Composition: the "Roman" Approach

Composition via ND-Simulation for Nondeterministic Available Services

Service composition

Problem of composition existence

- Given:
  - available services $B_1,\ldots,B_n$
  - target service T

  over the same environment (same set of atomic actions)
- Check whether T can be realized by **delegating** actions to $B_1,\ldots,B_n$ so as to ***mimic*** T over time *(forever!)*

Composition synthesis

*synthesis of the **orchestrator** that does the delegation*

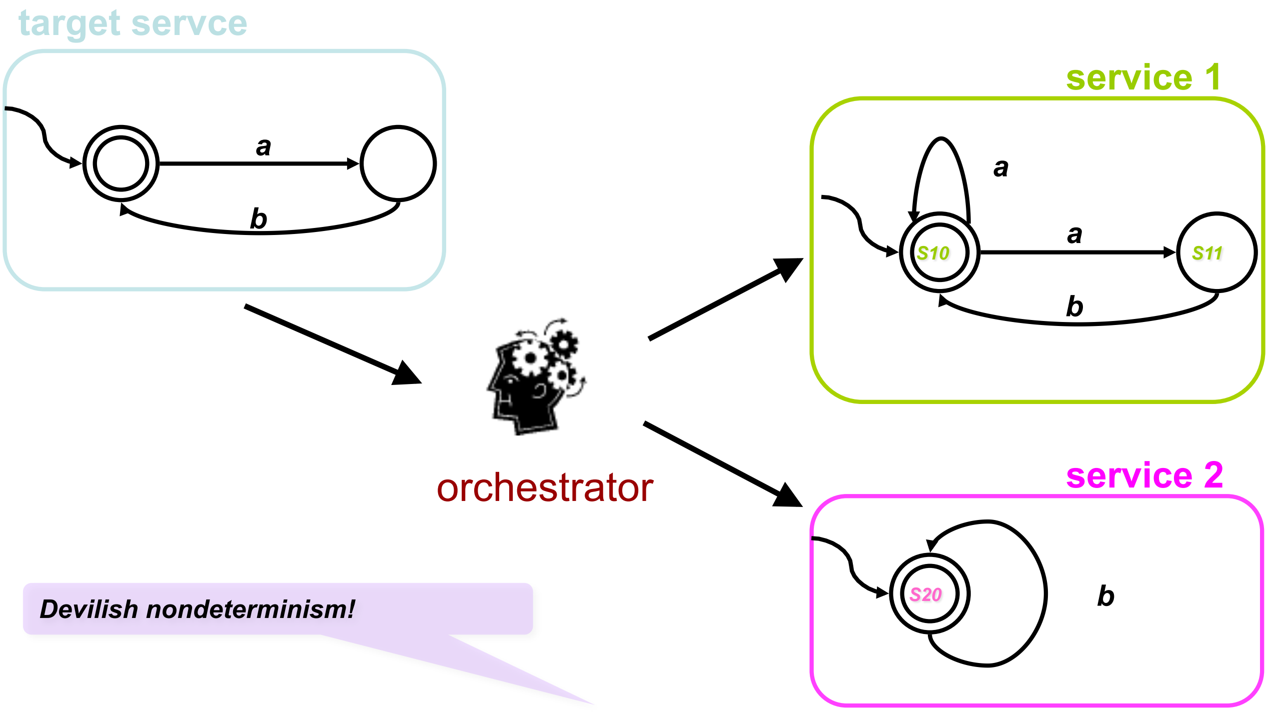
Nondeterminism in Available Services

Devilish (don't know)!

- Nondeterministic available services
  - **Incomplete information** on the actual **behavior**
  - **Mismatch between behavior description** (which is in terms of the environment actions) and **actual behavior** of the agents/devices
- Deterministic target service
  - it's a spec of a desired service: (devilish) nondeterminism is banned

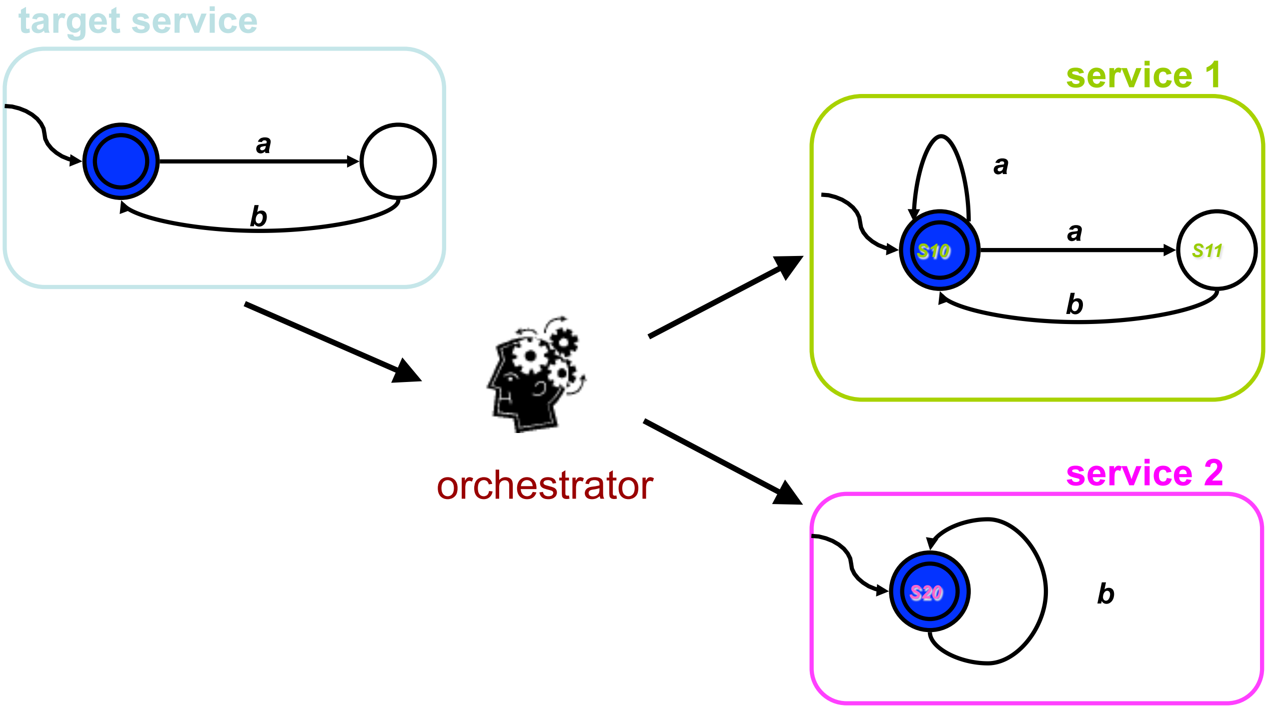
In general, devilish nondeterminism difficult to cope with eg. nondeterminism moves AI Planning from PSPACE (classical planning) to EXPTIME (contingent planning with full observability [Rintanen04])

