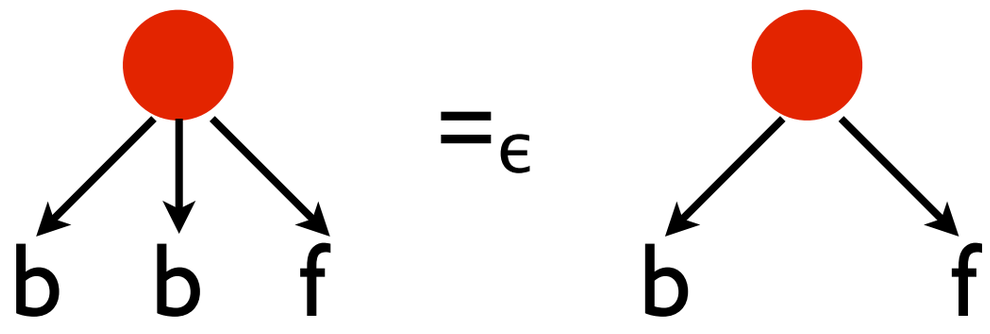
Repeats don't matter, so these should be the same:



Recall one of our critical ingredients was

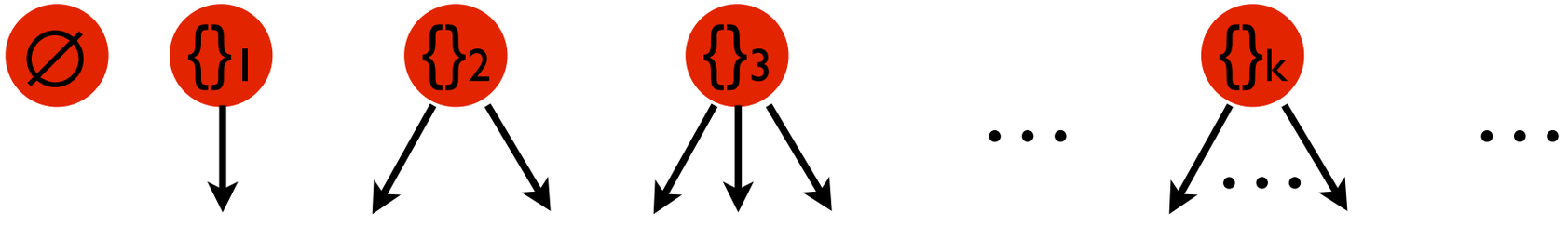
- All states of a type have the same transitions

How will we make matching transitions?

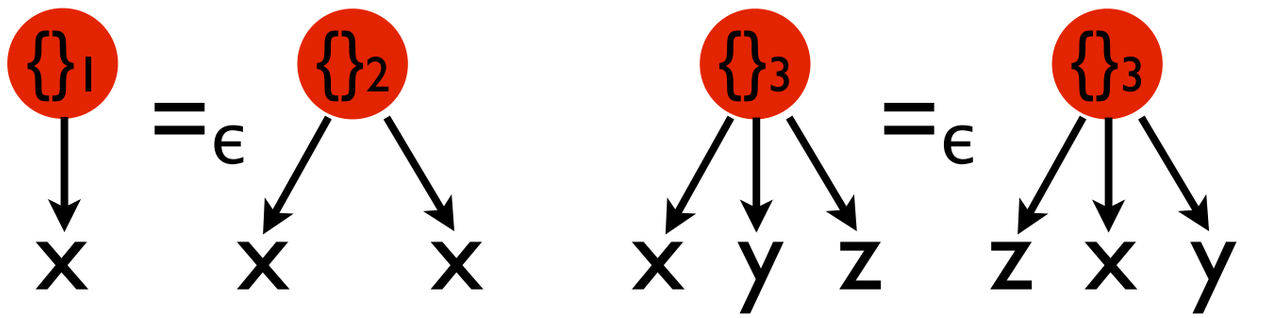


Adding new state types

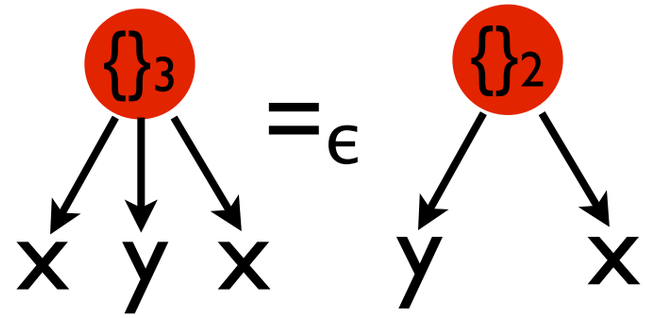
We subdivide the red type into ω-many subtypes…



