Composition: the "Roman" Approach
Composition via Simulation

Simulation

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

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

- A state s_0 of transition system S is **simulated by** a state t_0 of transition system T iff there **exists** a **simulation** between the initial states s_0 and t_0.

- Notably
  - **simulated-by** is a simulation
  - **simulated-by** is the largest simulation

Note it is a co-inductive definition!

- NB: A simulation is just one of the two directions of a bisimulation

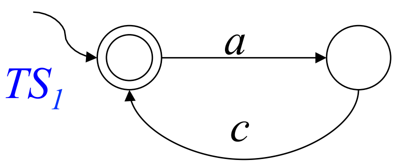
Computing Simulation on Finite Transition Systems

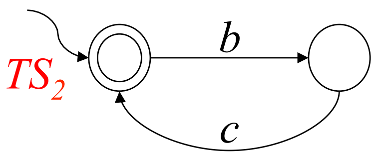
Algorithm ComputingSimulation
Input: transition system $TS_S = < A, S, S^0, \delta_S, F_S>$ and
transition system $TS_T = < A, T, T^0, \delta_T, F_T>$
Output: the **simulated-by** relation (the largest simulation)

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

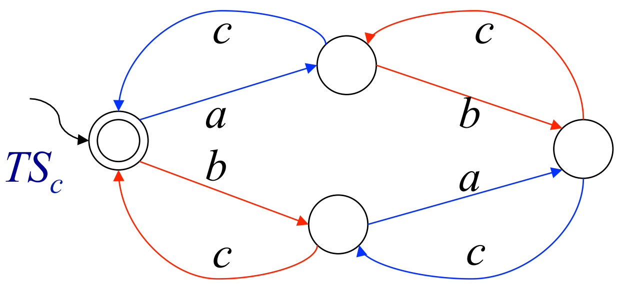
Potential Behavior of the Whole Community

- The potential behavior of the whole community is obtained by executing concurrently all TSs allowing for all possible interleaving (no synchronization).
- Available services:





- Resulting potential behavior described as a transition system TS_c



TS_c can be computed as the **asynchronous product** of TS_1 and TS_2.

Asynchronous Product of TSs (Community TS)

To compute the potential behavior of the community called Community TS we simply apply the asynchronous product

Let $TS_1, \cdots, TS_n$ be the TSs of the component services. The **asynchronous product** of $TS_1, \cdots, TS_n$, is defined as:
$TS_c = \langle A, S_c, S_c^0, \delta_c, F_c \rangle$ where:

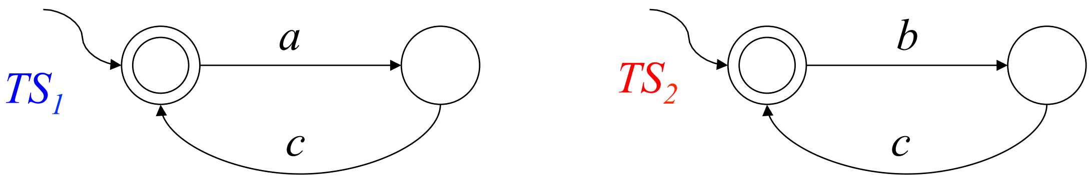
- A is the set of actions
- $S_c = S_1 \times \cdots \times S_n$
- $S_c^0 = \{(s^0_1, \cdots, s^0_n)\}$
- $F \subseteq F_1 \times \cdots \times F_n$
- $\delta_c \subseteq S_c \times A \times S_c$ is defined as follows:
  $(s_1, \cdots, s_n) \rightarrow_a (s'_1, \cdots, s'_n)$ iff
  1. $\exists$ i. $s_i \rightarrow_a s'_i \in \delta_i$
  2. $\forall$ j≠i. $s'_j = s_j$

Composition via Simulation

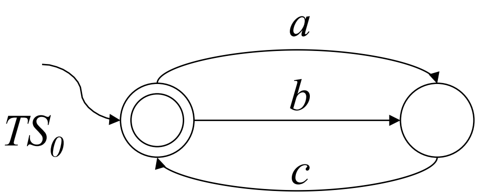
- **Thm[IJFCS08]**
  A composition realizing a target service TS TS_t exists if there **exists** a simulation relation between the initial state s_t^0 of TS_t and the initial state $(s_1^0, .., s_n^0)$ of the community TS TS_c.
- Notice if we take the union of all simulation relations then we get the largest simulation relation ***S***, still satisfying the above condition.
- **Corollary[IJFCS08]**
  A composition realizing a target service TS TS_t exists iff $(s_t^0, (s_1^0, .., s_n^0)) \in$ ***S***.
- **Thm[IJFCS08]**
  Computing the largest simulation ***S*** is polynomial in the size of the target service TS and the size of the community TS…
- ... hence it is EXPTIME in the size of the available services.

