Transition Systems and Bisimulation

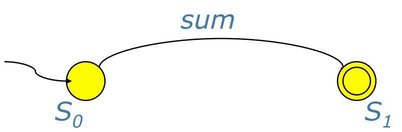
Giuseppe De Giacomo

Service Integration
A.A. 2008/09

Transition Systems

Concentrating on behaviors: SUM two integers

- Consider a program for computing the sum of two integers.
- Such a program has essentially two states
  - the state S0 of the memory before the computation: including the two number to sum
  - the state S1 of the memory after the computation: including the result of the computation
- Only one action, i.e. "sum", can be performed

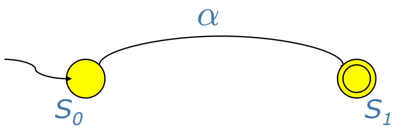


Concentrating on behaviors: CheckValidity

- Consider a program for computing the validity of a FOL formula:
- Also such a program has essentially two states
  - the state S_1 of the memory before the computation: including the formula to be checked
  - the state S_2 of the memory after the computation: including "yes", "no", "time-out"
- Only one action, i.e. "*checkValidity*", can be performed

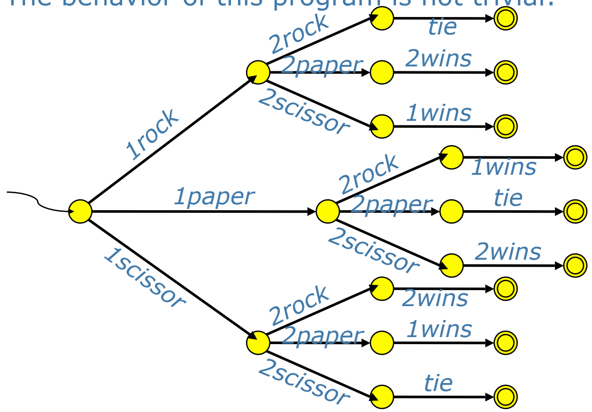
Concentrating on behaviors

- The programs SUM and CheckValidity are very different from a computational point of view.
  - SUM is trivial
  - CheckValidity is a theorem prover hence very complex
- However they are equally trivial from a behavioral point of view:
  - two states S_1 and S_2
  - a single action α causing the transition



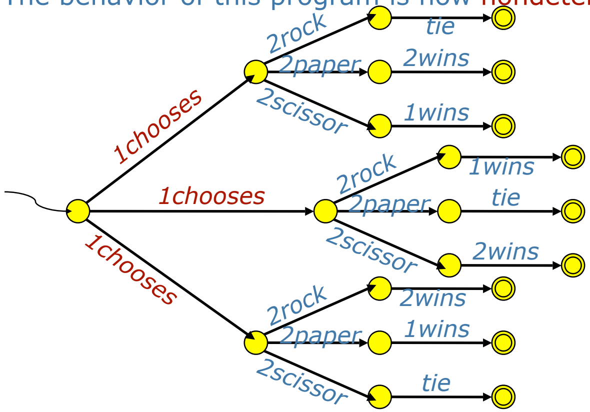
Concentrating on behaviors: RockPaperScissor

- Consider the program RockPaperScissor that allows to play two players the the well-known game.
- The behavior of this program is not trivial:



Concentrating on behaviors: RockPaperScissor (automatic)

- Consider a variant of the program RockPaperScissor that allows one players to play against the computer.
- The behavior of this program is now nondeterministic:

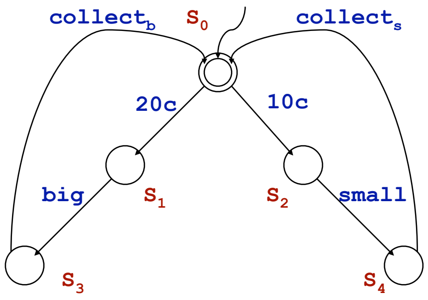


Concentrating on behaviors: WebPage

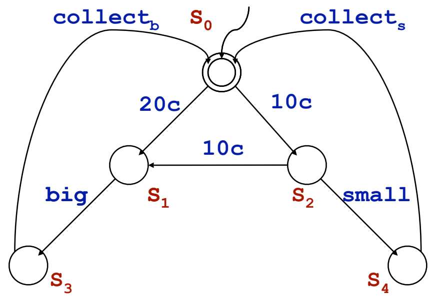
http://www.informatik.uni-trier.de/~ley/db/

A web page can have a complex behavior!