require some axioms…

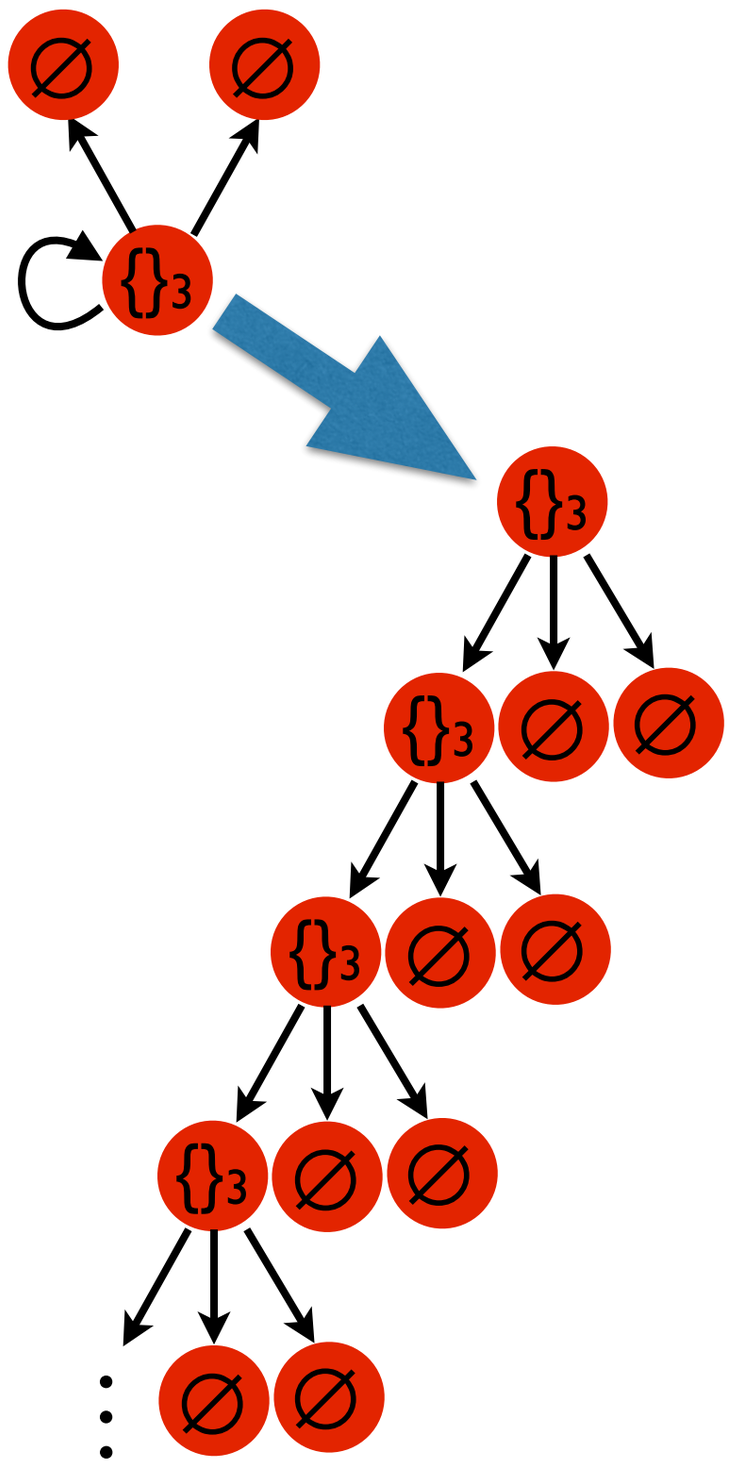


close under e&s…

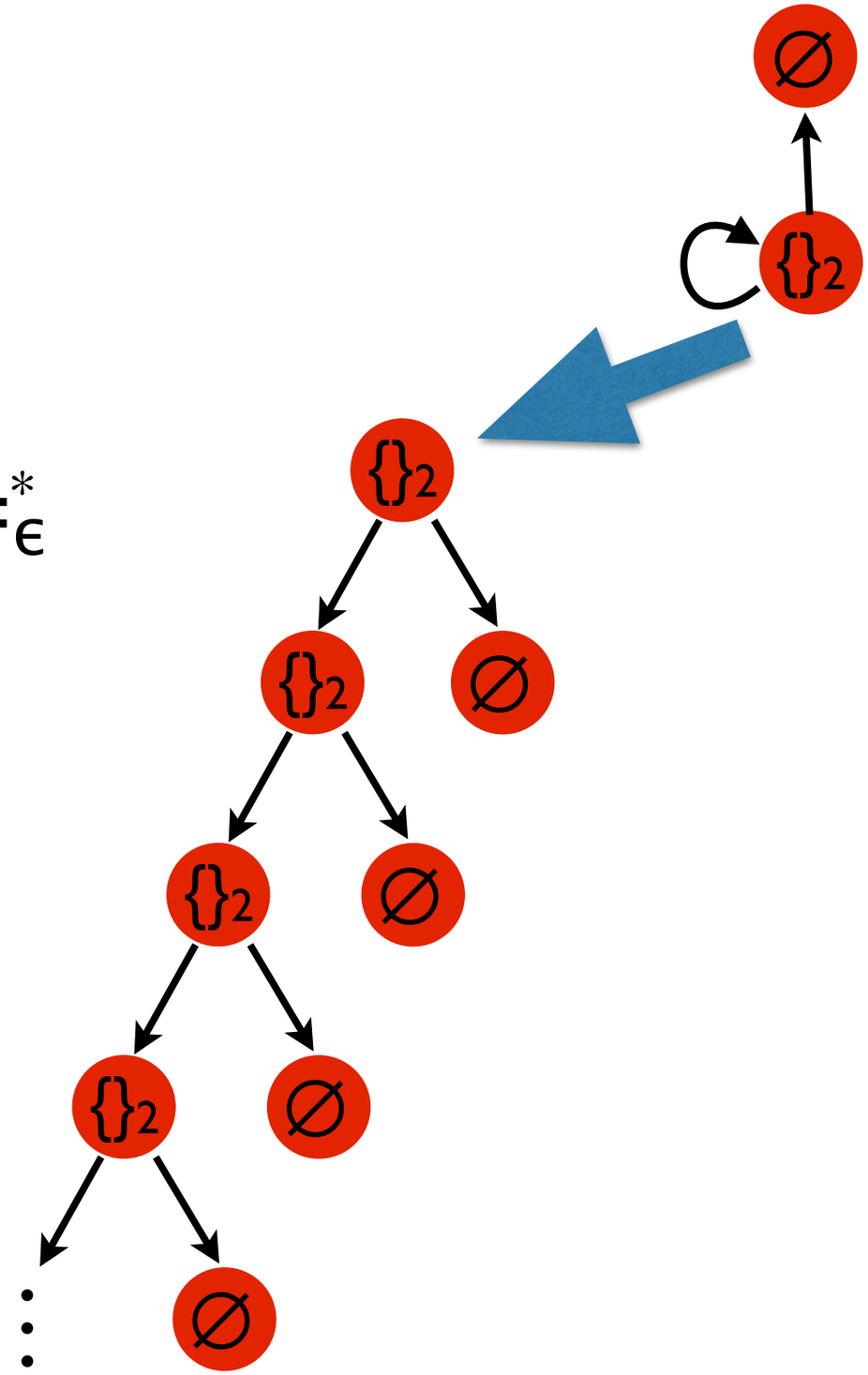


and relax our notion of equal tree unfolding.