Example: Nondeterministic Available Services

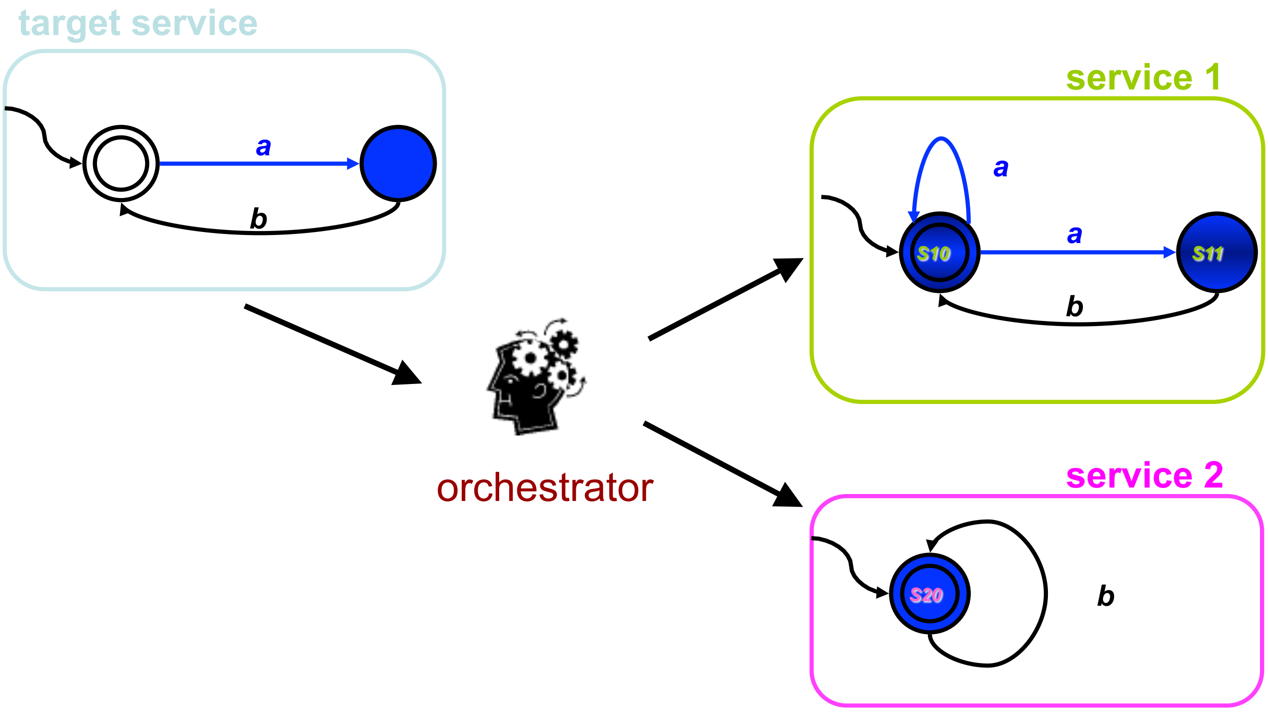


Available services represented as nondeterministic transition systems

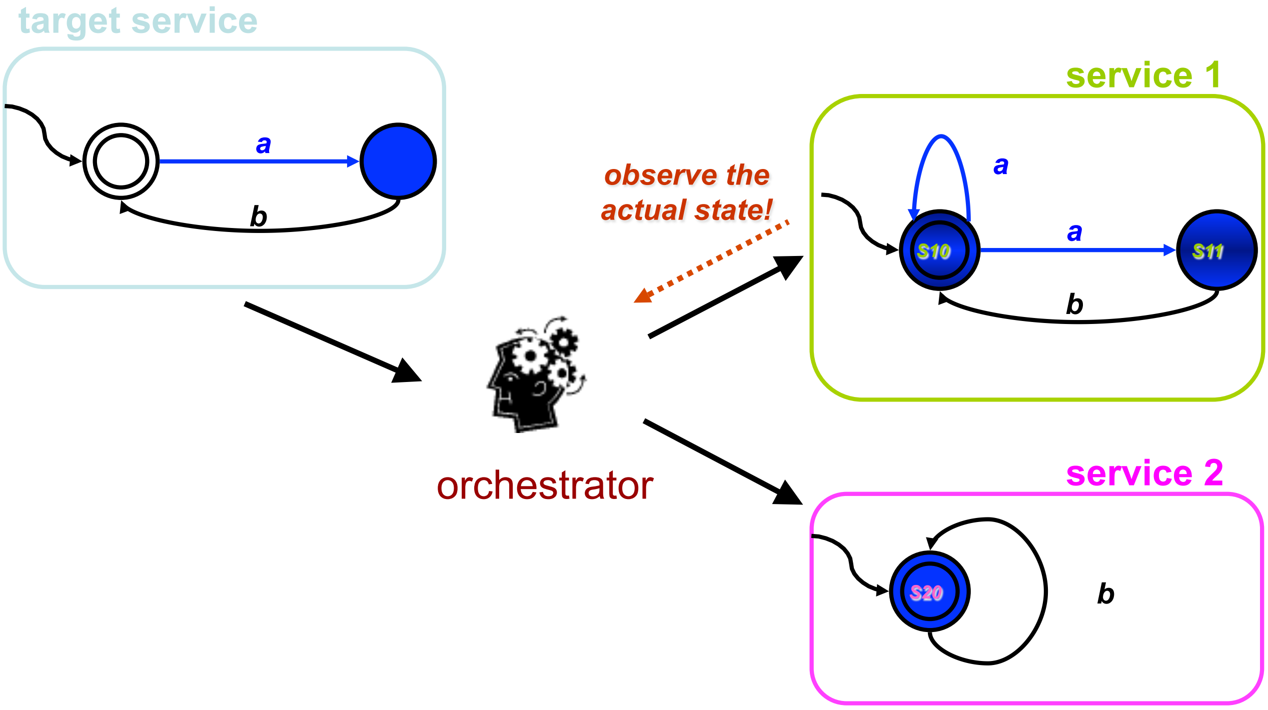
Example: Nondeterministic Available Services