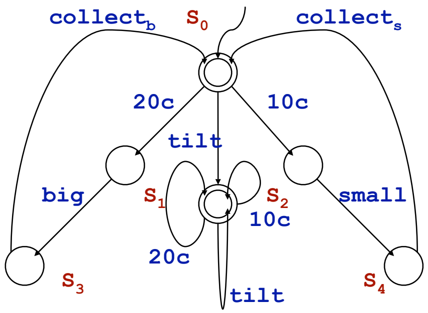
Concentrating on behaviors: Vending Machine



Concentrating on behaviors: Another Vending Machine



Concentrating on behaviors: Vending Machine with Tilt



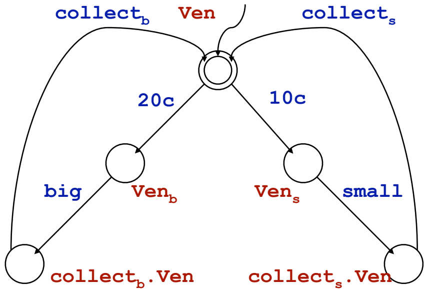
Transition Systems

- A transition system TS is a tuple T = < A, S, S^0, δ, 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
- Variants:
  - No initial states
  - Single initial state
  - Deterministic actions
  - States labeled by propositions other than Final/¬Final

(c.f. Kripke Structure)

Process Algebras are Formalisms for Describing TS

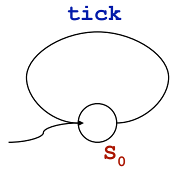
- Trans (a la CCS)
  - $Ven = 20c.Ven_b + 10c.Ven_s$
  - $Ven_b = big.collect_b.Ven$
  - $Ven_l = small.collect_s.Ven$
- Final
  - $\surd$ Ven



- *TS may have infinite states* - *e.g., this happens when generated by process algebras involving iterated concurrency*
- *However we have good formal tools to deal only with finite states TS*

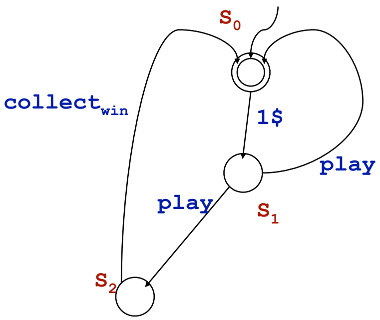
Example (Clock)

TS may describe (legal) nonterminating processes

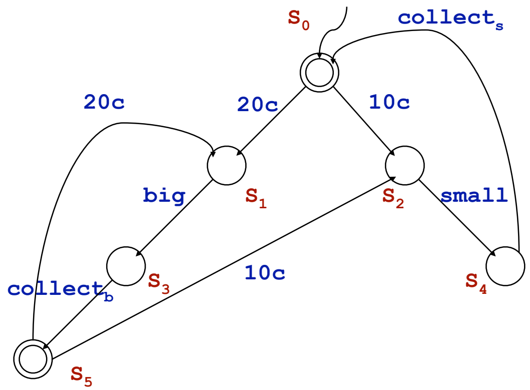


Example (Slot Machine)

