Transition Systems and Bisimulation

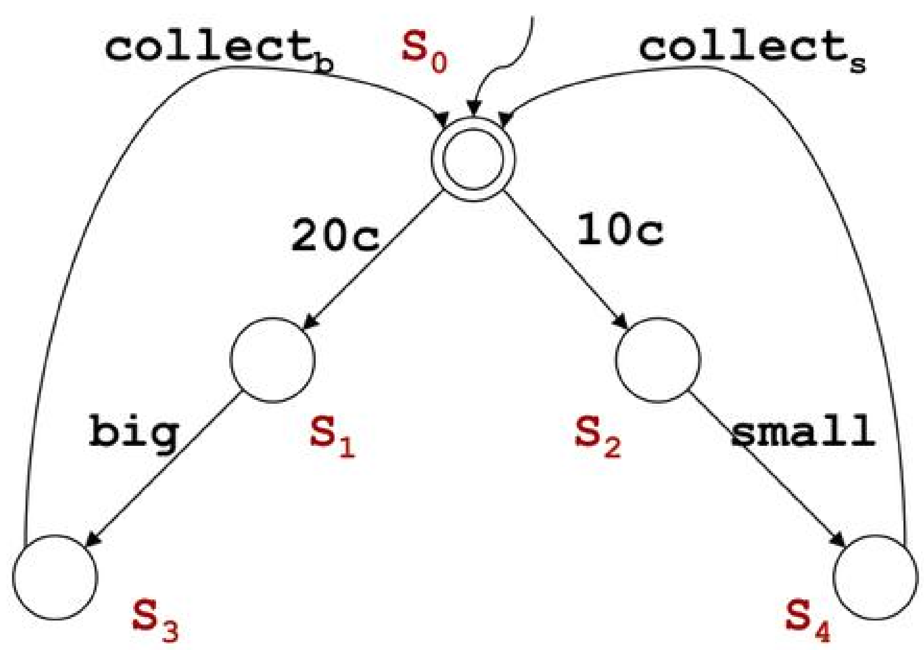
Giuseppe De Giacomo

Transition Systems and Bisimulation

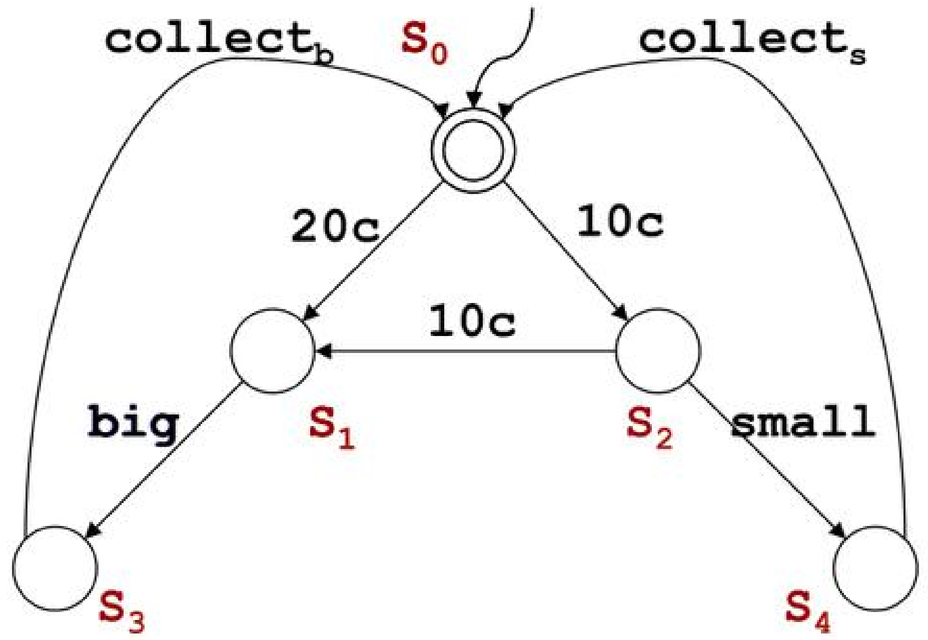
Giuseppe De Giacomo

Transition Systems

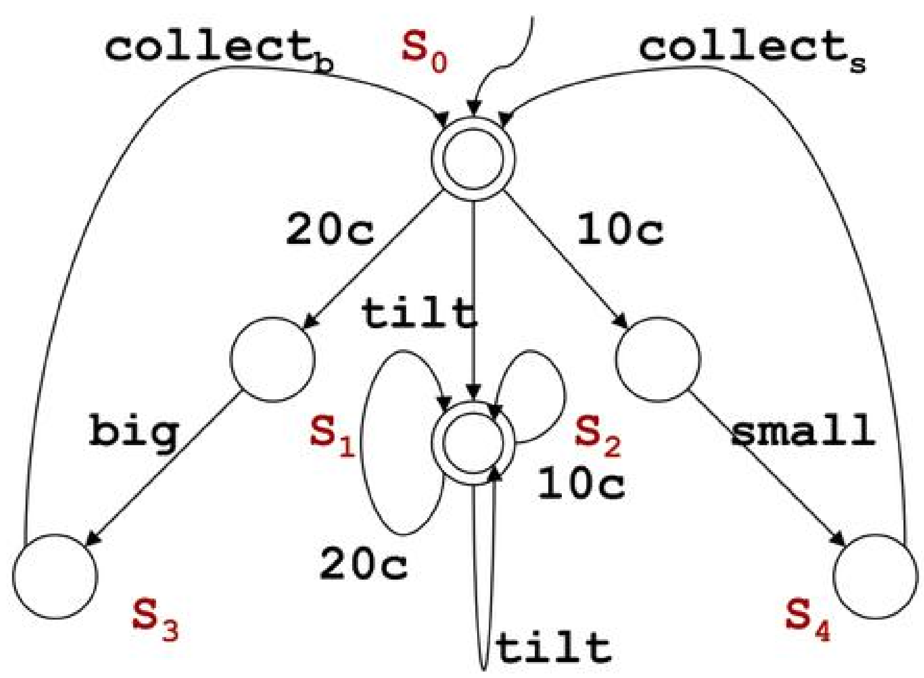
Concentrating on behaviors: Vending Machine