Example of Composition

- Available Services



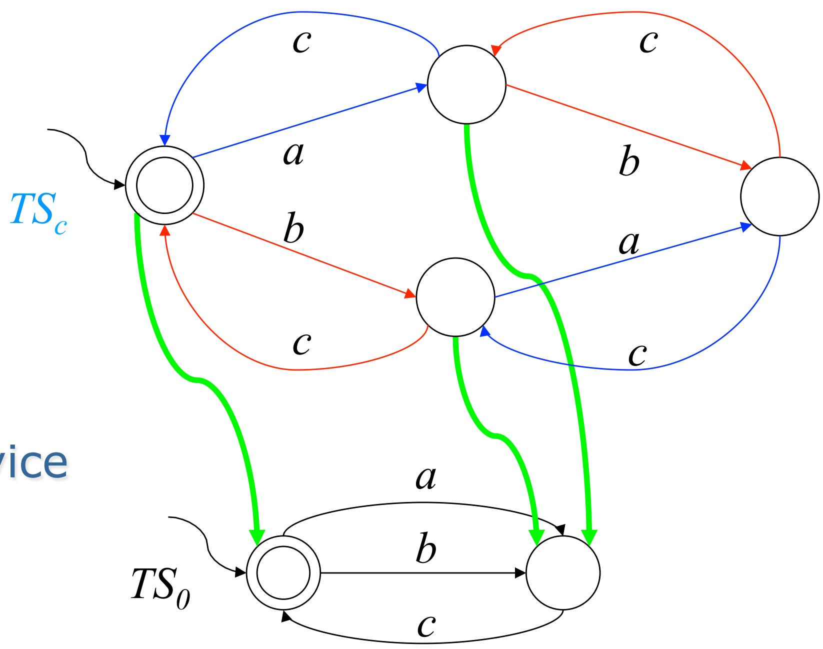
- Target Service



Example of Composition

Community TS

Target Service



Composition exists!

Orchestrator Generator

- Given the largest simulation ***S*** form TS_t to TS_c (which include the initial states), we can build the ***orchestrator generator***.

- This is an orchestrator program that can change its behavior reacting to the information acquired at run-time.

- Def: $OG = \langle A, [1,\ldots,n], S_r, s_r^0, \omega_r, \delta_r, F_r \rangle$ with
  - A : the **actions** shared by the community
  - [1,…,n]: the **identifiers** of the available services in the community
  - $S_r = S_t \times S_1 \times \cdots \times S_n$: the **states** of the orchestrator program
  - $s_r^0 = (s^0_t, s^0_1, \ldots, s^0_m)$: the **initial state** of the orchestrator program
  - $F_r \subseteq \{ (s_t, s_1, \ldots, s_n) \mid s_t \in F_t$: the **final states** of the orchestrator program
  - $\omega_r : S_r \times A_r \to [1,\ldots,n]$: the **service selection function**, defined as follows:

    $\omega_r(t, s_1,..,s_n, a)$ = { i | TS_t and TS_i can do a and remain in *S*}

    i.e., …= $\{i \mid s_t \to_{a,} s'_t \wedge \exists s_i'. \, s_i \to_{a,} s_i' \wedge (s_t', (s_1, \ldots, s'_i, \ldots, s_n)) \in S\}$

  - $\delta_r \subseteq S_r \times A_r \times [1,\ldots,n] \to S_r$: the **state transition function**, defined as follows:
    Let $k \in \omega_r(s_t, s_1, \ldots, s_k, \ldots, s_n, a)$ then
    $(s_t, s_1, \ldots, s_k, \ldots, s_n) \to_{a,k} (s_t', s_1, \ldots, s'_k, \ldots, s_n)$ where $s_k \to_{a,} s'_k$

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

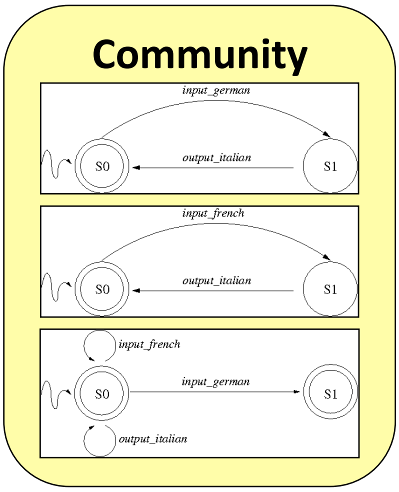
Orchestrator Generator

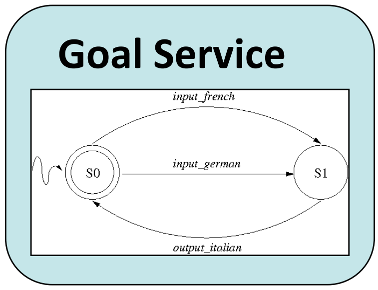
- For generating OG we need only to compute ***S*** and then apply the template above