Kind-of-equal tree unfolding



$=^{*}_{\epsilon}$



Unfolding semantics

"The" tree unfolding of a state is the equivalence class of trees obtainable by applying the axioms to the state's usual tree unfolding.

It suffices to prove that two states have **one** common unfolding to show they have the same set of possible unfoldings.

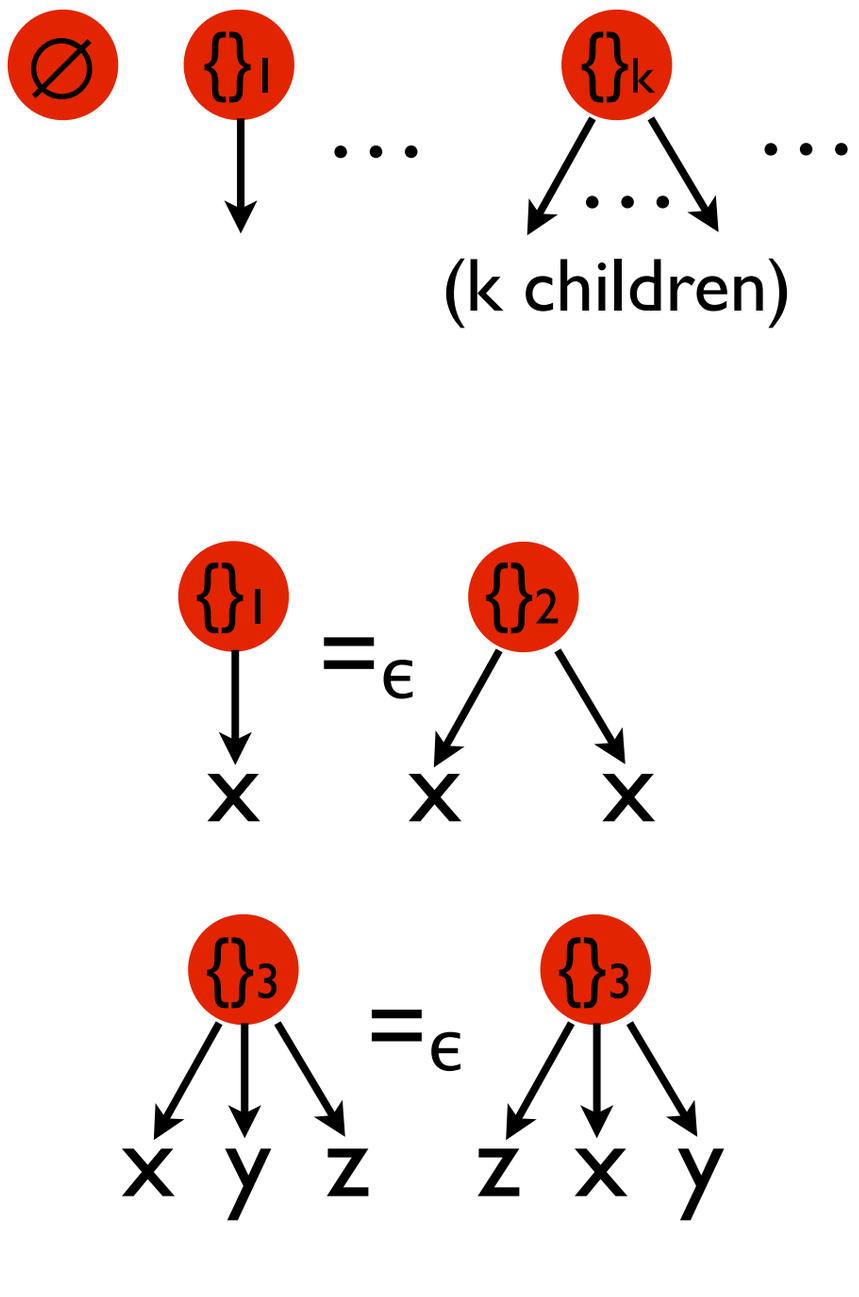
Seems like a lot of work…

Say a system type is *finitely-coverable* if every transition structure on a set X (potentially infinite) can be found as the image of transition structures on a Y, some finite subset of X, under the inclusion.

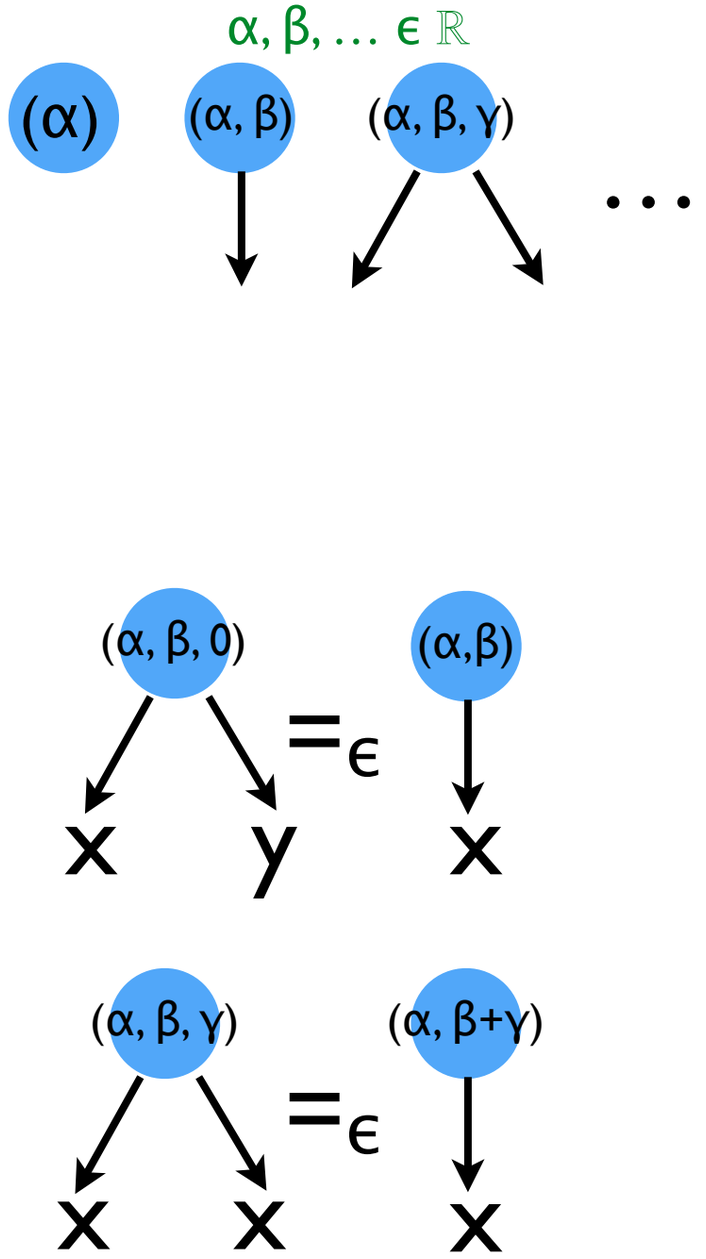
Theorem (Adámek, Porst, Gumm, Trnková):
A system type is finitely-coverable if and only if it has a presentation.