Concentrating on behaviors: Another Vending Machine

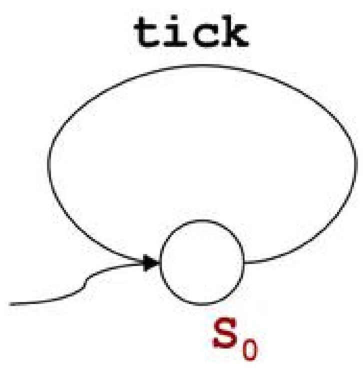


Concentrating on behaviors: Vending Machine with Tilt



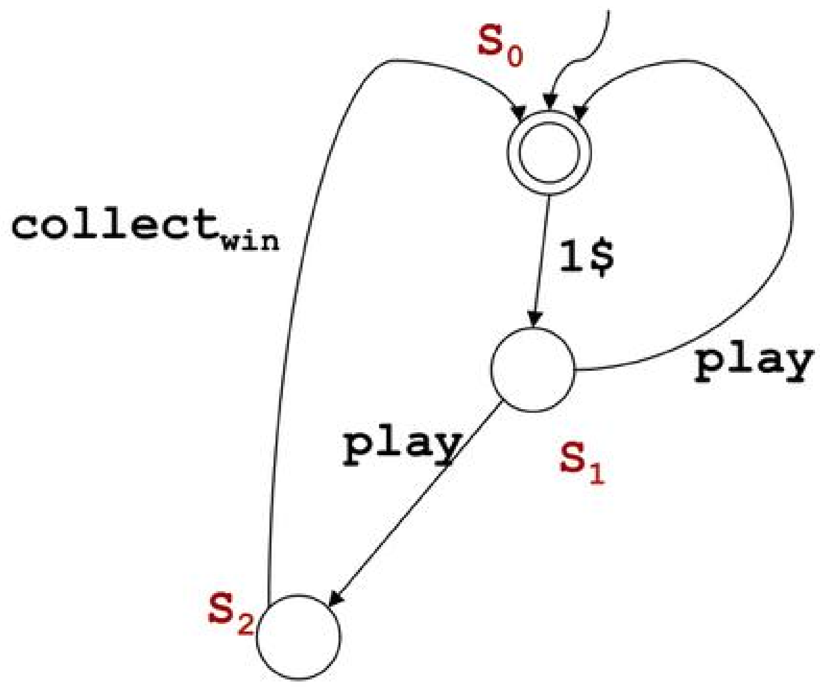
Example (Clock)