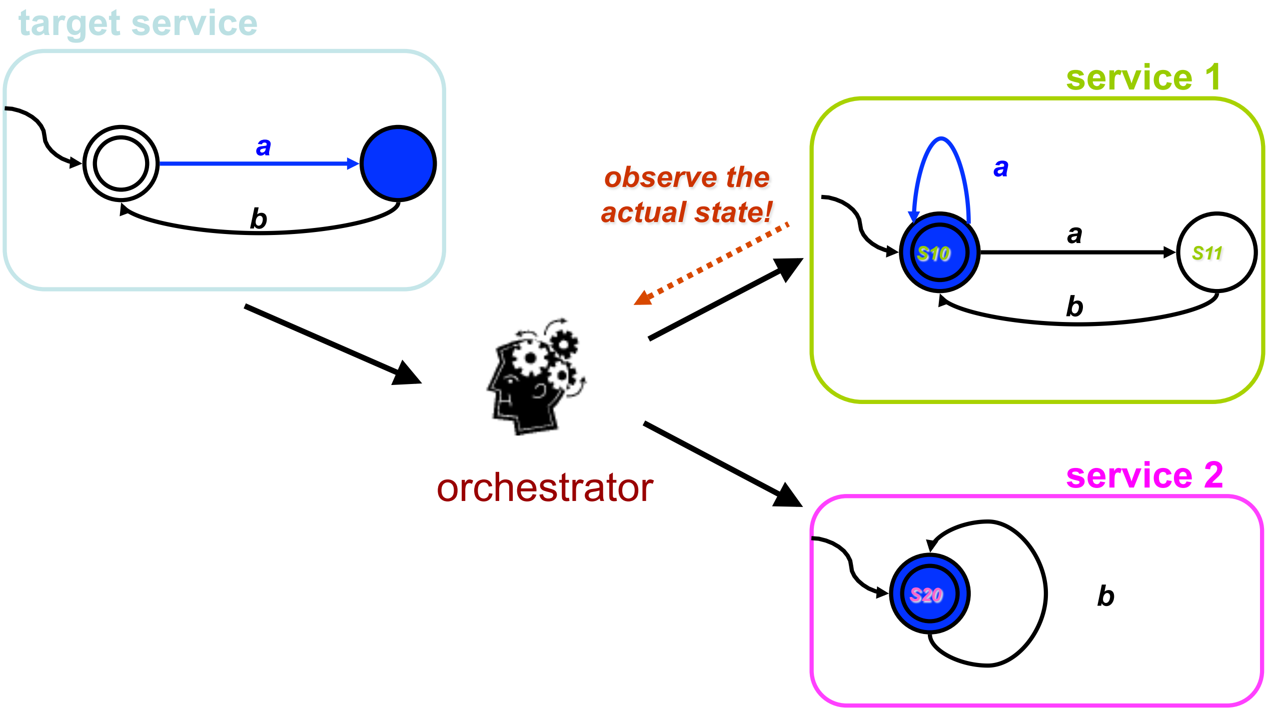
Example: Nondeterministic Available Services



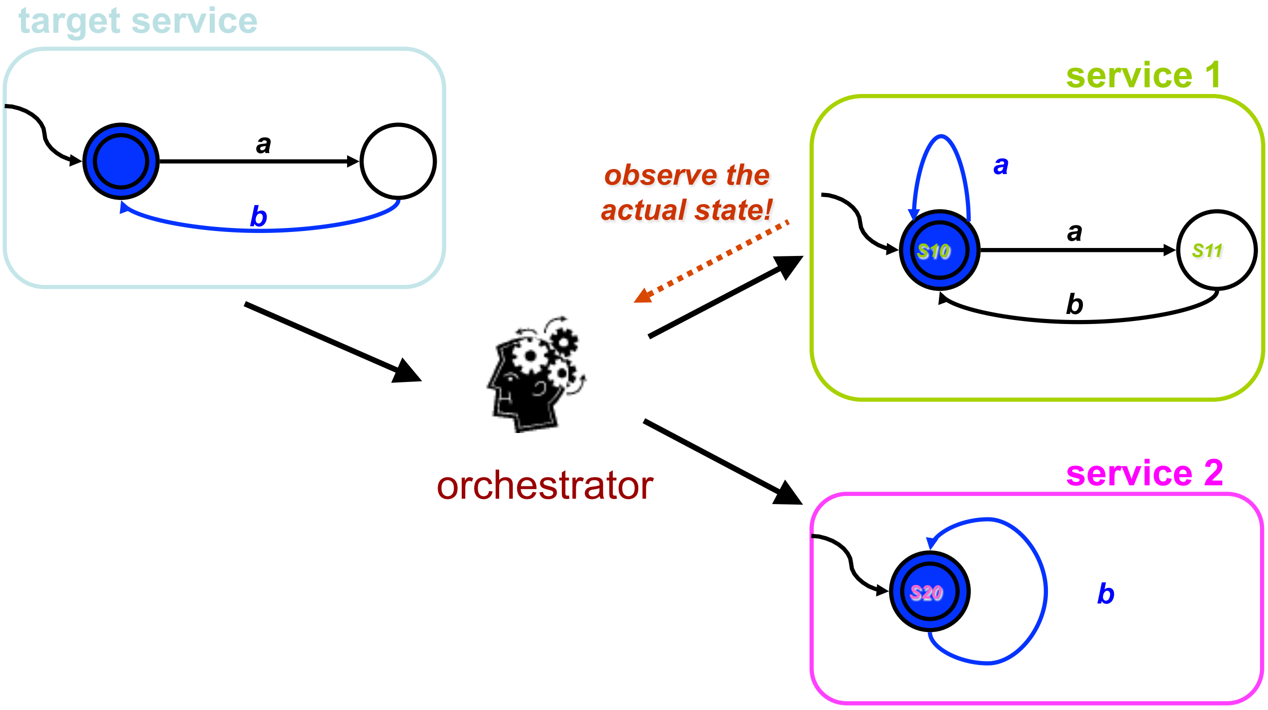
Example: Nondeterministic Available Services



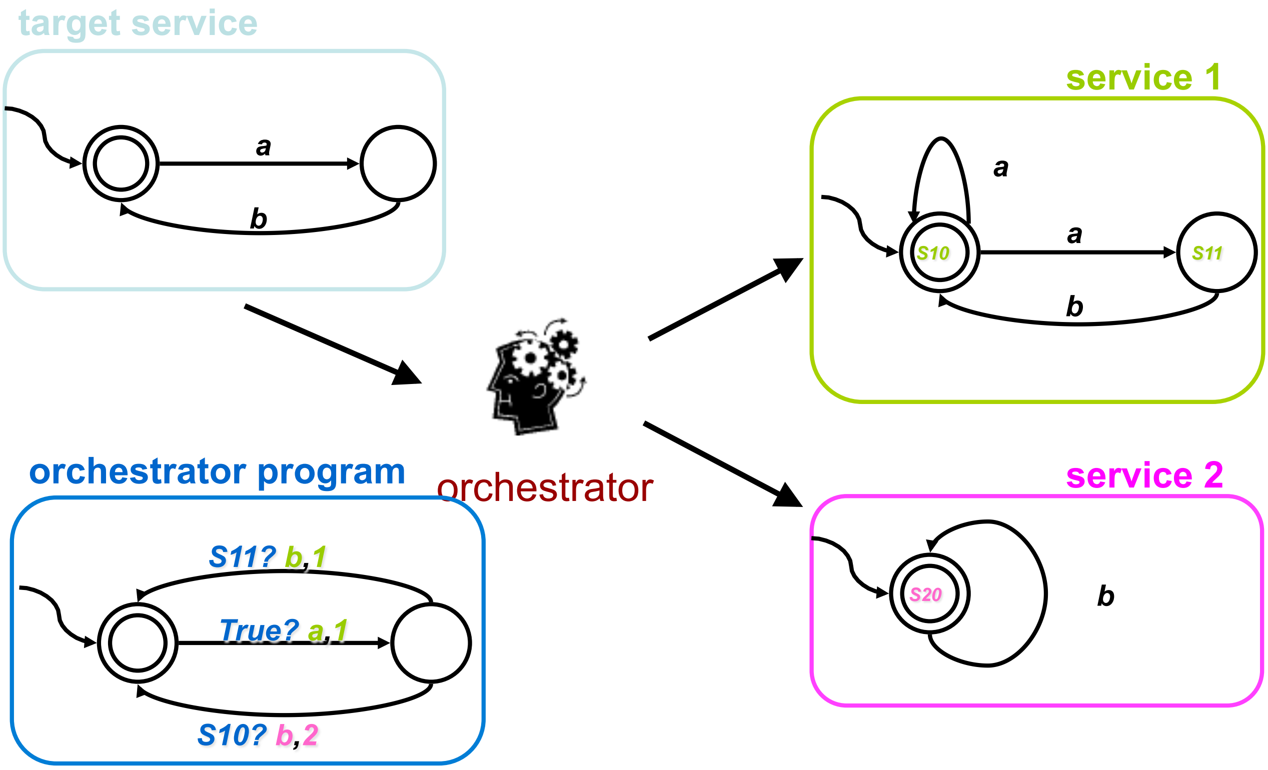
Example: Nondeterministic Available Services



