

# Composition of Services that Share an Infinite-State Blackboard

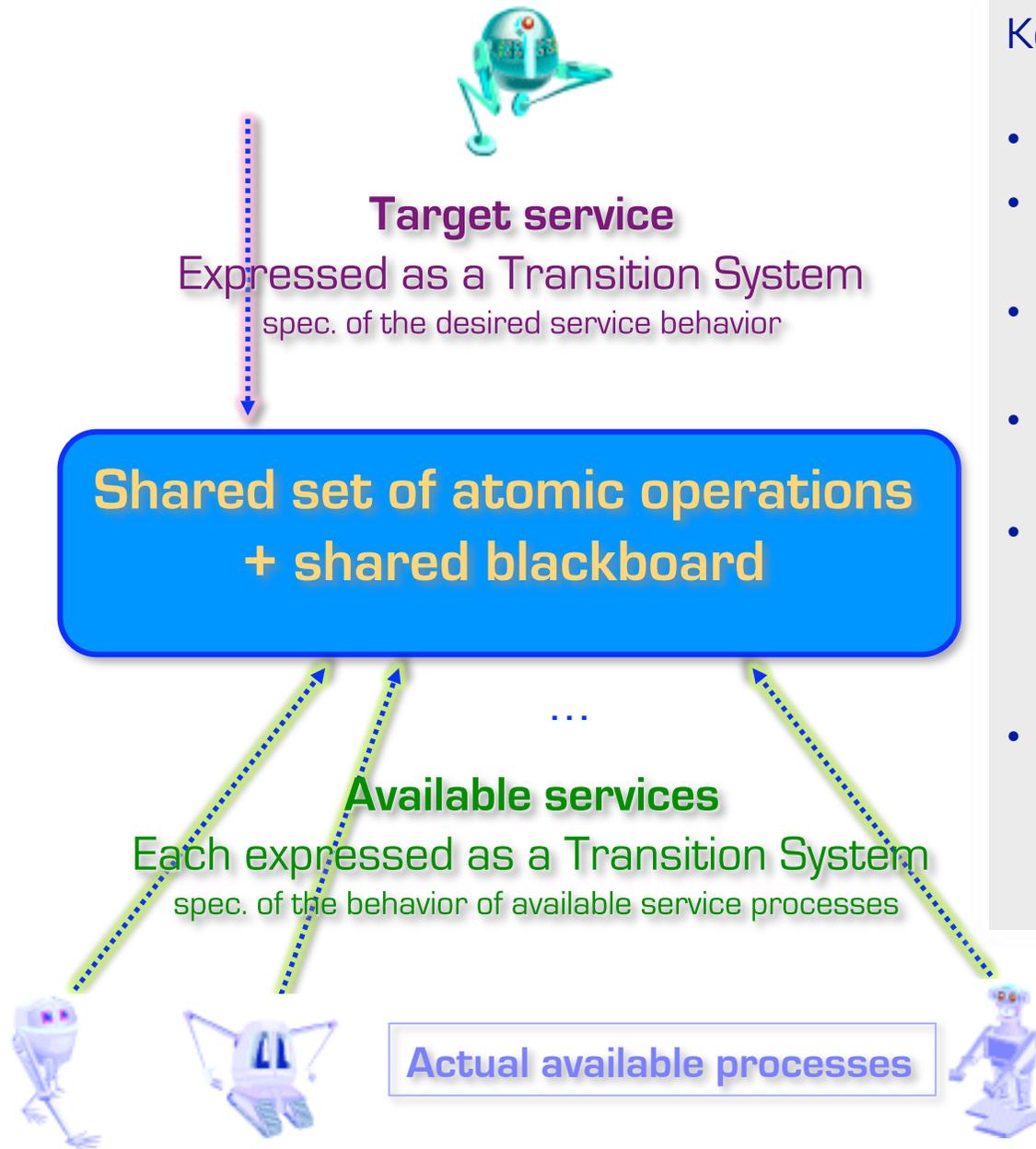
Fabio Patrizi & Giuseppe De Giacomo

SAPIENZA Università di Roma, Italy

*IIVWEB'09 - IJCAI'09 Workshop on Information Integration on the Web*

*Pasadena, California, July 11, 2009*

# Basic ideas

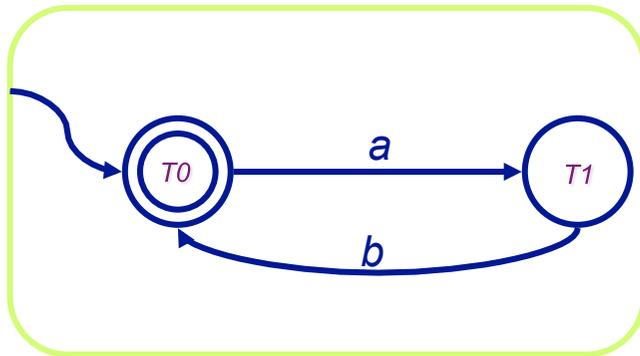


## Key points

- Services are **stateful**
- They share **atomic operations**
- They act over a **shared blackboard**
- **No available process for the target service**
- Must realize **target service** by **delegating** operation executions to **available services ...**
- **... by repurposing fragment of available services to realize the requested target service**