TS may describe (legal) nonterminating processes

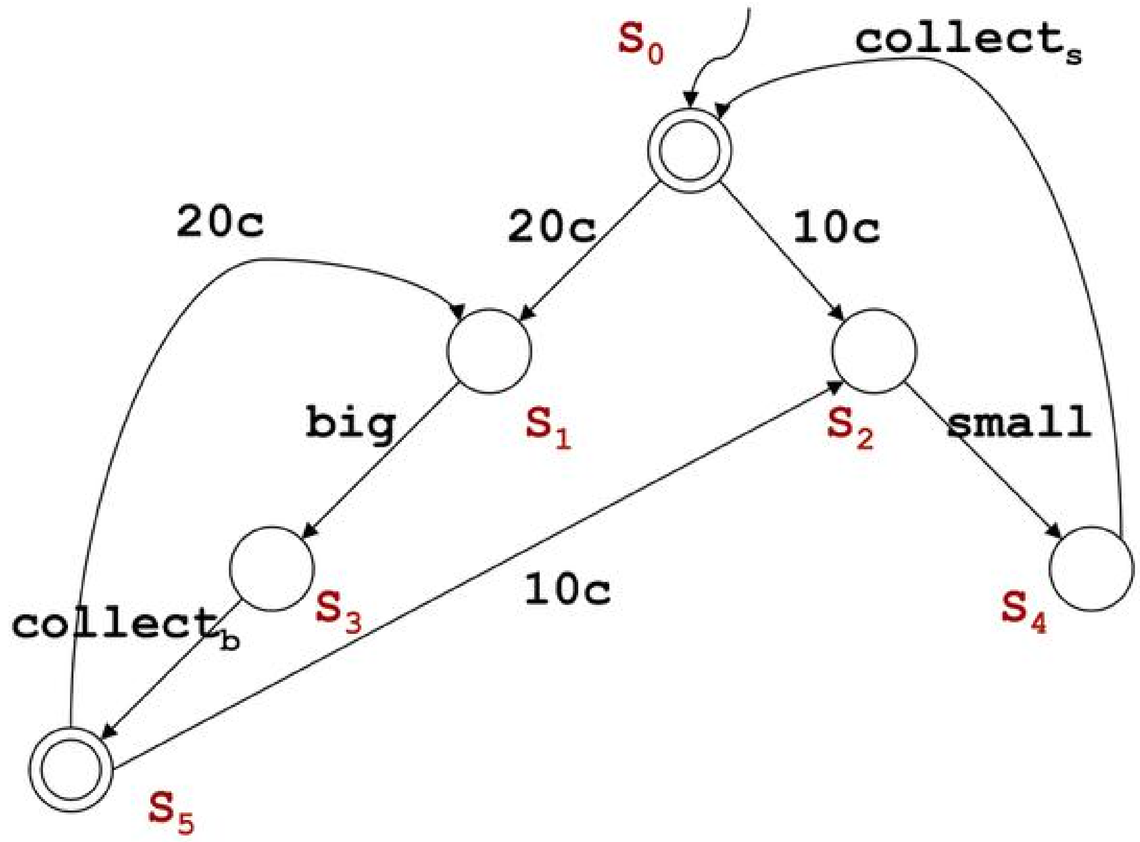


Example (Slot Machine)

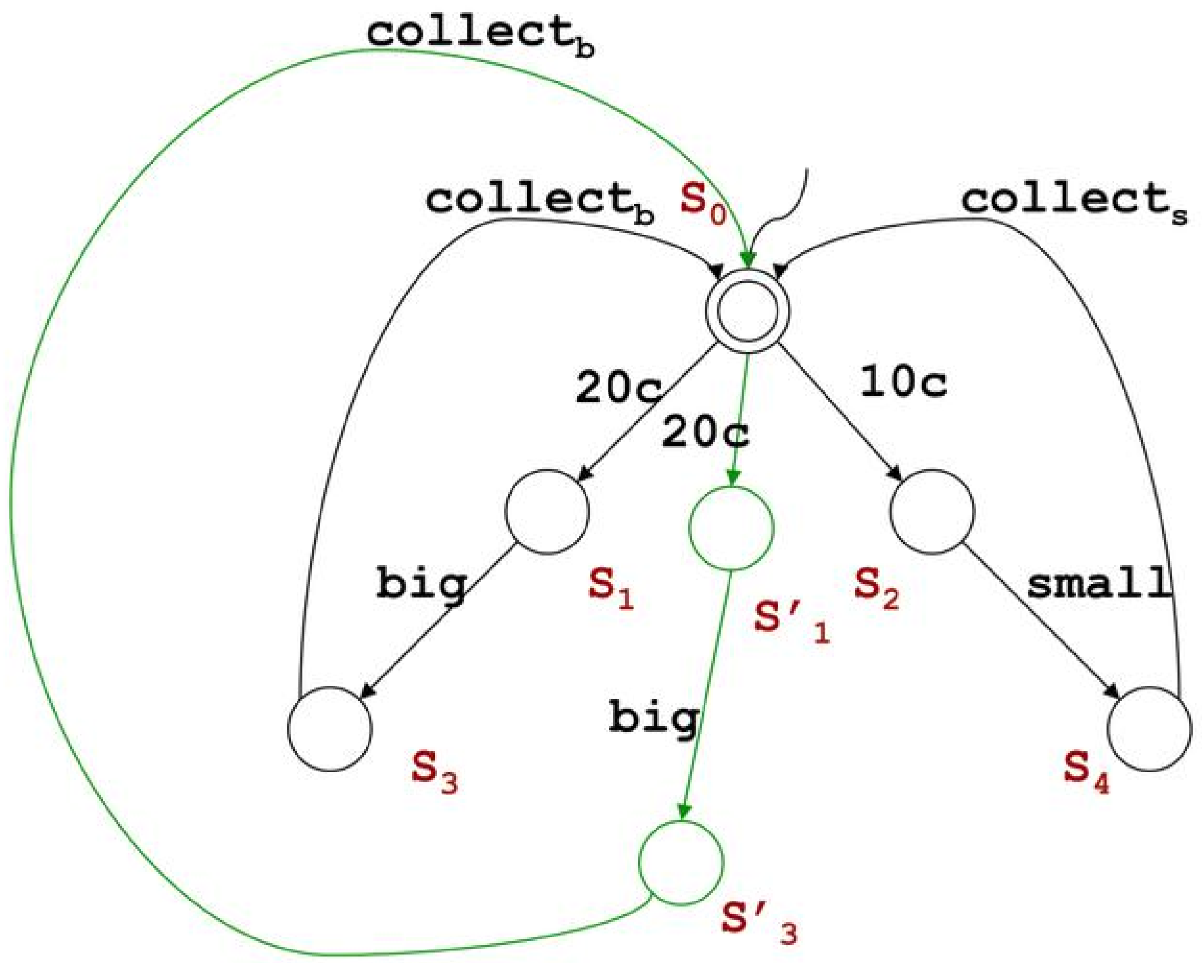
Nondeterministic transitions express
choice that is **not** under the **control** of clients



Example (Vending Machine - Variant 1)



Example (Vending Machine - Variant 2)



Transition Systems

SAPIENZA
UNIVERSITÀ DI ROMA

- A transition system TS is a tuple $T = < A, S, S^0, \delta, F>$ where:
  - A is the set of actions
  - S is the set of states
  - $S^0 \subseteq S$ is the set of initial states
  - $\delta \subseteq S \times A \times S$ is the transition relation
  - $F \subseteq S$ is the set of final states

(c.f. Kripke Structure)

- Variants:
  - No initial states
  - Single initial state
  - Deterministic actions
  - States labeled by propositions other than Final/¬Final

Inductive vs Coinductive Definitions: Reachability, Bisimilarity, …

Reachability

- A binary relation R is a **reachability-like relation** iff:
  - $(s,s) \in R$
  - if $\exists\, a, s'.\ s \rightarrow_a s' \wedge (s',s'') \in R$ then $(s,s'') \in R$

- A state s_0 of transition system S **reaches** a state s_f iff for **all** a **reachability-like relations** R we have $(s_0, s_f) \in R$.