Examples of presentations

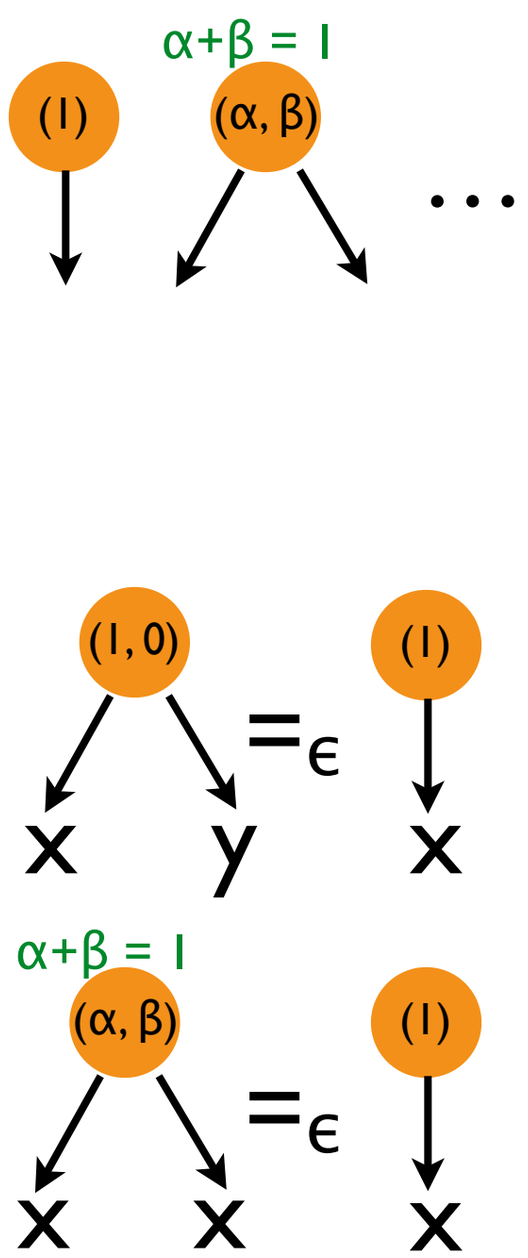
Finite powerset



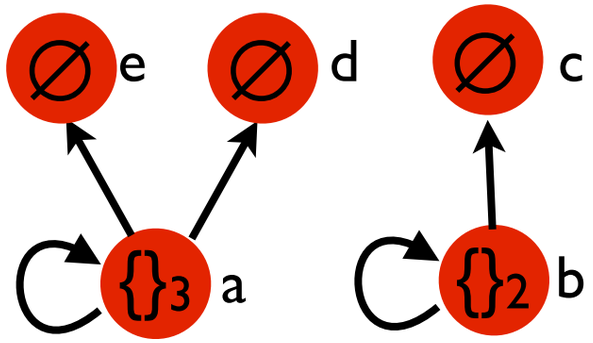
Weighted automata

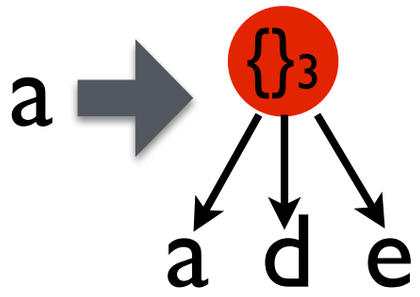


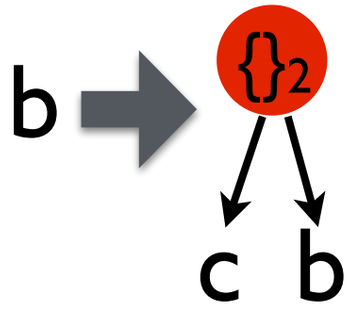
Markov chains



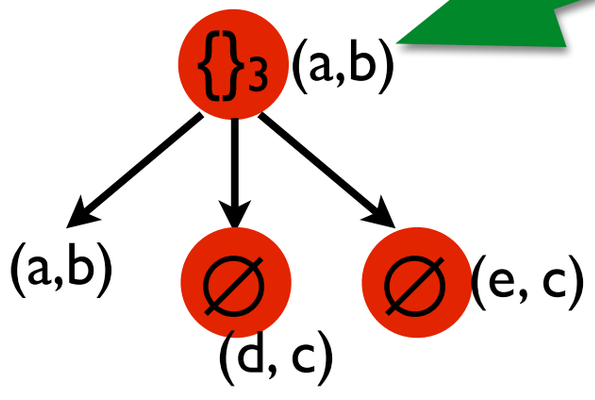
Bisimulation for powerset



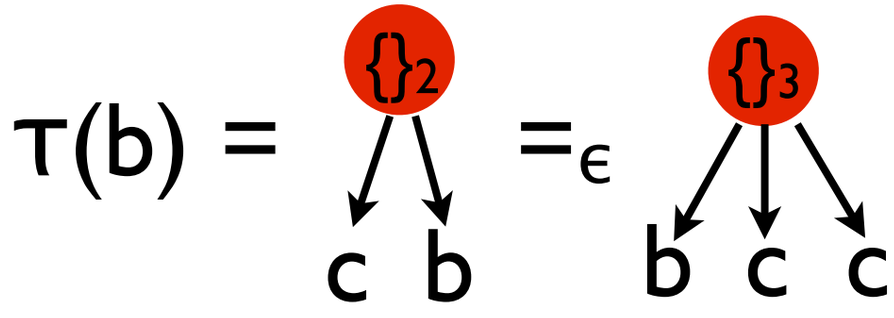




This witnesses a common unfolding:



Ok, because



Presentation bisimulations

Data: a transition presentation scheme (state types and axioms), a transition system in that scheme

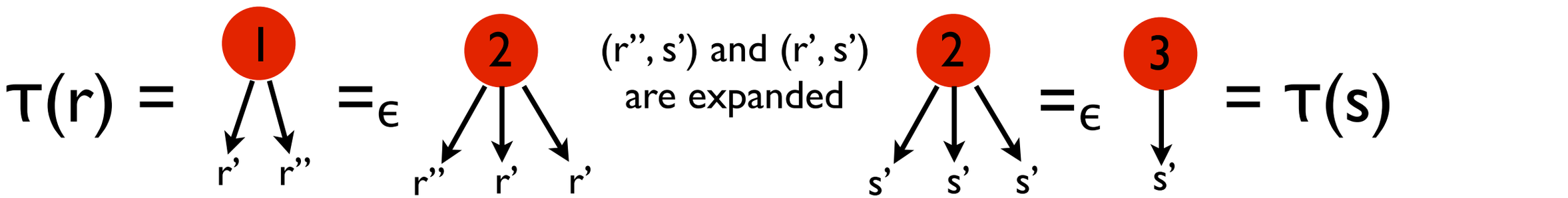
Algorithm: A normal bisimulation, but to expand a pair (r, s), pick representatives of $[\tau(r)]_{=\epsilon}$ and $[\tau(s)]_{=\epsilon}$ with common type.

Proposition: If (r, s) is in a presentation bisimulation, then r and s have the same tree

Up-to techniques II

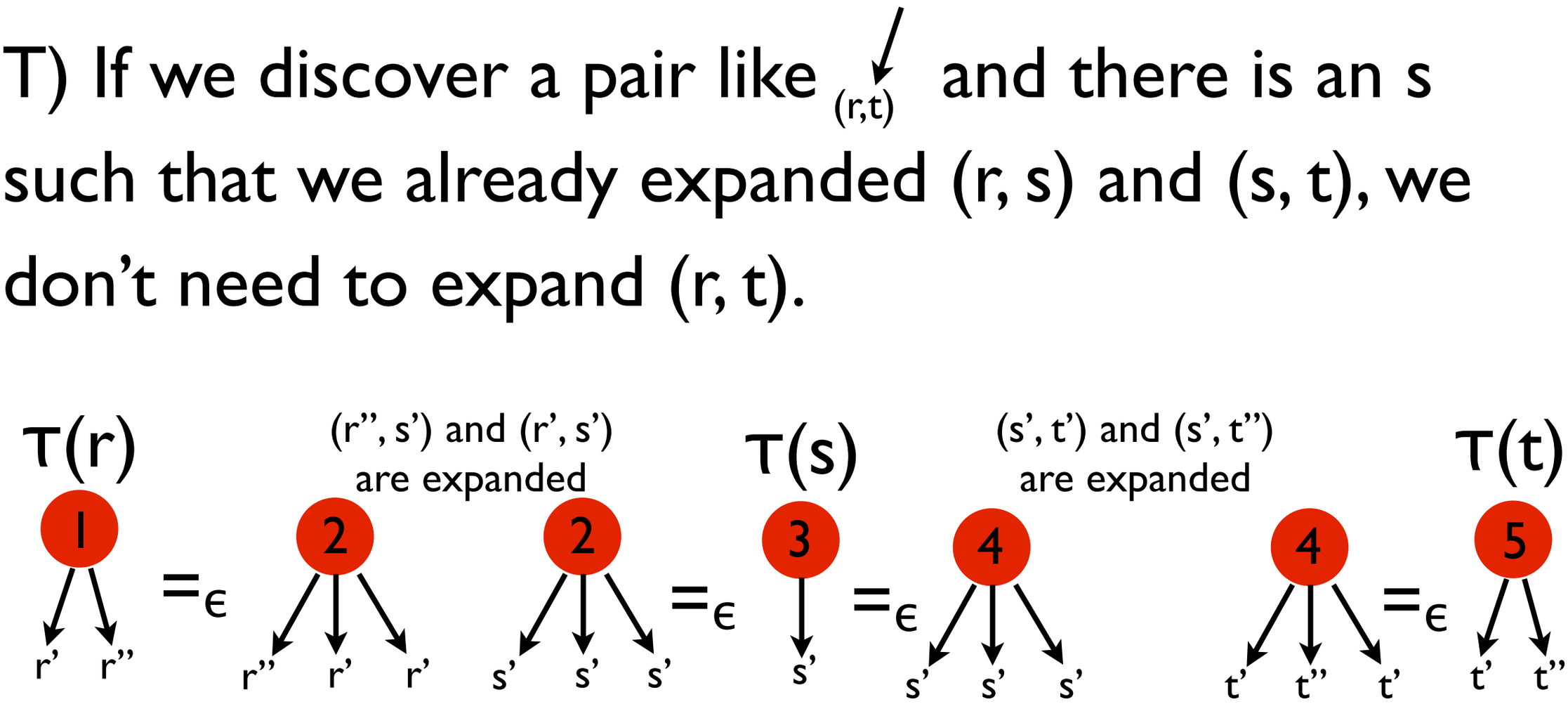
R) If we discover a pair like $\downarrow_{(r,r)}$ we don't need to expand (r, r).