# Simple example of service composition without the shared blackboard

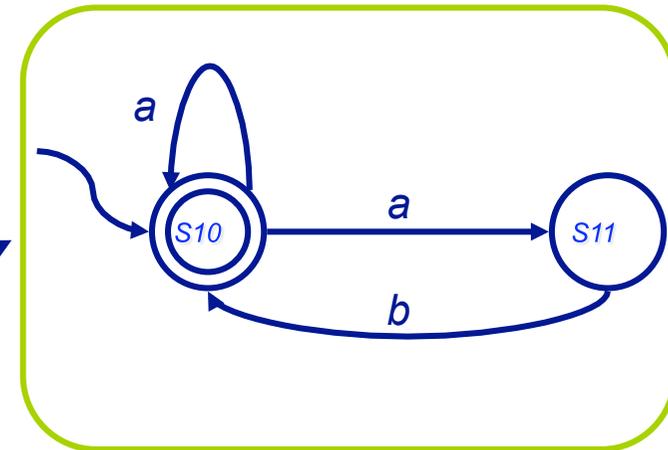
target service



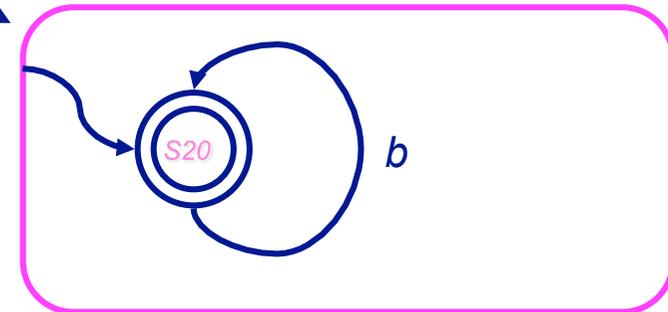
orchestrator

*Devilish nondeterminism!*

service 1



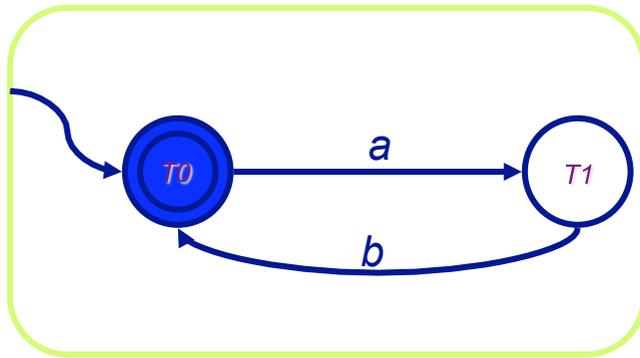
service 2



*For simplicity we don't consider blackboard for now.*

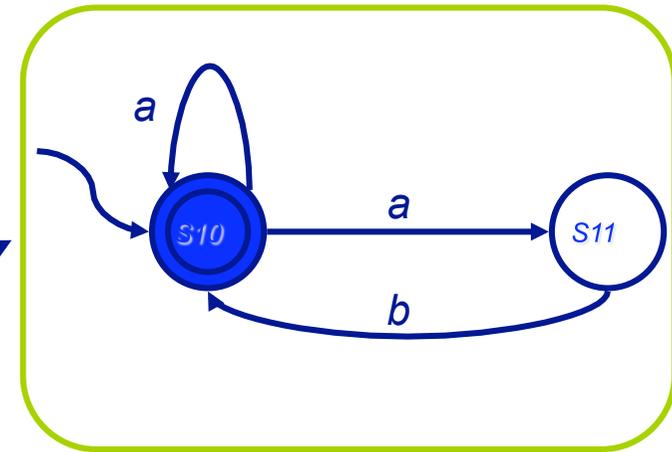
# Simple example of service composition

target service

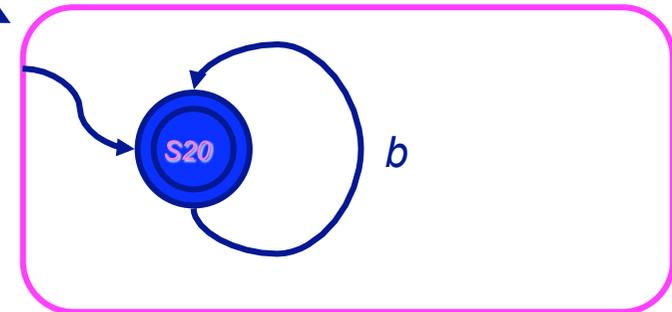


orchestrator

service 1

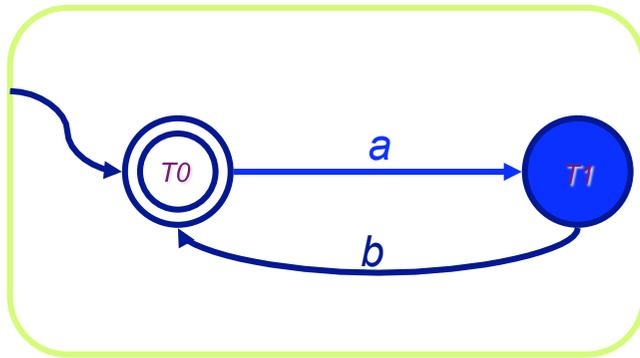


service 2



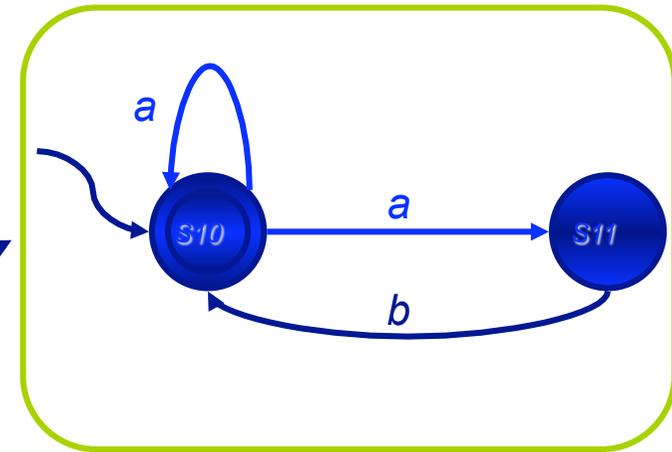
# Simple example of service composition

target service

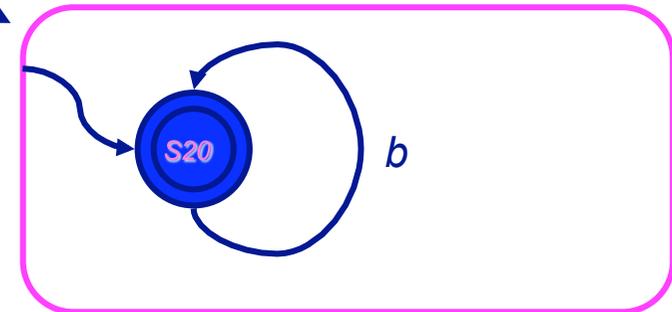


orchestrator

service 1

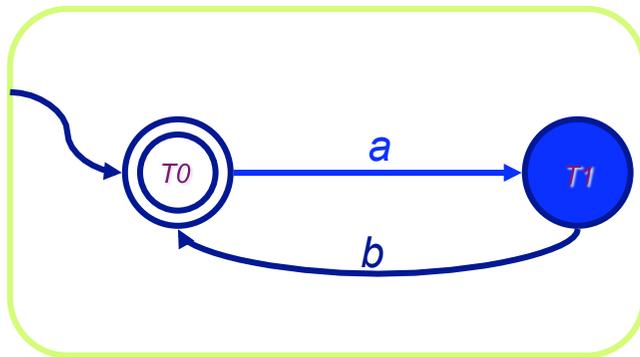


service 2



# Simple example of service composition

target service

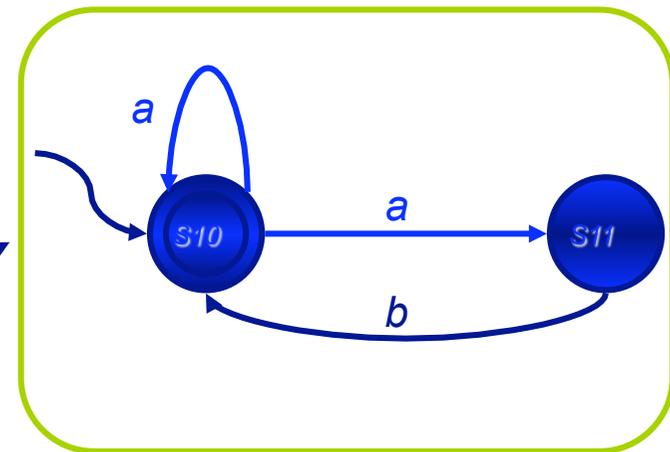


*observe the  
actual state!*

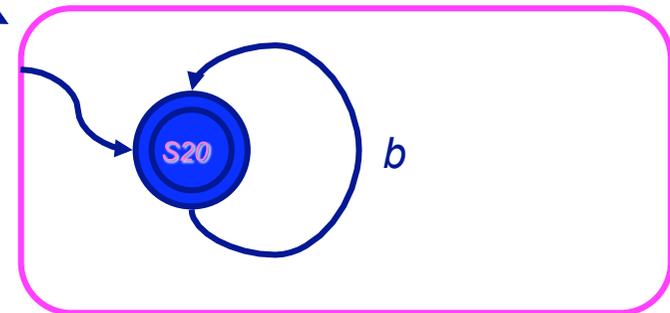


orchestrator