Example: Nondeterministic Available Services



An Orchestrator Program Realizing the Target Service



Orchestrator Programs

contains all the observable information up the current situation

- **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 of the form, which alternate 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
- *Problem: synthesize a orchestrator program P that realizes the target service making use of the available services*

Technical Results: Theoretical

Thm[IJCAI'07] Checking the existence of orchestrator program realizing the target service is **EXPTIME-complete**.

Polynomial Reduction to PDL SAT
EXPTIME-hardness due to Muscholl&Walukiewicz07 for deterministic services

Thm [IJCAI'07] If a **orchestrator program exists** there exists one that is **finite state**.

Exploits the finite model property of PDL

Note: same results as for deterministic services!

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)

Composition via ND-Simulation

Directly based on

... *orchestrating* the concurrent execution of available services $B_1,\ldots,B_n$ so as to ***mimic*** the target service T

Thm: *Composition exists iff the asynchronous (Cartesian) product C of $B_1,\ldots,B_n$ can* ***(ND-)simulate*** *T*

Composition via ND-Simulation

- We consider binary relations R satisfying the following co-inductive condition (ND-similarity):

  $(t,(s_1, .., s_n)) \in R$ implies that
  - if t is *final* then s_i, with i=1, .., n, is *final*
  - for **all** actions a
    - If $t \rightarrow_a t'$ then $\exists$ k $\in$ 1..n.
      - $\exists s_k' . s_k \rightarrow_a s_k'$
      - $\forall s_k' . s_k \rightarrow_a s_k' \supset (t',(s_1,..,s_k', .., s_n)) \in R$

  Note similar in the spirit to simulation relation!
  But more involved, since it deals with

  - *the existential choice (as the simulation) of the service, and*
  - *the universal condition on the nondeterministic branches!*

- A composition realizing a target service TS TS_t exists if there **exists** a relation R satisfying the above condition between the initial state t^0 of TS_t and the initial state $(s_1^0, .., s_n^0)$ of the community big TS TS_c.
- Notice if we take the union of all such relation R then we get the largest relation ***RR*** satisfying the above condition.
- A composition realizing a target service TS T exists iff $(t^0 , (s_1^0, .., s_n^0)) \in$ ***RR***.

Algorithm for ND-simulation

Algorithm Compute (ND-)simulation
Input: target service $T = \langle A, S_T, t^0, \delta_T, F_T\rangle$ and ..
available services $\mathcal{S}_i = \langle A, S_i, s_i^0, \delta_i, F_i\rangle$, i = 1,..,n
Output: the **simulated-by** relation ***RR*** (the largest simulation)

Body

$R = \emptyset$
$R' = S_T \times S_1 \times .. \times S_n$
while $(R \neq R')$ {
  $R := R'$
  $R' := R' - \{(t, s_1,..,s_n) \mid \exists\, t \rightarrow_a t'$ in $T \wedge \neg\exists\, k = 1,..,n$ s.t.
  $(\exists\, s_k \rightarrow_a s_k' \wedge \forall\, s_k \rightarrow_a s'_k \supset (t', s_1,..,s'_k,..,s_n) \in R')\}$
}
return R'
End

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

Composition via ND-Simulation

- Given the maximal ND-simulation ***RR*** form TS_t to TS_c (which includes 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: $OG = \langle A, [1,\dots,n], S_r, s_r^0, \omega_r, \delta_r, F_r \rangle$ 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
  - $s_r^0 = (s^0_t, s^0_1, \dots, s^0_m)$: the **initial state** of the orchestrator program
  - $F_r \subseteq \{ (s_t, s_1, \dots, s_n) \mid s_t \in F_t$: the **final states** of the orchestrator program
  - $\omega_r : S_r \times A_r \to [1,\dots,n]$: the **service selection function**, defined as follows:

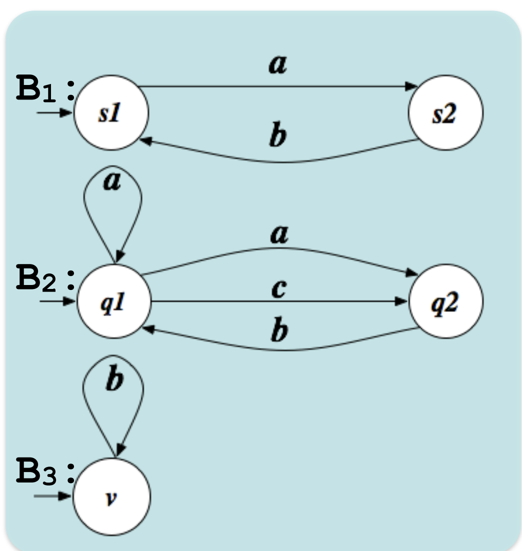
    $\omega_r(t, s_1,..,s_n, a)$ = { i | TS_t and TS_i can do a and remain in RR}

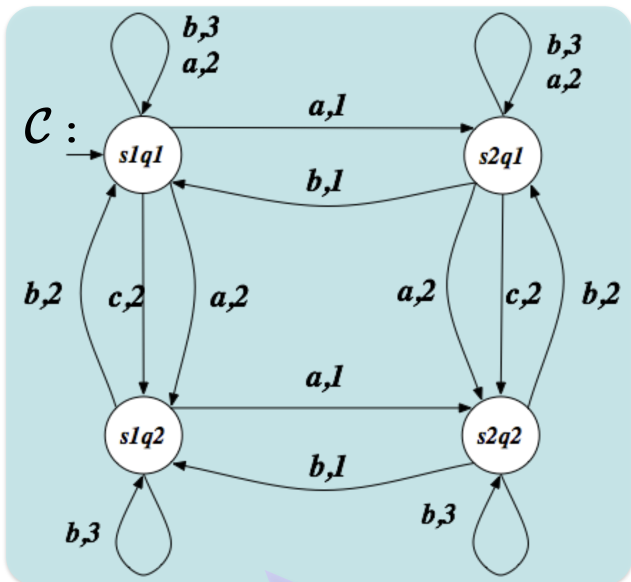
    i.e. …$=\{i \mid s_t \to_a s'_t \wedge \exists\, s_i'.\ s_i \to_a s_i' \wedge \forall\, s_i'.\ s_i \to_a s_i' \supset (s_t', (s_1, \dots, s_i', \dots, s_n)) \in RR\}$
  - $\delta_r \subseteq S_r \times A_r \times [1,\dots,n] \times S_r$: the **state transition relation**, defined as follows:
    - Let $k \in \omega_r(s_t, s_1, \dots, s_k, \dots, s_n, a)$ then
      $(s_t, s_1, \dots, s_k, \dots, s_n) \to_{a,k} (s_t', s_1, \dots, s'_k, \dots, s_n)$ for each $s_k \to_a s'_k$