- For running an orchestrator from the OG we need to store and access ***S*** *(polynomial time, exponential space)* …
- … and compute ω_r and δ_r at each step *(polynomial time and space)*

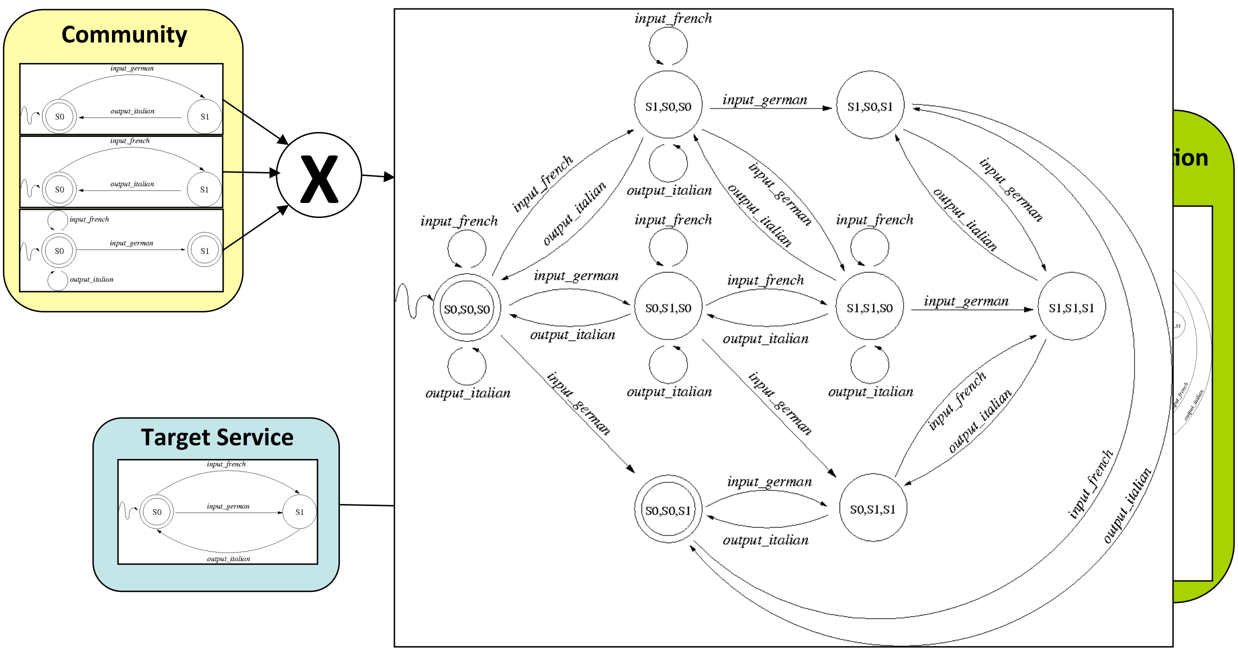
Example of composition via simulation (1)

- A *Community* of *services* over a shared alphabet $\mathcal{A}$
- A (Virtual) *Goal* service over $\mathcal{A}$

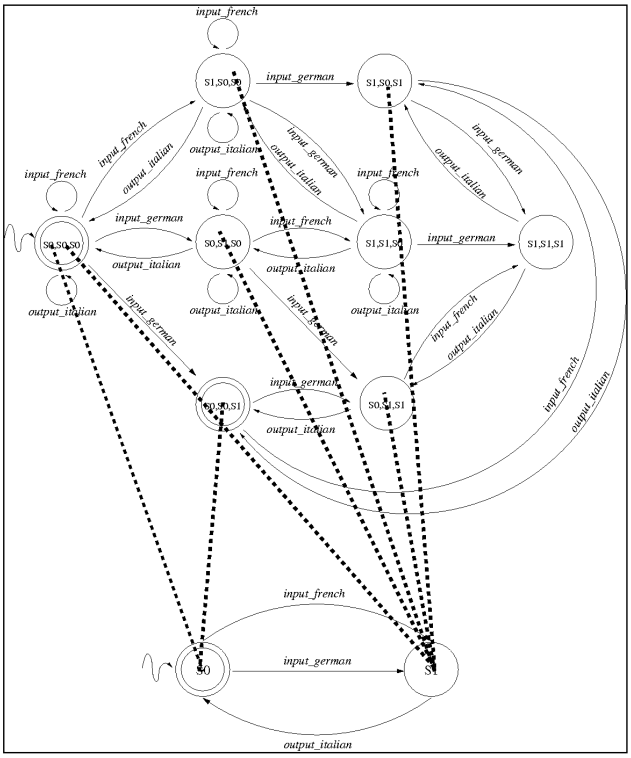




Example of composition via simulation (2)



Example of composition via simulation (3)



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

Example of composition via simulation (4)