service 1

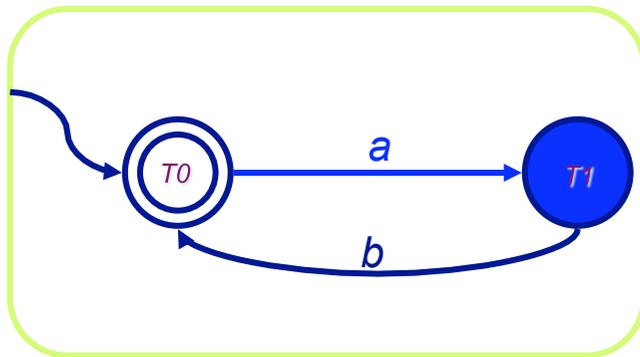


service 2



# Simple example of service composition

target service

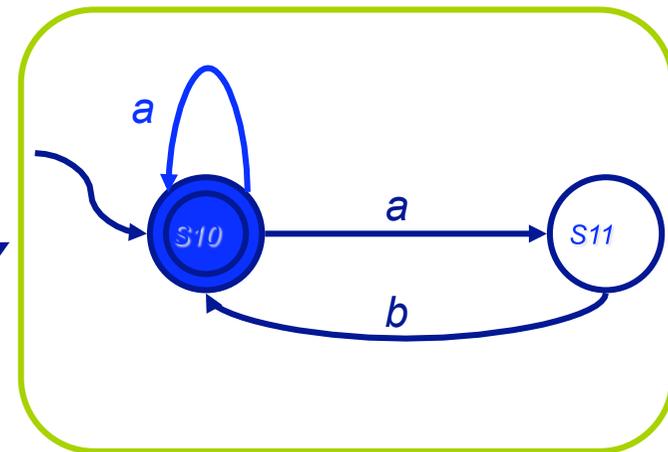


*observe the actual state!*

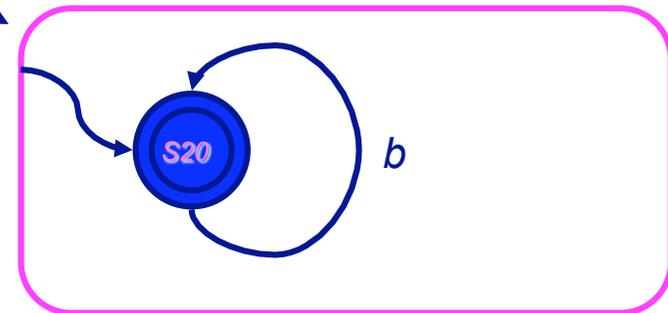


orchestrator

service 1

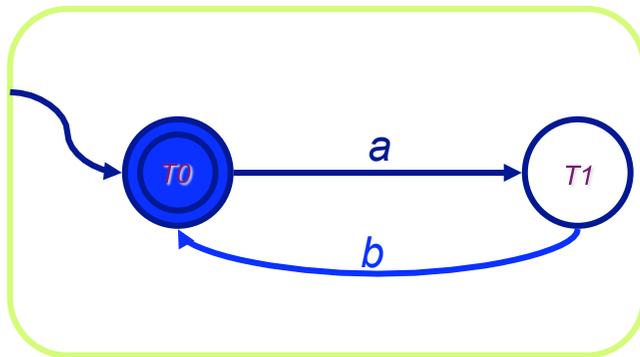


service 2



# Simple example of service composition

target service

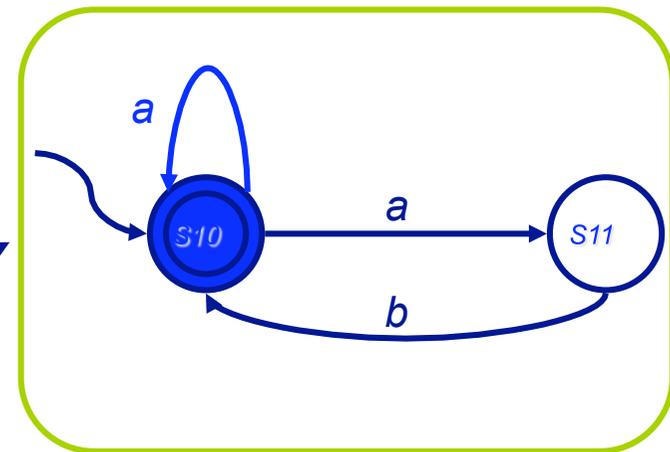


*observe the actual state!*

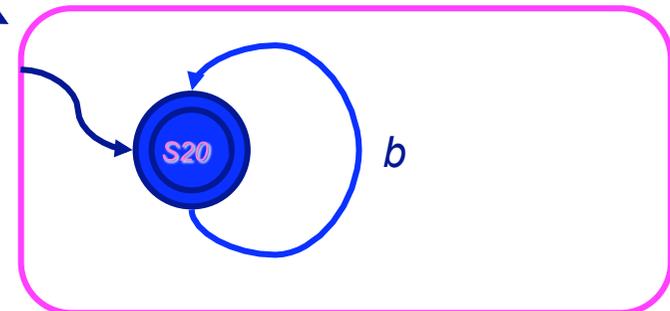


orchestrator

service 1

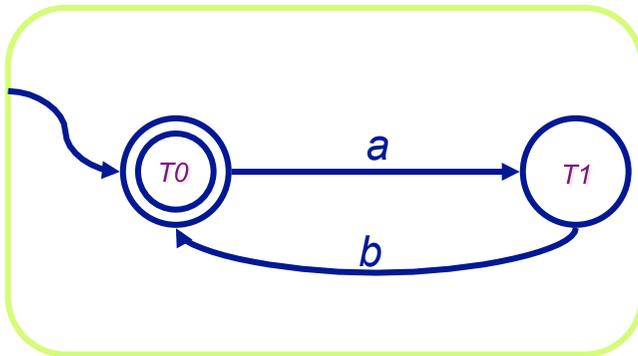


service 2

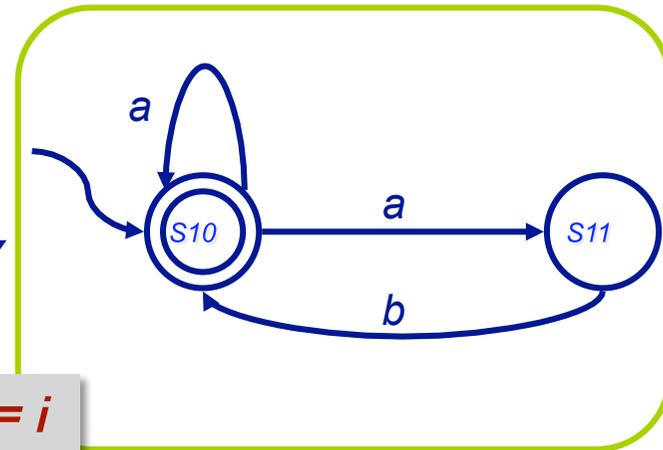


## Simple example of service composition

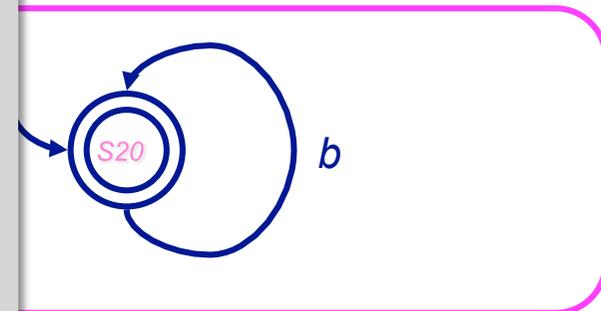
target service



service 1



service 2



- **Orchestrator program** is any function  $P(h,a) = i$  that takes a **history**  $h$  and an **action**  $a$  to execute and **delegates**  $a$  to one of the available services  $i$