- Notably that
  - **reaches** is a reachability-like relation itself
  - **reaches** is the smallest reachability-like relation

Note it is a inductive definition!

Computing Reachability on Finite Transition Systems

Algorithm ComputingReachability

Input: transition system TS
Output: the **reachable-from** relation (the smallest reachability-like relation)

Body

$R = \emptyset$
$R' = \{(s,s) \mid s \in S\}$
while ($R \neq R'$) {
  $R := R'$
  $R' := R' \cup \{(s,s'') \mid \exists s',a.\ s \rightarrow_a s' \wedge (s',s'') \in R\}$
}
return R'

YdoB

*This algorithm is based on computing iteratively fixpoint approximates for the **least fixpoint**, starting from the empty set.*

Bisimulation

Intuition:

Two (states of two) transition systems are bisimilar if they have the same behavior.

In the sense that:

- *Locally they (the two **states**) look indistinguishable*
- *Every **action** that can be done on one of them can also be done on the other remaining indistinguishable*

Bisimulation

- A binary relation R is a **bisimulation** iff:

  $(s,t) \in R$ implies that
  - s is *final* iff t is *final*
  - for all actions a
    - if $s \rightarrow_a s'$ then $\exists t' . t \rightarrow_a t'$ and $(s',t') \in R$
    - if $t \rightarrow_a t'$ then $\exists s' . s \rightarrow_a s'$ and $(s',t') \in R$

- A state s_0 of transition system S is **bisimilar**, or simply **equivalent,** to a state t_0 of transition system T iff there **exists** a **bisimulation** between the initial states s_0 and t_0.

- Notably
  - **bisimilarity** is a bisimulation
  - **bisimilarity** is the largest bisimulation

Note it is a co-inductive definition!

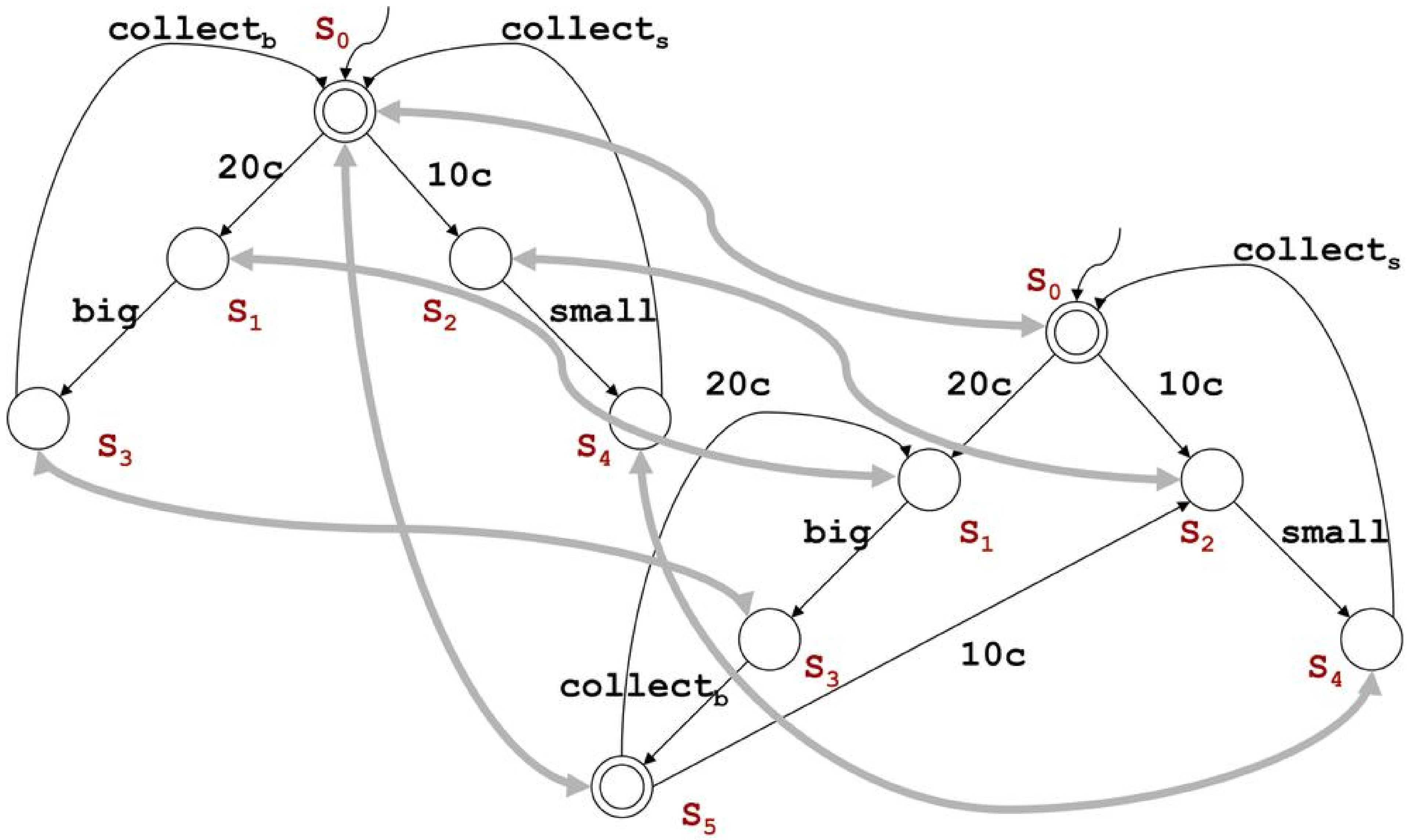
Computing Bisimulation on Finite Transition Systems