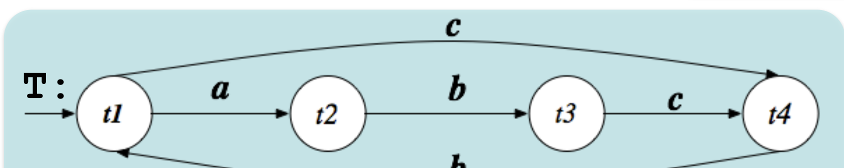
Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
SAPIENZA Università di Roma

Example of composition by ND-simulation:

1. compute asynchronous product of available services





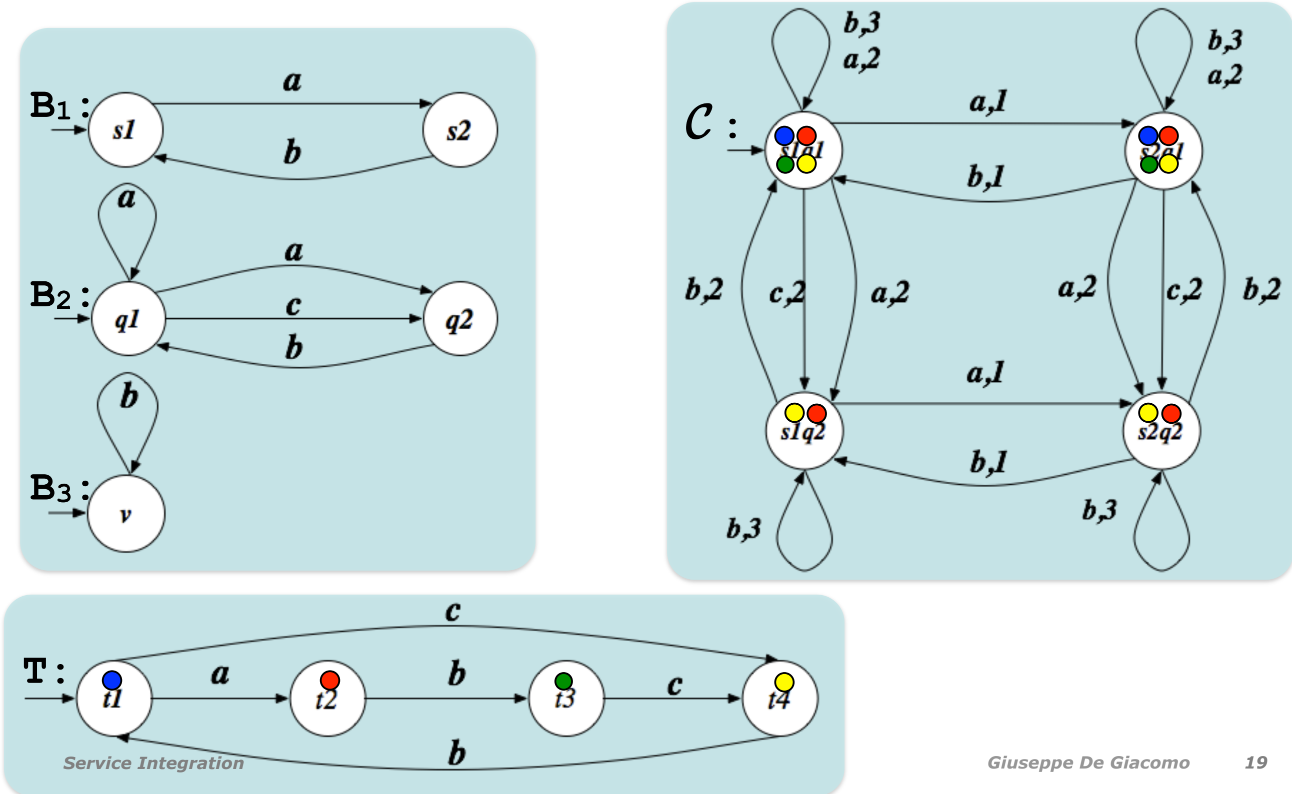


NB: asynchronous product construction can be virtual

Example of composition by ND-simulation:

2. compute ND-simulation (of T by C)

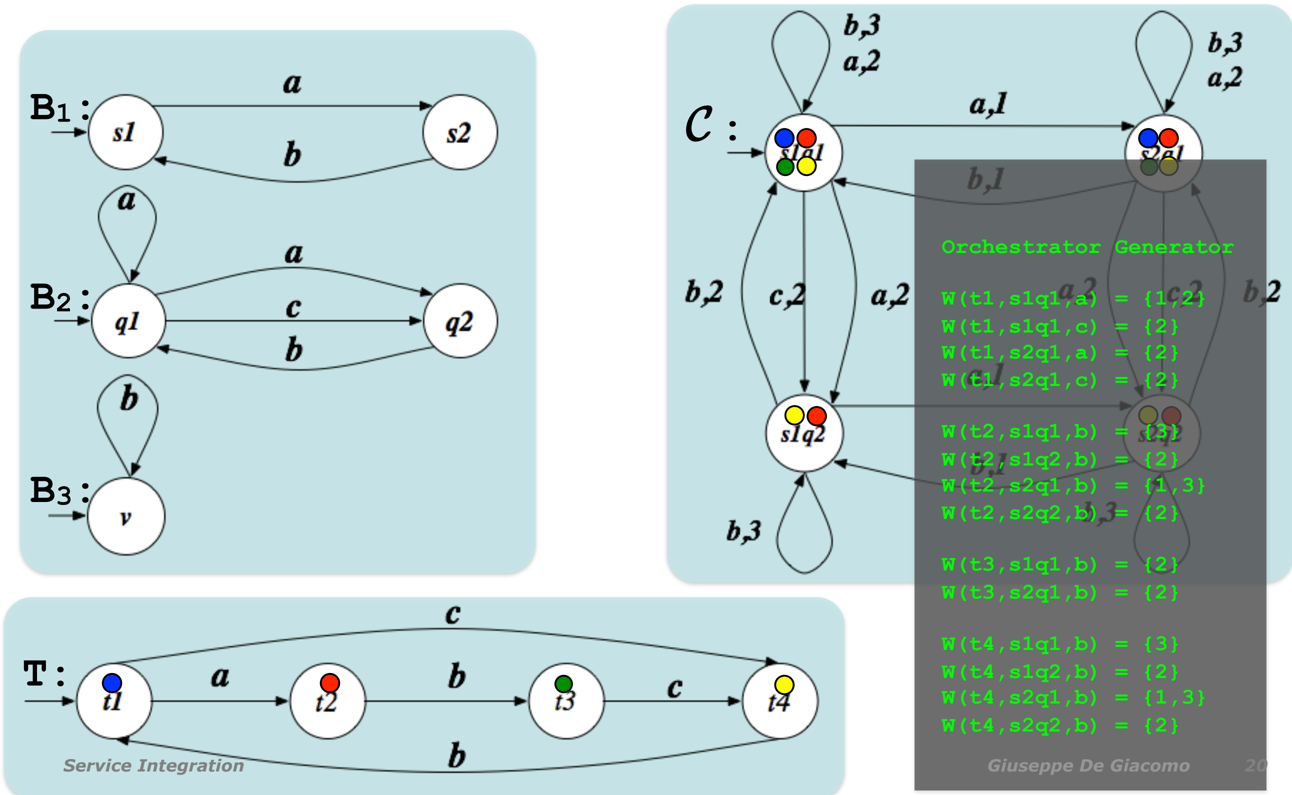
Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
SAPIENZA Università di Roma