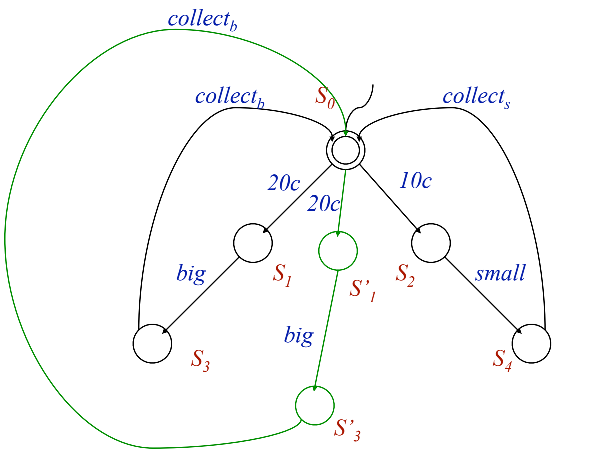
Nondereminisic transitions express
choice that is not under the control of clients



Example (Vending Machine - Variant 1)



Example (Vending Machine - Variant 2)



Inductive vs Coinductive Definitions: Reachability, Bisimilarity, …

Reachability

- A binary relation R is a **reachability-like relation** iff:
  - (s,s) ∈ R
  - if ∃ a, s′. s $\rightarrow_a$ s′ ∧ (s′,s′′) ∈ R then (s,s′′)∈ 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)∈ 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 = ∅
R′ = {(s,s) | s ∈ S}
while (R ≠ R′) {
  R := R′
  R′ := R′ ∪ {(s,s′′) | ∃ s′,a. s $\rightarrow_a$ s′ ∧ (s′,s′′)∈ R }
}
return R′
YdoB

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

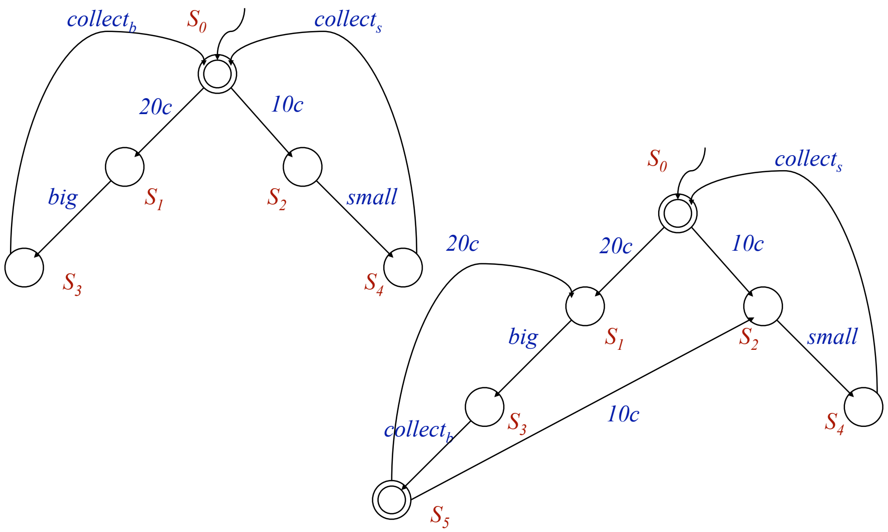
Note it is a co-inductive definition!

Computing Bisimilarity on Finite Transition Systems

Algorithm ComputingBisimulation
Input: transition system $TS_S = \langle A, S, S^0, \delta_S, F_S \rangle$ and
transition system $TS_T = \langle A, T, T^0, \delta_T, F_T \rangle$
Output: the **bisimilarity** relation (the largest bisimulation)