Algorithm ComputingBisimulation
Input: transition system TS_S = < A, S, S^0, δ_S, F_S> and
transition system TS_T = < A, T, T^0, δ_T, F_T>
Output: the **bisimilarity** relation (the largest bisimulation)

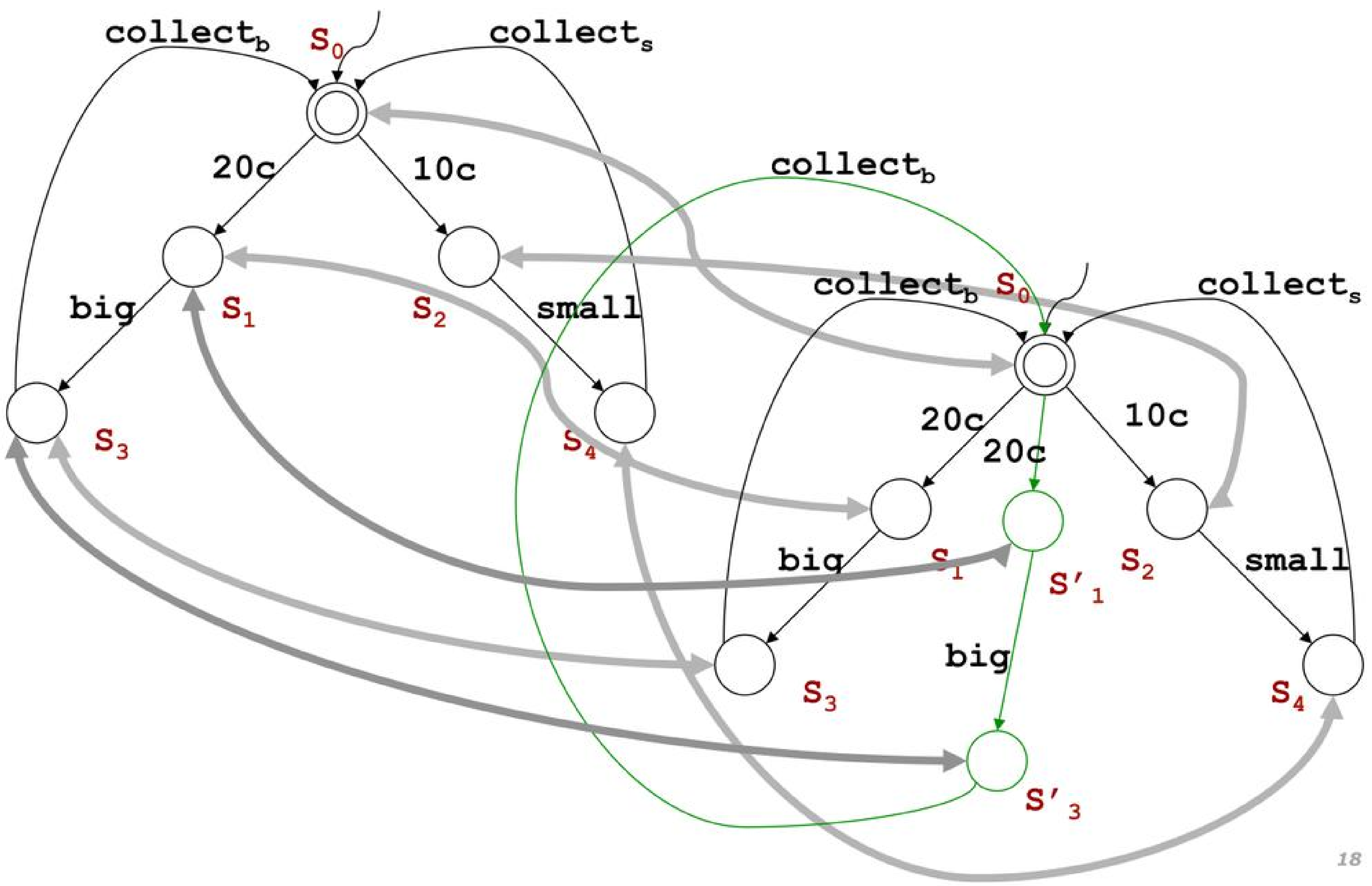
Body
$R = S \times T$
$R' = R - \{(s,t) \mid \neg(s \in F_S \equiv t \in F_T)\}$
while ($R \neq R'$) {
$R := R'$
$R' := R' - (\{(s,t) \mid \exists s', a.\ s \to_a s' \wedge \neg\exists t'.\ t \to_a t' \wedge (s',t') \in R'\}$
$\{(s,t) \mid \exists t', a.\ t \to_a t' \wedge \neg\exists s'.\ s \to_a s' \wedge (s',t') \in R'\})$
}
return R'
Ydob

*This algorithm is based on computing iteratively fixpoint approximates for the **greatest fixpoint**, starting from the total set (SxT).*