- A **history** is a sequence that alternates states of the available services with actions performed:  

$$[s_1^0, s_2^0, \dots, s_n^0] a_1 [s_1^1, s_2^1, \dots, s_n^1] \dots a_k [s_k^1, s_2^k, \dots, s_n^k]$$
- Observe that to take a decision  $P$  has **full access to the past**, but no access to the future

## Synthesizing compositions

- Techniques for computing compositions:
- Reduction to PDL SAT
- Simulation-based 
- LTL synthesis as model checking of game structure

*(all techniques are for finite state services)*

## Simulation relation

Given a target service  $T$  and (the asynchronous product of) available services  $\mathcal{C}$ , a **(ND-)simulation** is a relation  $R$  between the states  $t \in \mathcal{T}$  and  $(s_1, \dots, s_n)$  of  $\mathcal{C}$  such that:

$(t, s_1, \dots, s_n) \in R$  implies that

for all  $t \rightarrow_a t'$  in  $T$ , exists a  $B_i \in \mathcal{C}$  s.t.

- $\exists s_i \rightarrow_a s'_i$  in  $B_i \wedge$
  - $\forall s_i \rightarrow_a s'_i$  in  $B_i \Rightarrow (t', s_1, \dots, s'_i, \dots, s_n) \in R$
- If **exists a simulation** relation  $R$  (such that  $(t^0, s_1^0, \dots, s_n^0) \in R$ , then we say that or **T is simulated by  $\mathcal{C}$**  (or  **$\mathcal{C}$  simulates T**).
  - **Simulated-by** is
    - (i) a simulation;
    - (ii) the largest simulation.

*Simulated-by is a coinductive definition*

## Using simulation for composition

- Given the largest simulation  $R$  of  $T$  by  $C$ , we can build every composition through the **orchestrator generator (OG)**.
- **OG** =  $\langle A, [1, \dots, n], S_r, s_r^0, \delta_r, \omega_r \rangle$  with
  - $A$  : the **actions** shared by the behaviors
  - $[1, \dots, n]$ : the **identifiers** of the available services in the community
  - $S_r = S_T \times S_1 \times \dots \times S_n$  : the **states** of the orchestrator generator
  - $s_r^0 = (t^0, s^0_1, \dots, s^0_n)$  : the **initial state** of the orchestrator generator
  - $\omega: S_r \times A_r \rightarrow 2^{[1, \dots, n]}$  : the **output function**, defined as follows:

$$\omega(t, s_1, \dots, s_n, a) = \{ i \mid \exists t' \rightarrow_a t' \text{ in } T \wedge \exists s_i \rightarrow_a s'_i \text{ in } B_i \wedge (t', s_1, \dots, s'_i, \dots, s_n) \in R \}$$

- $\delta \subseteq S_r \times A \times [1, \dots, n] \rightarrow S_r$  : the **state transition function**, defined as follows

$$(t, s_1, \dots, s_i, \dots, s_n) \xrightarrow{a, i} (t', s_1, \dots, s'_i, \dots, s_n) \text{ iff } i \in \omega(t, s_1, \dots, s_i, \dots, s_n, a)$$

# Adding data

*Adding data is crucial in certain contexts:*

- **Data** - rich description of the **static information** of interest.
- **Behaviors** - rich description of the **dynamics** of the process

*But makes the approach extremely challenging:*

- We get to work with infinite transition systems
- Simulation can still be used for capturing composition
- But it cannot be computed explicitly anymore.