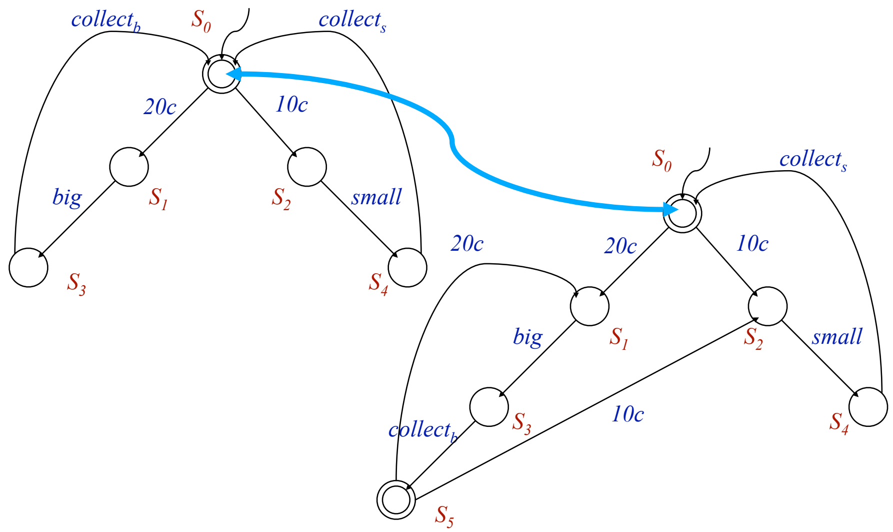
Body
$R = S \times T$
$R' = S \times T - \{(s,t) \mid \neg(s \in F_S \equiv t \in F_T)\}$
while $(R \neq R')$ {
$R := R'$
$R' := R' - (\{(s,t) \mid \exists s',a.\ s \rightarrow_a s' \wedge \neg\exists t' .\ t \rightarrow_a t' \wedge (s',t') \in R' \}$
$\{(s,t) \mid \exists t',a.\ t \rightarrow_a t' \wedge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

Example of Bisimulation

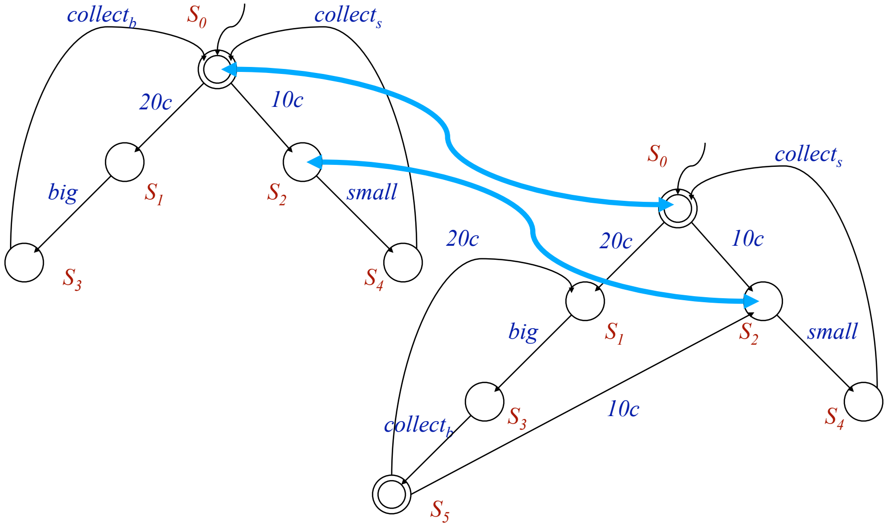


Dipartimento di Informatica e Sistemistica "Antonio Ruberti" SAPIENZA Università di Roma