Example of composition by ND-simulation:

3. compute orchestrator generator from the ND-simulation

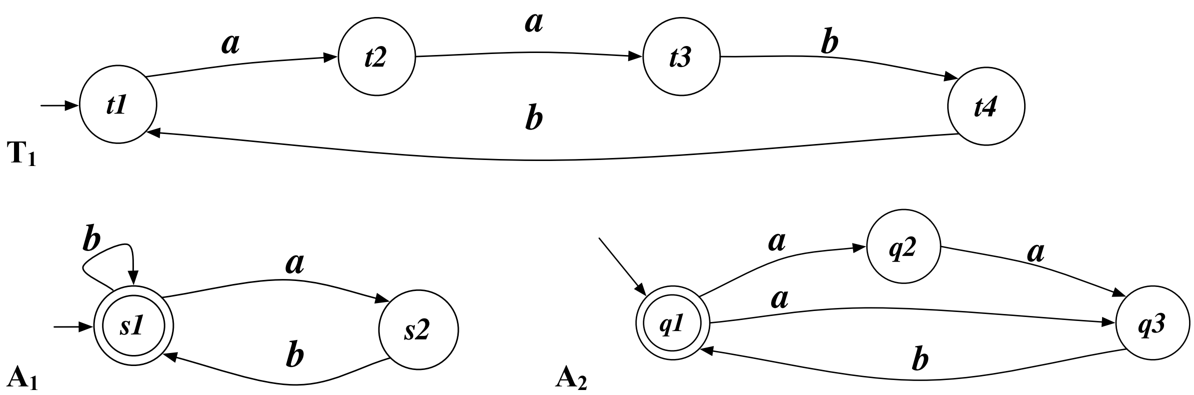
Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
SAPIENZA Università di Roma




```
Orchestrator Generator

W(t1,s1q1,a) = {1,2}
W(t1,s1q1,c) = {2}
W(t1,s2q1,a) = {2}
W(t1,s2q1,c) = {2}

W(t2,s1q1,b) = {3}
W(t2,s1q2,b) = {2}
W(t2,s2q1,b) = {1,3}
W(t2,s2q2,b) = {2}

W(t3,s1q1,b) = {2}
W(t3,s2q1,b) = {2}

W(t4,s1q1,b) = {3}
W(t4,s1q2,b) = {2}
W(t4,s2q1,b) = {1,3}
W(t4,s2q2,b) = {2}
```

Exercises

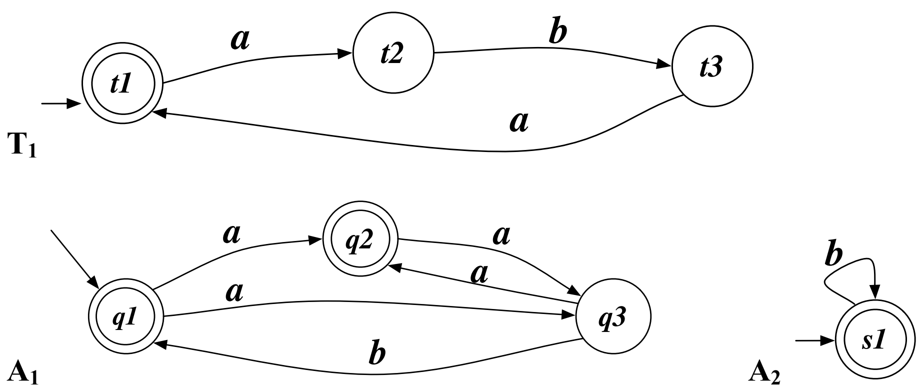
Given the following the available services A_1 and A_2 and the target service T_1, check whether a composition realizing it exists, and if it does, produce the output relation of orchestrator generator. If not, single out the target state that cannot be simulated (ND-simulated), and propose a change to the available services so as to guarantee the composition.



Exercises

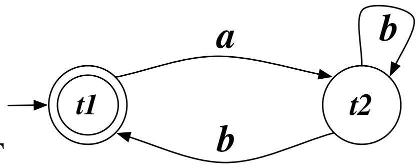
Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
SAPIENZA Università di Roma

Given the following the available services A_1 and A_2 and the target service T_1, check whether a composition realizing it exists, and if it does, produce the output relation of orchestrator generator. If not, single out the target state that cannot be simulated (ND-simulated), and propose a change to the available services so as to guarantee the composition.

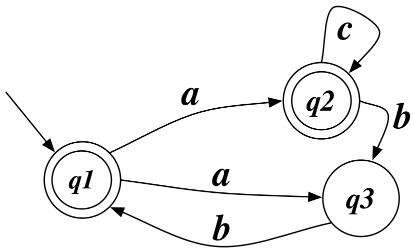


Exercises

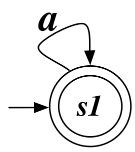
Given the following the available services $\mathbf{A_1}$ and $\mathbf{A_2}$ and the target service **T**, check whether a composition realizing it exists, and if it does, produce the output relation of orchestrator generator. If not, single out the target state that cannot be (ND-)simulated, and propose a change to the available services so as to guarantee the composition.



T



$\mathbf{A_1}$

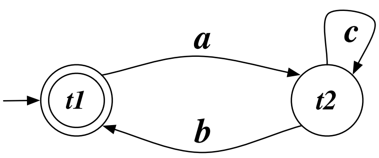


$\mathbf{A_2}$

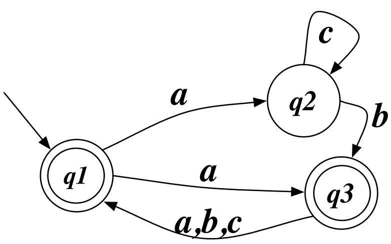
Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
SAPIENZA Università di Roma

Exercise

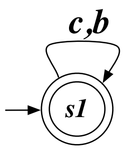
Given the following the available services $\mathbf{A_1}$ and $\mathbf{A_2}$ and the target service **T**, check whether a composition realizing it exists, and if it does, produce the output relation of orchestrator generator. If not, single out the target state that cannot be (ND-)simulated, and propose a change to the available services so as to guarantee the composition.