Example of Bisimulation



Example of Bisimulation

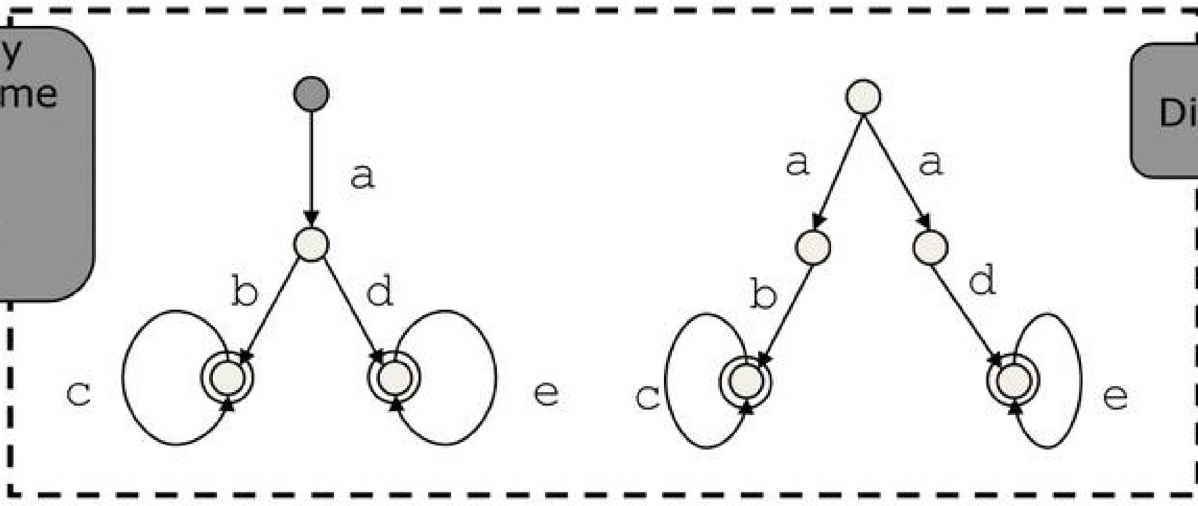


Automata vs.Transition Systems

- Automata
  - define sets of runs (or traces or strings): (finite) length sequences of actions
- TSs
  - … but I can be interested also in the alternatives "encountered" during runs, as they represent client's "choice points"

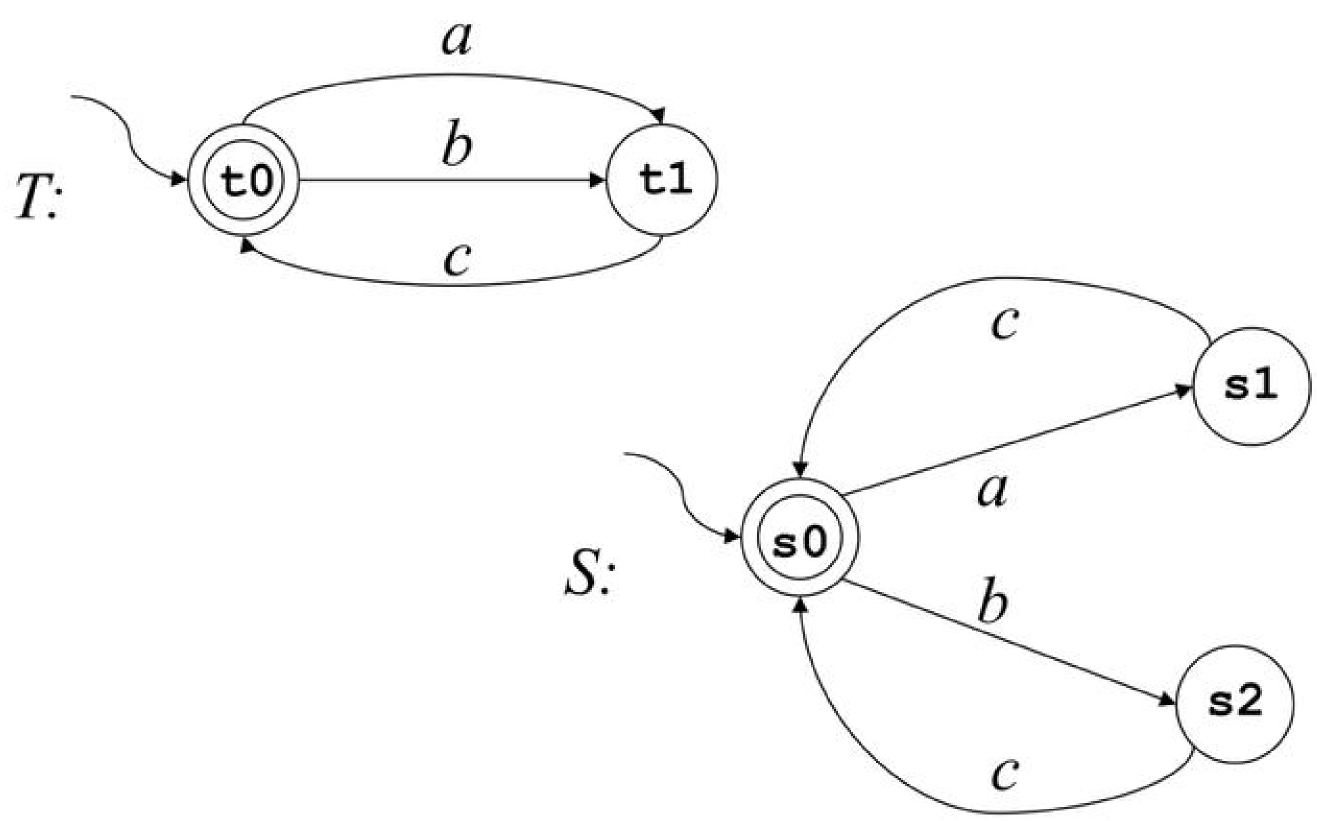
As automata they recognize the same language:

abc* + ade*



Different as TSs

Example of Bisimulation



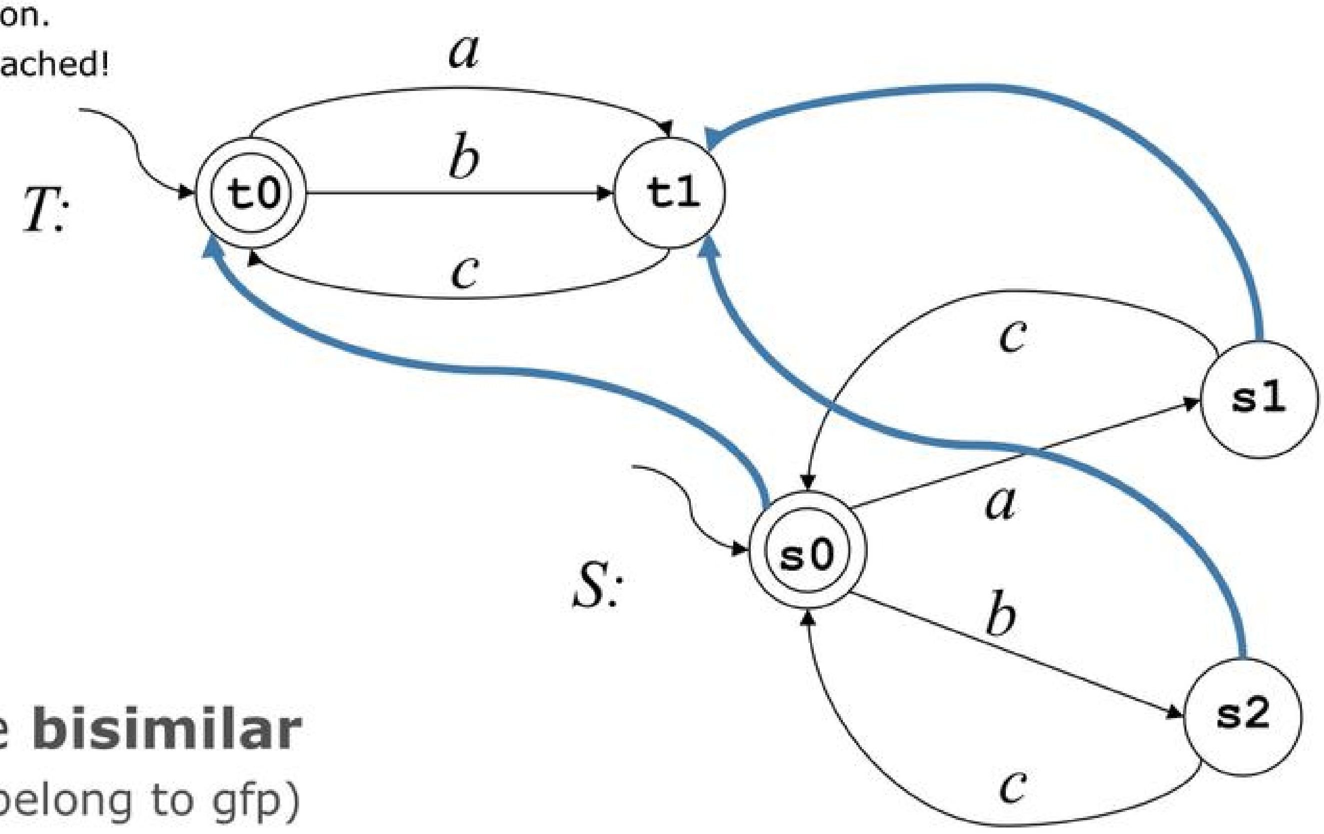
Are S and T **bisimilar?**

Computing Bisimulation

We need to compute the greatest fixpoint (gfp): we do it by computing approximates starting from the Cartesian product:

- R0={(t0,s0), (t0,s1), (t0,s2), (t1,s0), (t1,s1),(t1,s2)} – Cartesian product
- R1={(t0,s0),(t1,s1),(t1,s2)} – removed those pairs that violate local condition on final (final iff final)
- R2={(t0,s0),(t1,s1),(t1,s2)} – removed those pairs where one can do action and other cannot copy remaining in the relation.