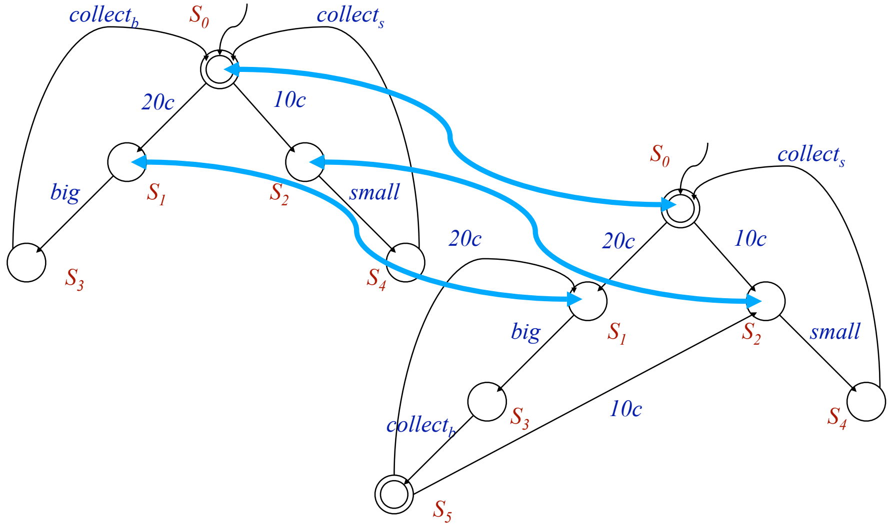
Example of Bisimulation



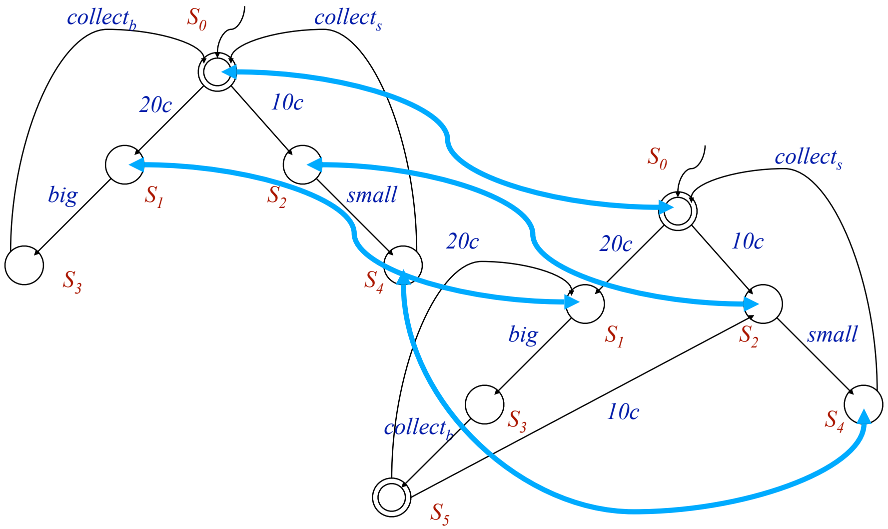
Example of Bisimulation



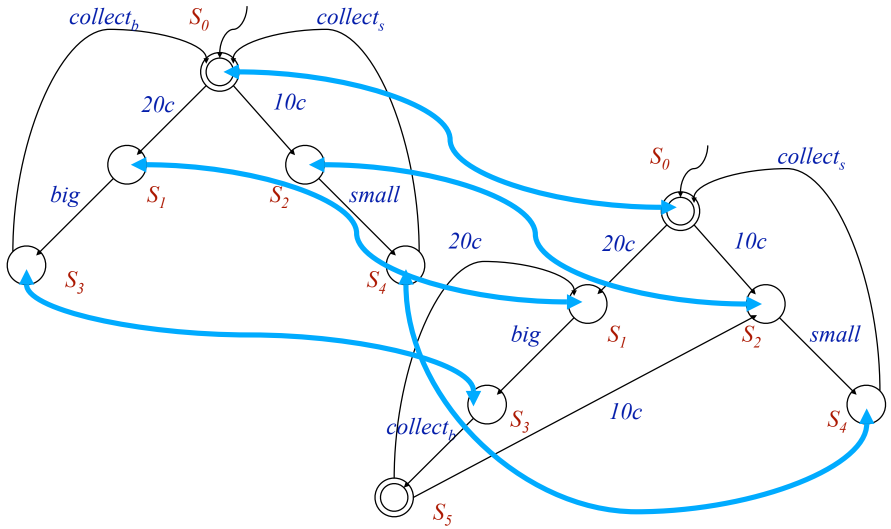
Example of Bisimulation