T



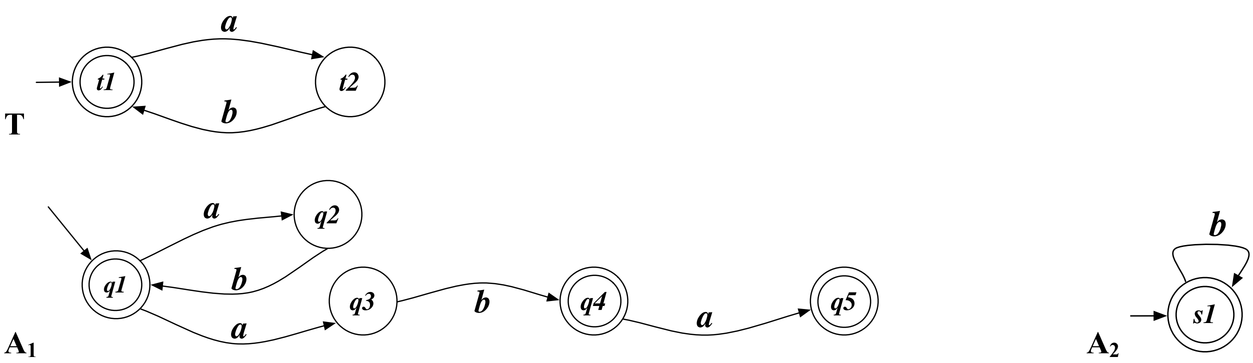
$\mathbf{A_1}$



$\mathbf{A_2}$

Exercises

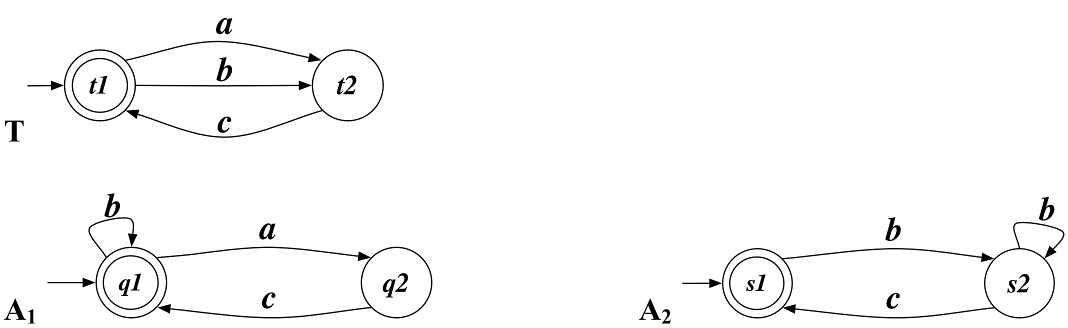
Given the following the available services $\mathbf{A_1}$ and $\mathbf{A_2}$ and the target service **T**, prove whether a composition realizing it exists, and, if it does, produce the output relation of orchestrator generator. If not, single out the target state that cannot be (ND-)simulated, and propose a small change to the available services so as to guarantee the composition.



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

Exercises

Given the following the available services $\mathbf{A_1}$ and $\mathbf{A_2}$ and the target service **T**, prove whether a composition realizing it exists, and, if it does, produce the output relation of orchestrator generator. If not, single out the target state that cannot be (ND-)simulated, and propose a small change to the available services so as to guarantee the composition and show that the composition now exists.



Composition ND-Simulation

- Computing ***RR*** is polynomial in the size of the target service TS and the size of the community TS…
- ... composition can be done in EXPTIME in the size of the available services

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

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

Results

- ***Thm:*** *choosing at each point any value in returned by the orchestrator generator gives us a composition.*

- ***Thm:*** *every composition can be obtained by choosing, at each point a suitable value among those returned by the orchestrator generator.*

*Note: there **infinitely many compositions** but only **one orchestrator generator** that captures them all*

- ***Thm:*** *computing the orchestrator generator is EXPTIME, and in fact exponential only in the number (and not the size) of the available behaviors.*

Composition in the Roman Model was shown to be EXPTIME-hard
[Muscholl&Walukiewicz07]

Just-in-time composition

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

- Once we have the orchestrator generator ...
- ... we can **avoid choosing any particular composition** apriori ...
- ... and **use directly ω_r** to choose the available behavior to which delegate the next action.
- We can be ***lazy*** and make such choice ***just-in-time***, possibly adapting reactively to ***runtime*** feedback.

Failures

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

- Available services may become unexpectedly unavailable for various reasons. We consider 4 kinds of behavioral failures:
  1. A service **temporarily freezes**; it will eventually resume in the same state it was in;
  2. A service unexpectedly and arbitrarily (i.e., without respecting its transition relation) **changes its current state**;
  3. A **service dies**; that is, it becomes permanently unavailable;
  4. A dead service unexpectedly comes **alive again** (this is an opportunity more than a failure).

Parsimonious failure recovery (1)

Algorithm Computing ND-simulation - parameterized version

Input: - target service $T = \langle A, S_T, t^0, \delta_T, F_T \rangle$
- available services $\mathcal{S}_i = \langle A, S_i, s_i^0, \delta_i, F_i \rangle$, $i = 1,..,n$
- relation R_{raw} including the simulated-by relation
- relation R_{sure} included the simulated-by relation

Output: the **simulated-by** relation (the largest simulation)

Body

$Q = \emptyset$

$Q' = R_{raw} - R_{sure}$ //Note $R' = Q' \cup R_{sure}$

while ($Q \neq Q'$) {

$Q := Q'$

$Q' := Q' - \{(t, s_1,..,s_n) \mid \exists\, t \to_a t'$ in $T \wedge \neg\exists\, k = 1,..,n$ s.t.
$(\exists\, s_k \to_a s_k' \wedge \forall\, s_k \to_a s'_k \supset (t', s_1,..,s'_k,..,s_n) \in Q' \cup R_{sure})\}$

}

return $Q' \cup R_{sure}$

End

Parsimonious failure recovery (2)

- Let $[1,.., n] = W \cup F$ be the available services.
- Let $R_{W \cup F}$ be the **simulated-by** relation of target by services $W \cup F$.
- Then the following holds:
- $R_W \subseteq \pi_W(R_{W \cup F})$
  - $\pi_W(R_{W \cup F})$ is the **projection on W** of a relation: easy to compute
  - *Note: $\pi_W(R_{W \cup F})$ is not a simulation of target by services W*
- $R_W \times F \subseteq R_{W \cup F}$
  - $R_W \times F$ is the **cartesian product** of 2 relations (F is trivial): easy to compute
  - *Note: $R_W \times F$ is a simulation of target by services $W \cup F$*

Parsimonious failure recovery (3)

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

- If **services F die**
  compute simulated-by $\boldsymbol{R}_{W}$ with $\mathbf{R_{raw}} = \pi_{W}(\boldsymbol{R}_{W \cup F})$!

- If **dead services F** come back
  compute simulated-by $\boldsymbol{R}_{W \cup F}$ with $\mathbf{R_{sure}} = \boldsymbol{R}_{W} \times F$!