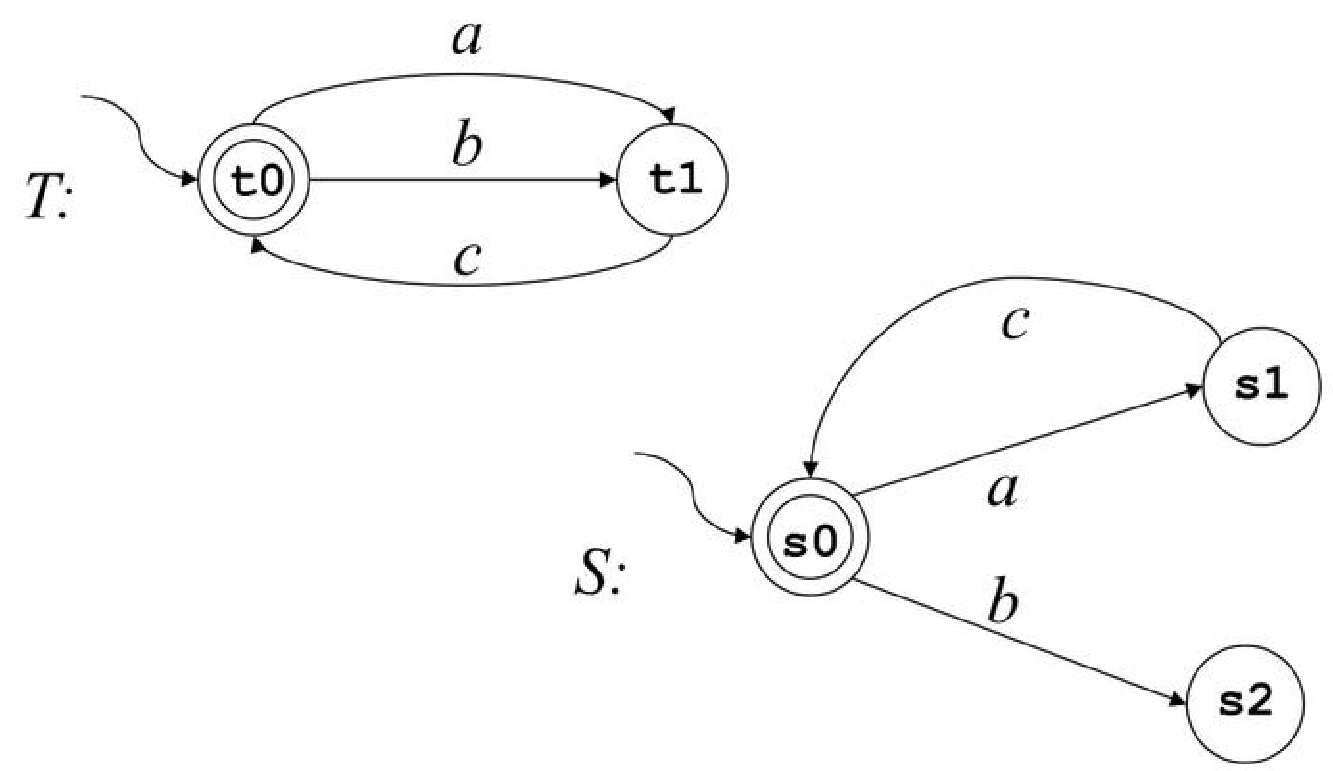
R1=R2 greatest fixpoint reached!



S and T are **bisimilar**

((t0,s0) do belong to gfp)

Example of NON Bisimulation



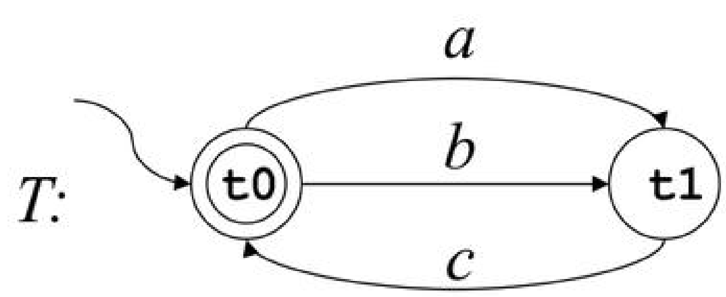
Are S and T **bisimilar?**

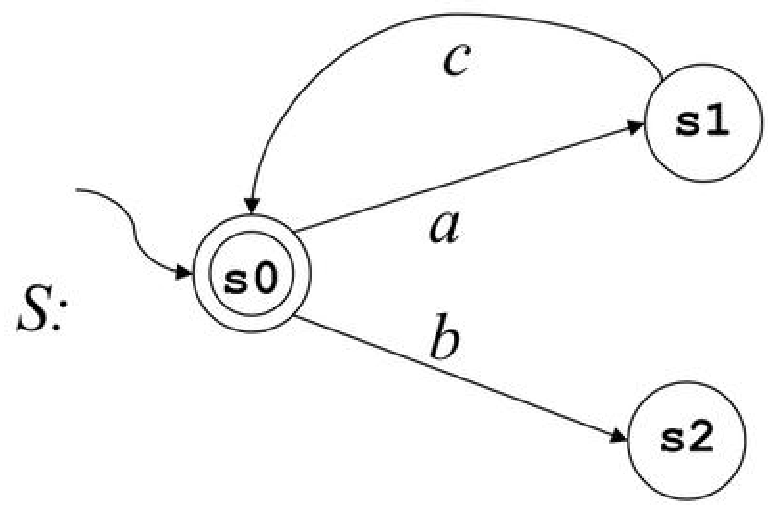
Computing Bisimulation

We need to compute the greatest fixpoint: we do it by computing approximates starting from the cartesian product:

- R0={(t0,s0), (t0,s1), (t0,s2), (t1,s0), (t1,s1),(t1,s2)} – cartesian product
- R1={(t0,s0),(t1,s1),(t1,s2)} – removed those pairs that violate local condition on final (final iff final)
- R2={(t0,s0),(t1,s1)} – removed (t1,s2) since t1 can do c but s2 cannot.
- R3={(t1,s1)} – removed (t0,s0) since t0 can do b, s2 can do b as well, but then the resulting states (t1,s2) are NOT in R2.
- R4 = {} – removed (t1,s1) since t1 can do c, s1 can do c as well, but then the resulting states (t0,s0) are NOT in R3.
- R5 = {}

R4=R5 greatest fixpoint reached!





S and T are NOT **bisimilar**

((t0,s0) do not belong to gfp)