*We are currently investigating two orthogonal approaches to deal with them.*

- *Based on SitCalc (see “Composition of ConGolog Programs” - IJCAI09 - next Wednesday, July 15)*
- *Based on “symbolic abstraction” (eg., the current paper)*

# Infinite-state shared blackboard

We consider a **shared blackboard**, where data can be added and removed.

- ▶ The blackboard is modeled as an **associative list**: set of pairs  $\langle attribute, value \rangle$
- ▶ The **maximal size** of the blackboard is fixed...
- ▶ ... but it can contain values an **infinite, ordered ( $\leq$ ) and dense (interpretation) domain  $\Delta$**  (e.g., alphanumeric strings).

Example of blackboard  $R$ :

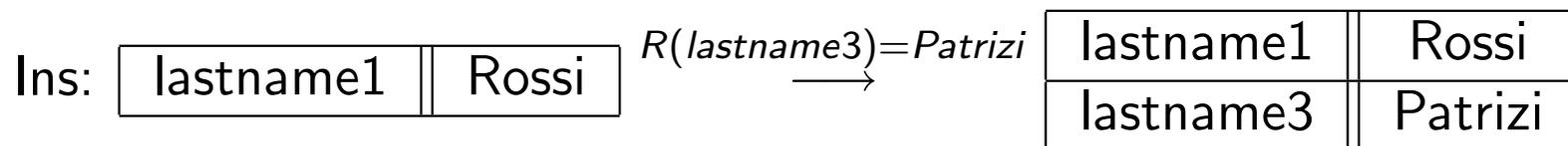
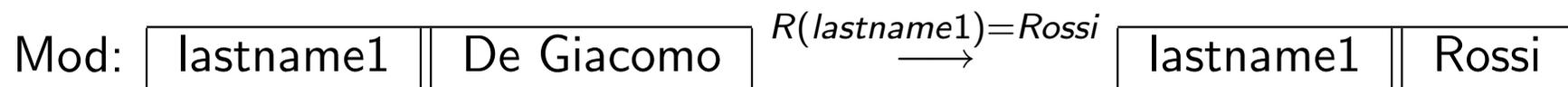
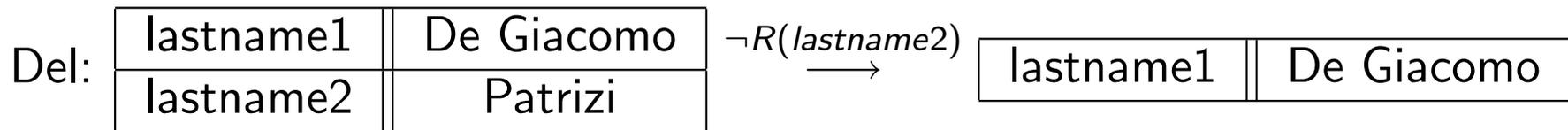
person1	Giuseppe De Giacomo
person2	Fabio Patrizi

The blackboard is a sort of *artifact*, see [Deutsch,Hull,Patrizi,Vianu-ICDT09]

# Atomic operations on the blackboard

- ▶ tuple insertion/modification:  $R(\chi) = v$
- ▶ tuple deletion:  $\neg R(\chi)$

## Examples



- ▶ Attributes can be added and removed
- ▶ Atomic operations can be arbitrarily concatenated

# Atomic operations on the blackboard (cont.d)

Operations with formal parameters:

$$o(q) = \{\langle \phi_1(q), \nu_1(q) \rangle, \dots, \langle \phi_m(q), \nu_m(q) \rangle\}$$

- ▶  $\phi_i(q)$ , **condition** over  $R, \Delta, \leq$   
e.g.:  $isDef(R(name)) \wedge R(name) \leq q \wedge q \leq R(name)$
- ▶  $\nu_i(q)$ , **sequence of atomic operations**  
e.g.:  $R(name) = q, \neg R(lastname), \dots$
- ▶ the **formal parameter**  $q$  is resolved with **actual parameter** given by the client at run time.