S) If we discover a pair like $\downarrow_{(s,r)}$ and we already expanded (r, s), we don't need to expand (s, r).



Up-to techniques II

T) If we discover a pair like (r,t) and there is an s such that we already expanded (r, s) and (s, t), we don't need to expand (r, t).



We might not find a common type for τ(r) and τ(t)!

An odd situation

"Have the same tree unfoldings" is an equivalence relation on states.

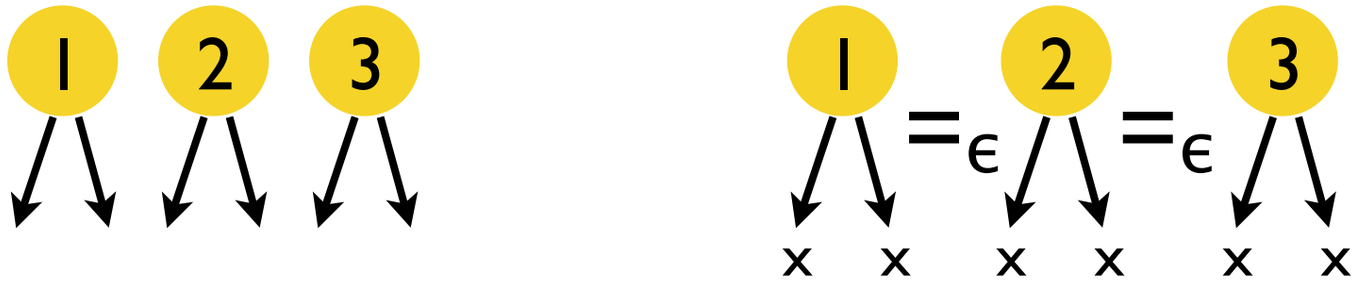
"Are related by a bisimulation" may not be transitive.

Proposition: If (r, s) is in a presentation bisimulation, then r and s have the same tree unfoldings.

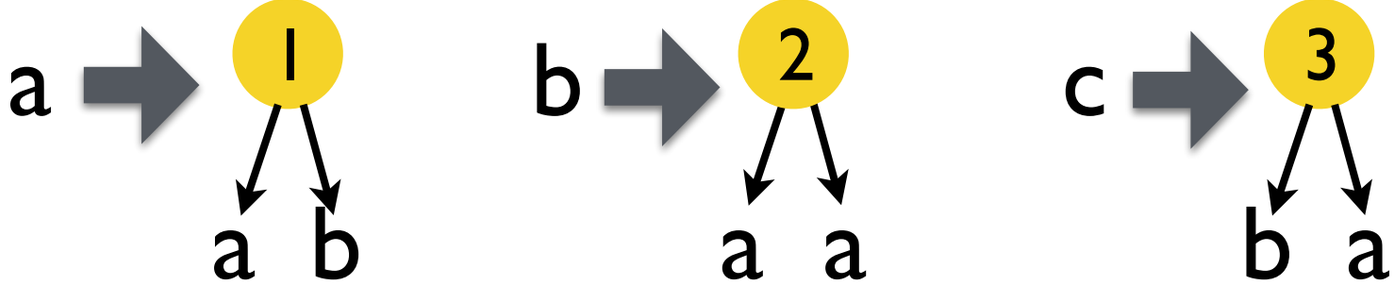
Converse does not hold!!

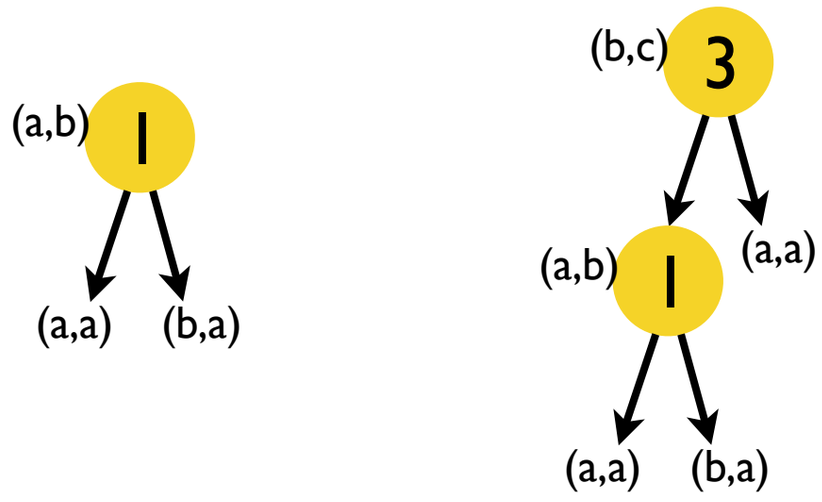
Concrete counterexample

Consider the following presentation:



And this transition system:

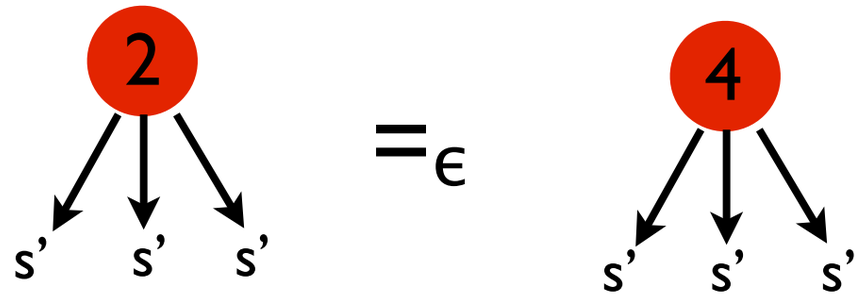




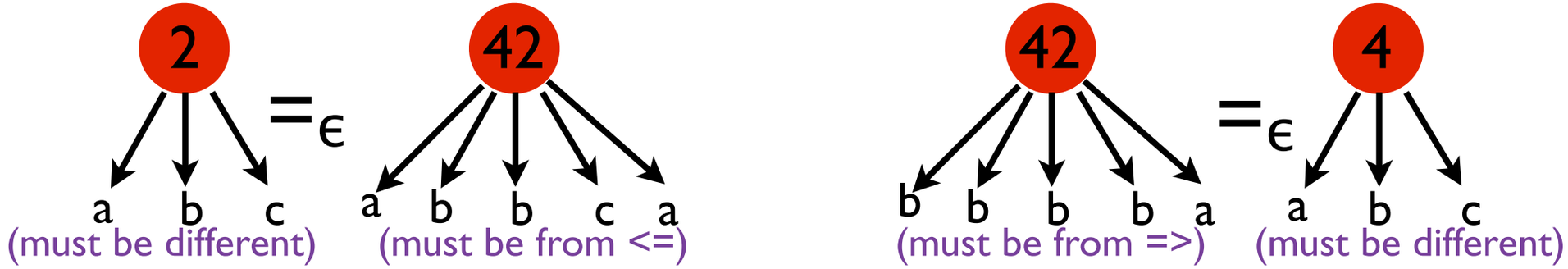
…but there are no rewrites for (a, c)!!

Resolution: domination

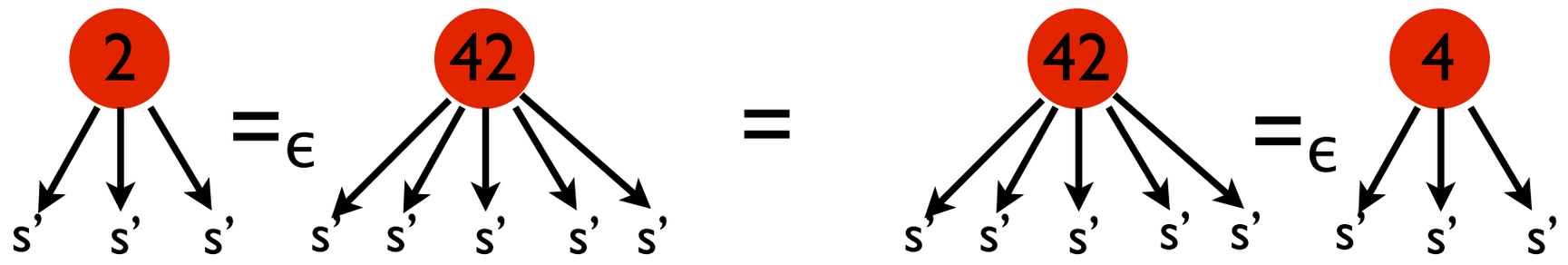
Insist that whenever there is an equation like…



there is a common type and equations of the form…



and when you follow the obvious substitution…



Weak pullback preservation

Theorem (Adámek, Gumm, Trnková):
A system type has a dominated presentation if and only if it preserves weak pullbacks.

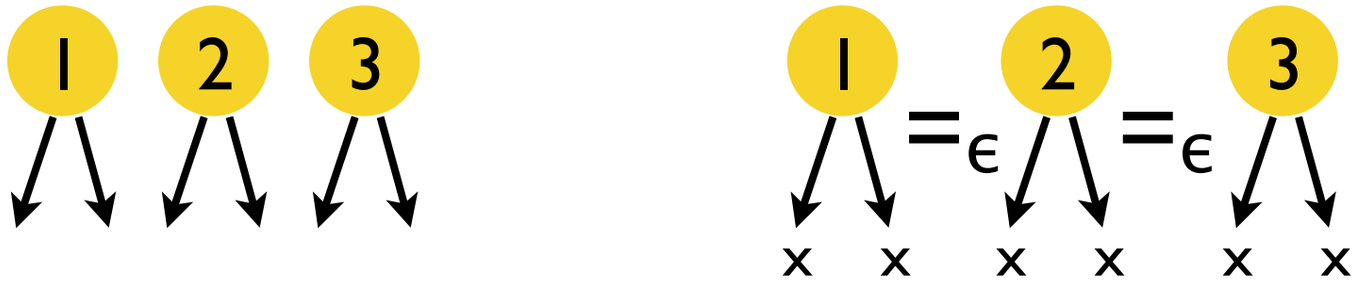
From *Universal Coalgebra*:

Convention 4.4.[3] *In the rest of this paper, set functors* $F : Set \to Set$ *are assumed to preserve weak pullbacks.*[4] *If (the proof of) a lemma, proposition, or theorem actually makes use of this assumption, then it is marked with an asterisk.*

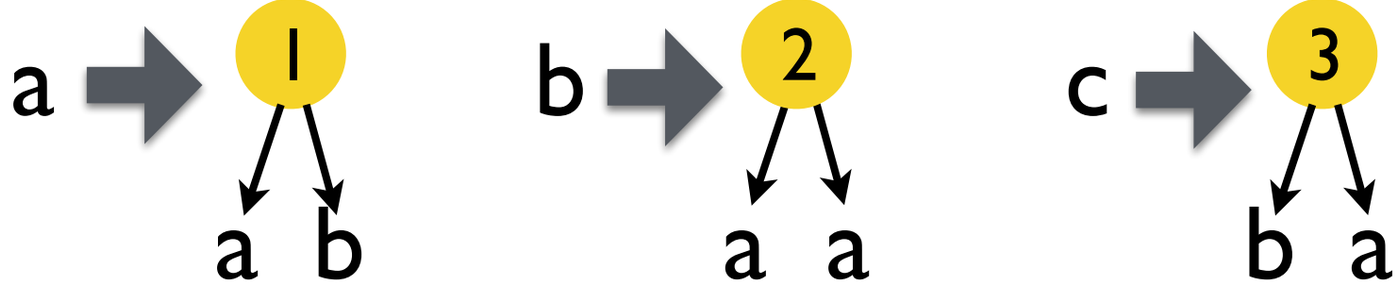
Precongruence

Concrete counterexample

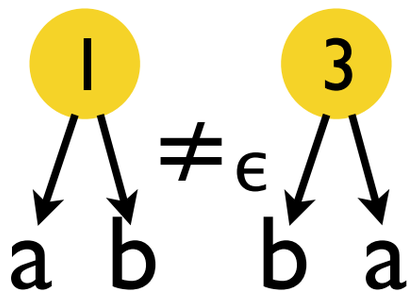
Recall the Aczel-Mendler presentation:



And this transition system:

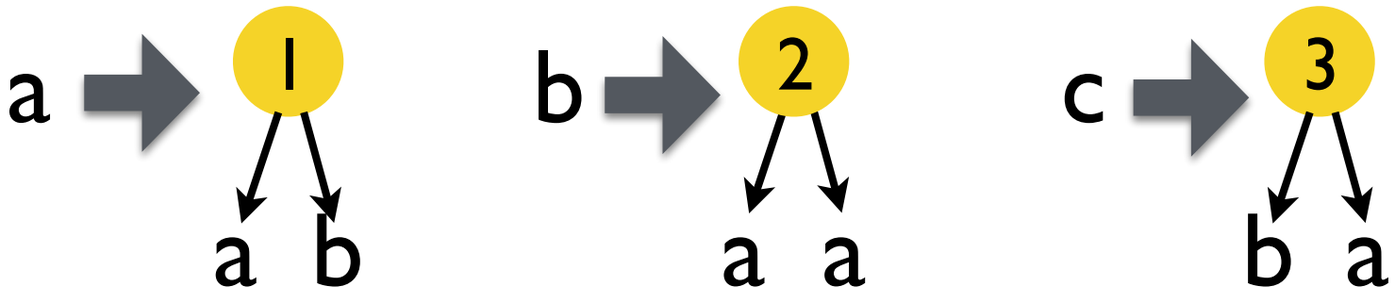


There is no bisimulation with the pair (a, c) since

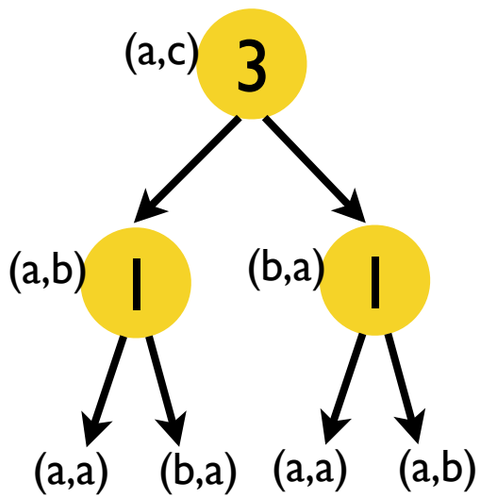


…but a and b have the same unfolds

Concrete counterexample



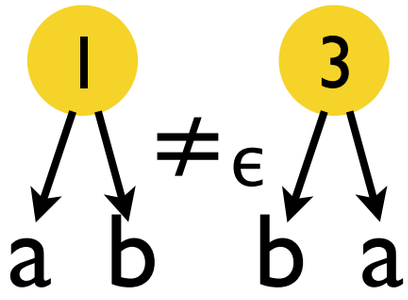
An informal justification:



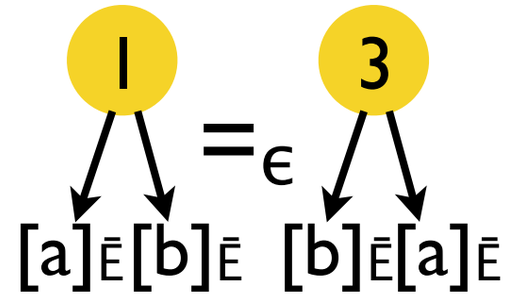
Let E = {(a, c), (a, b), (b, a)}

and $\bar{E} = e(E)$

It is still true that:



But notice:



A different way

Instead of: A normal bisimulation, but to expand a pair (r, s), pick representatives of $[\tau(r)]_{=\epsilon}$ and $[\tau(s)]_{=\epsilon}$ with common type.

Do this: To expand (r, s), optimistically choose any transition structure $T_{r,s}$ whatsoever. After the tree is finished, let E relate expanded pairs as above. Justify all choices by verifying

$$[\tau(r)]_{\bar{E}} =_\epsilon [T_{r,s}]_{\bar{E}} =_\epsilon [\tau(s)]_{\bar{E}}$$

Such an E is a *precongruence*.

Trust me, I'm a doctor…

Proposition: If (r, s) is a pair in a precongruence, then r and s have the same tree unfoldings.

Proof: It turns out this is precisely the condition needed for the quotient map to be a "machine morphism", which we combine with the fact that morphisms preserve unfoldings.

Up-to techniques III

R) If we discover a pair like $\underset{(r,r)}{\swarrow}$ we don't need to expand (r, r).

We will still need to justify

$$[\tau(x)]_{e(E-(r,r))} =_\epsilon [T_{x,y}]_{e(E-(r,r))} =_\epsilon [\tau(y)]_{e(E-(r,r))}$$

for the remaining (x, y) in E-(r,r). But e(E-(r,r)) = e(E).

Up-to techniques III

S) If we discover a pair like $\downarrow_{(s,r)}$ and we already expanded (r, s), we don't need to expand (s, r).

We will still need to justify

$$[\tau(x)]_{e(E-(s,r))} =_\epsilon [T_{x,y}]_{e(E-(s,r))} =_\epsilon [\tau(y)]_{e(E-(s,r))}$$

for the remaining (x, y) in E-(s,r).

But e(E-(s,r)) = e(E) since (r, s) is still in E.

T) Transitive closure is similar.

Better guarantees

Theorem (Aczel, Mendler)**:** A pair (r, s) is a precongruence if and only if r and s have the same tree unfoldings.

Theorem (S)**:** A logic which finds precongruences is sound and complete with respect to unfolding semantics. On finite systems, it is decidable.

Concluding remarks

Main messages

- Presentations of transition systems cover all the finitary system types
- Precongruences detect tree unfolding semantics better than bisimulations for finitary systems
- Precongruences support more sound up-to techniques than bisimulations

Future work

- Language inclusion is another kind of (transitive) relation on tree unfoldings. Weak pullback preservation is often used here.
- Generalize to non-Set coalgebras?

Thanks!