- Remember:
  - $\boldsymbol{R}_{W} \subseteq \pi_{W}(\boldsymbol{R}_{W} \cup F)$
  - $\boldsymbol{R}_{W} \times F \subseteq \boldsymbol{R}_{W} \cup F$ and $\boldsymbol{R}_{W} \times F$ is a simulation of target by services $W \cup F$

Tools for computing composition based on simulation

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

- Computing simulation is a well-studied problem (related to computing **bisimulation** a key notion in process algebra).
  Tools, like the Edinburgh Concurrency Workbench and its clones, can be adapted to compute composition via simulation.

- Also **LTL-based synthesis** tools, like TLV, can be used for (indirectly) computing composition via simulation [AIJ'13]

We are currently focusing on the second approach.

Extensions

- **Nondeterministic** (angelic) **target** specification
  - Loose specification in client request [ICSOC' 04]
  - **Angelic (don't care)** vs devilish (don't know) nondeterminism
  - Exist the **maximal realizable target** [IJCAI'13]
- **Distributing** the orchestration
  - Often a centralized orchestration is unrealistic: eg. services deployed on mobile devices
    - too tight coordination
    - too much communication
    - orchestrator cannot be embodied anywhere
  - Drop centralized orchestrator in favor of **independent controllers** on single available services (exchanging messages)
  - Under suitable conditions: a distributed orchestrator exists iff a centralized one does
  - Still decidable (EXPTIME-complete) [AAAI' 07]
- Dealing with **data**
  - This is the single most difficult issue to tackle
    - First results: actions as DB updates, see [VLDB' 05]
    - Literature on Abstraction in Verification
  - From finite to **infinite transition** systems!
  - **Enormous progresses** in the very last years [ICSOC'10, BPM'11, KR'12, PODS'13]
- Other:
  - **Partial observability** of available services [ICAPS'09]
  - **Security and trust** aware composition [SWS' 06]
  - Automatic **Workflows** Composition of Mobile Services [ICWS'07]
  - Applications in smart homes [CAISE'12]

References

[ICSOC' 03] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella: Automatic Composition of E-services That Export Their Behavior. ICSOC 2003: 43-58

[WES' 03] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella: A Foundational Vision of e-Services. WES 2003: 28-40

[TES' 04] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella: : A Tool for Automatic Composition ofServices Based on Logics of Programs. TES 2004: 80-94

[ICSOC' 04] Daniela Berardi, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella, Diego Calvanese: Synthesis of underspecified composite e-services based on automated reasoning. ICSOC 2004: 105-114

[IJCIS' 05] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella: Automatic Service Composition Based on Behavioral Descriptions. Int. J. Cooperative Inf. Syst. 14(4): 333-376 (2005)

[VLDB' 05] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Richard Hull, Massimo Mecella: Automatic Composition of Transition-based Semantic Web Services with Messaging. VLDB 2005: 613-624

[ICSOC' 05] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Massimo Mecella: Composition of Services with Nondeterministic Observable Behavior. ICSOC 2005: 520-526

[SWS' 06] Fahima Cheikh, Giuseppe De Giacomo, Massimo Mecella: Automatic web services composition in trustaware communities. Proceedings of the 3rd ACM workshop on Secure web services 2006. Pages: 43 - 52.

[AISC' 06] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Massimo Mecella. Automatic Web Service Composition: Service-tailored vs. Client-tailored Approaches. In Proc. AISC 2006, International Workshop jointly with ECAI 2006.

[FOSSACS' 07] Anca Muscholl, Igor Walukiewicz: A lower bound on web services composition. Proceedings FOSSACS, LNCS, Springer, Volume 4423, page 274–287 - 2007.

[IJCAI' 07] Giuseppe De Giacomo, Sebastian Sardiña: Automatic Synthesis of New Behaviors from a Library of Available Behaviors. IJCAI 2007: 1866-1871

[AAAI' 07] Sebastian Sardiña, Fabio Patrizi, Giuseppe De Giacomo: Automatic synthesis of a global behavior from multiple distributed behaviors. In Proceedings of the Conference on Artificial Intelligence (AAAI), Vancouver, Canada, July 2007.

[IJFCS08] Daniela Berardi, Fahima Cheikh, Giuseppe De Giacomo, Fabio Patrizi: Automatic Service Composition via Simulation. IJFCS, 2008

[KR08] Sebastian Sardiña, Fabio Patrizi, Giuseppe De Giacomo: Behavior composition in the presence of failure. KR 2008.

[ICAPS08] Sebastian Sardiña, Giuseppe De Giacomo: Realizing Multiple Autonomous Agents through Scheduling of Shared Devices. ICAPS 2008.

[IEEEBul08] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella, Fabio Patrizi. Automatic service composition and synthesis: the Roman Model. Bull. of the IEEE Computer Society Technical Committee on Data Engineering, 31(3):18-22, 2008.

References

[WS-FM'09] Giuseppe De Giacomo, Fabio Patrizi: Automated Composition of Nondeterministic Stateful WS-FM 2009: 147-160

[IJCAI'09] Sebastian Sardiña, Giuseppe De Giacomo: Composition of ConGolog Programs. IJCAI 2009: 904-910

[ICAPS'09] Giuseppe De Giacomo, Riccardo De Masellis, Fabio Patrizi: Composition of Partially Observable Services Exporting their Behaviour. ICAPS 2009

[AAMAS'10] Giuseppe De Giacomo, Fabio Patrizi, Sebastian Sardiña: Agent programming via planning programs. AAMAS 2010

[AAMAS'10b] Giuseppe De Giacomo, Paolo Felli: Agent composition synthesis based on ATL. AAMAS 2010

[AAAI'10] Giuseppe De Giacomo, Paolo Felli, Fabio Patrizi, Sebastian Sardiña: Two-Player Game Structures for Generalized Planning and Agent Composition. AAAI 2010

[CAISE'12] Giuseppe De Giacomo, Claudio Di Ciccio, Paolo Felli, Yuxiao Hu, Massimo Mecella: Goal-Based Composition of Stateful Services for Smart Homes. CAISE 2012: 194-211

[AIJ'13] Giuseppe De Giacomo, Fabio Patrizi, Sebastian Sardiña: Automatic behavior composition synthesis. Artif. Intell. 196: 106-142 (2013)

On data and processes

[VLDB' 05] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Richard Hull, Massimo Mecella: Automatic Composition of Transition-based Semantic Web Services with Messaging. VLDB 2005: 613-624

[ICSOC'10] Piero Cangialosi, Giuseppe De Giacomo, Riccardo De Masellis, Riccardo Rosati: Conjunctive Artifact-Centric Services. ICSOC 2010: 318-333

[BPM'11] Babak Bagheri Hariri, Diego Calvanese, Giuseppe De Giacomo, Riccardo De Masellis, Paolo Felli: Foundations of Relational Artifacts Verification. BPM 2011: 379-395

[KR'12] Francesco Belardinelli, Alessio Lomuscio, Fabio Patrizi: An Abstraction Technique for the Verification of Artifact-Centric Systems. KR 2012

[PODS'13] Babak Bagheri Hariri, Diego Calvanese, Giuseppe De Giacomo, Alin Deutsch, Marco Montali: Verification of Relational Data-Centric Dynamic Systems with External Services. PODS 2013