Successor relation:

$\bar{R} \xrightarrow{o, \bar{q}} \bar{R}'$  ( $q \in \Delta$ ) iff:

- ▶  $\exists \phi_i(q) \mid \langle \bar{R}, \leq \rangle \models \phi_i(\bar{q})$
- ▶  $\bar{R} \xrightarrow{\nu_i(\bar{q})} \bar{R}'$

- ▶ **Nondeterministic**: several  $\phi_i$ 's can be satisfied at the same time
- ▶ **Not input-bounded**: client can choose any value from  $\Delta$  as actual parameter
- ▶ For simplicity we use 1 parameter per operation in this talk

# Composition

Given:

- ▶ an initial state of the blackboard  $\bar{R}_0$
- ▶ a deterministic *target service*  $S_t$
- ▶ a set of  $n$  available *nondeterministic services*  $\{S_1, \dots, S_n\}$

Find a **composition**, i.e., a **simulation**  $S_t$  by the asynchronous product of  $S_1, \dots, S_n$   $\Sigma$ , such that  $\langle s_{t0}, \langle s_{10}, \dots, s_{n0} \rangle, \bar{R}_0 \rangle \in \Sigma$

As before, the core problem amounts to building a **simulation relation**.

# From infinite to finite states

**Objective:** build a finite abstraction on the an infinite blackboard configurations and adopt finite-state reasoning

- ▶ The blackboard is **infinite-state**
- ▶ But for every blackboard state  $\bar{R}$  we have  $|adom(\bar{R})| \leq b$
- ▶ We get a finite representation of the infinite-state system by **abstracting over actual values** in the blackboard.

# Abstracting over actual values

**Intuition:** since  $|adom(\bar{R})| \leq b \dots$

- ▶ replace  $adom(\bar{R})$  with a symbolic version  $ad\hat{om}(\bar{R}) = \{\hat{a}_1, \dots, \hat{a}_b\}$
- ▶ define a mapping  $m : adom(\bar{R}) \longrightarrow ad\hat{om}(\bar{R})$  which preserves  $\leq$  and  $\bar{R}$  (resp.  $\hat{\leq}$  and  $\hat{R}$ )

## Example

$$\bar{R} = \begin{array}{|c|c|} \hline 12 & 3 \\ \hline 1 & 15 \\ \hline 3 & 3 \\ \hline \end{array}$$

$$adom(\bar{R}) = \{1, 3, 12, 15\}$$

$$1 \leq 3 \leq 12 \leq 15$$

$$\begin{array}{l} m(1) = \hat{a}_3, \quad m(3) = \hat{a}_2 \\ m(12) = \hat{a}_1, \quad m(15) = \hat{a}_4 \end{array}$$

$$\longleftarrow m \longrightarrow$$

$$\hat{R} = \begin{array}{|c|c|} \hline \hat{a}_1 & \hat{a}_2 \\ \hline \hat{a}_3 & \hat{a}_4 \\ \hline \hat{a}_2 & \hat{a}_2 \\ \hline \end{array}$$

$$ad\hat{om}(\bar{R}) = \{\hat{a}_1, \hat{a}_2, \hat{a}_3, \hat{a}_4\}$$

$$a_3 \leq a_2 \leq a_1 \leq a_4$$

# Non-symbolic vs. symbolic simulation

- ▶ Q: What is the relation between (non-symbolic) simulation and symbolic simulation (the simulation performed on the symbolic abstraction)?
- ▶ A: **they are equivalent (!)**

## Theorem:

A (non-symbolic) simulation of the target service by the available services exists iff the symbolic simulation does.

**Finite-state techniques apply!**

*From the **orchestrator generator** associated to the symbolic simulation one easily extracts the orchestrator generator for the original (non-symbolic) setting.*

# Mixing data and service integration: A real challenge for the whole CS

We have all the issues of data integration but in addition ...

- Behavior: description of the **dynamics** of the process!
- Behavior should be formally and **abstractly** described: conceptual modeling of dynamics (not a la OWL-S). Which?
  - Workflows community may help
  - Business process community may help
  - Services community may help
  - Process algebras community may help
  - AI & Reasoning about actions community may help
  - DB community may help
  - ... may help
- Techniques for **analysis/synthesis** of **services** in presence of **unbounded data** can come from different communities:
  - Verification (CAV) community: abstraction to finite states
  - AI (KR) community: working directly in FOL/SOL, e.g., SitCalc