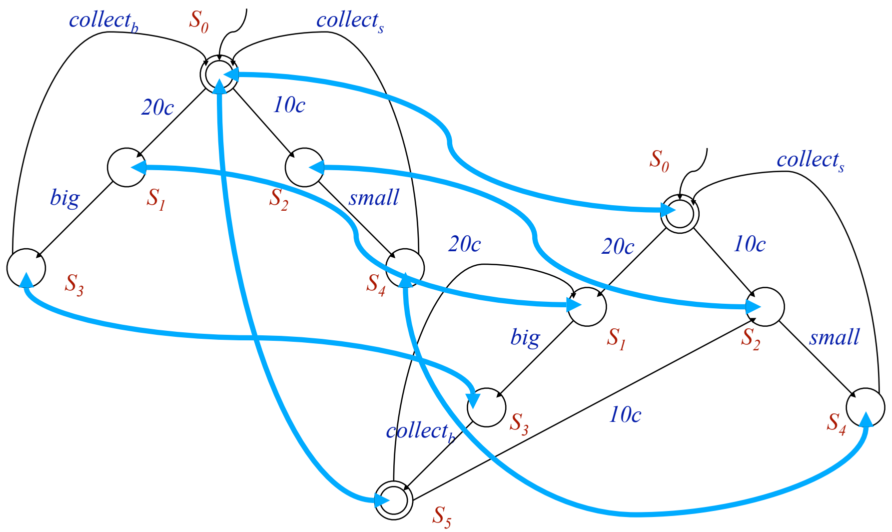
Example of Bisimulation



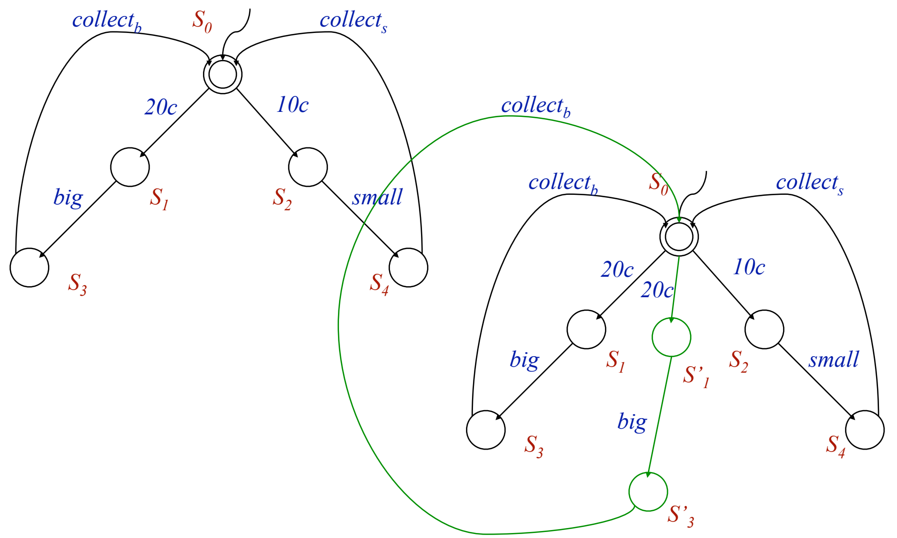
Example of Bisimulation



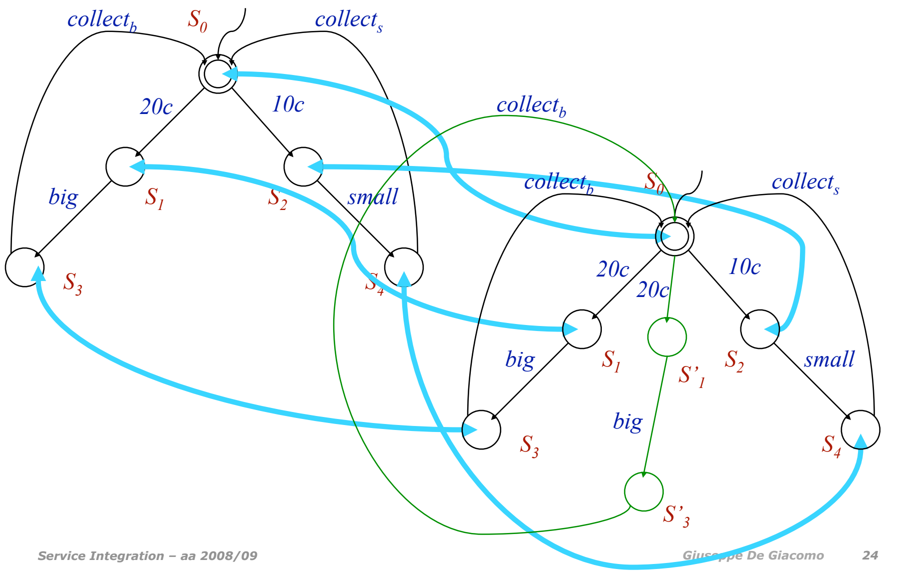
Example of Bisimulation



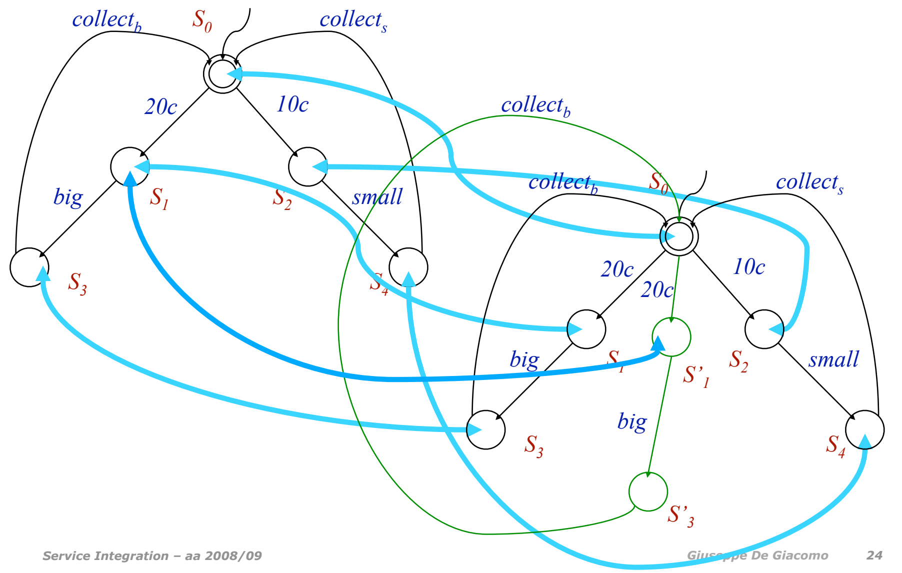
Example of Bisimulation



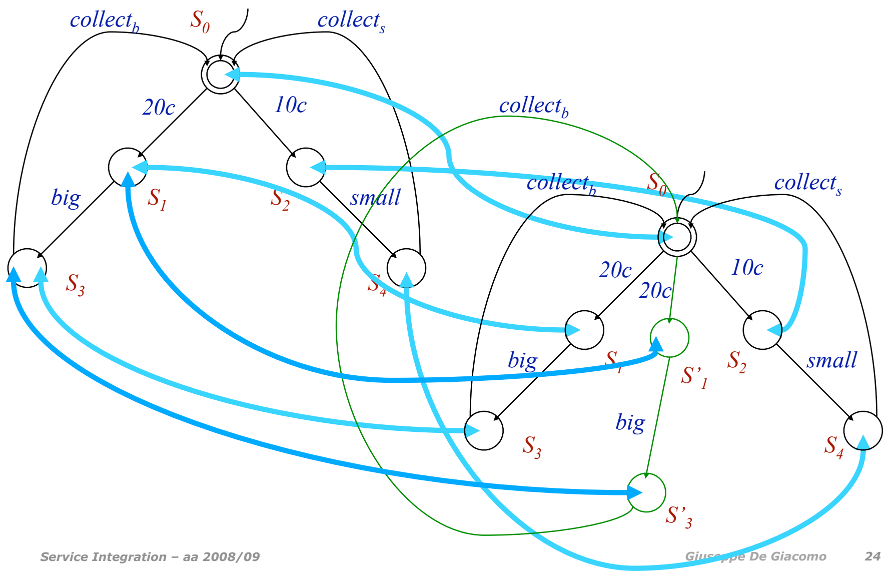
Example of Bisimulation



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"