La Sapienza
Università degli Studi di Roma

Automatic Composition of *e*-Services that Export their Behavior

D. Berardi, D. Calvanese, G. De Giacomo,
M. Lenzerini, M. Mecella

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"

lastname@dis.uniroma1.it
http://www.dis.uniroma1.it/~*lastname*/

Basics on *e*-Services

- *e*-Service is interactive program

  typically delivered over the Internet

- … that exports its behavior …

  i.e., its process

- … in terms of an abstract description

  e.g., state chart, UML state-transition diagram, FSM

- A client selects and interacts with it according to the description exported

Community of *e*-Services

- A community of *e*-Services is
  - a set of *e*-Services …
  - … that share implicitly a *common understanding* on a common set of actions …
  - … and export their behavior using this common set of actions
- A client specifies needs as e-Service behavior using the common set of actions of the community

e-Service exports its behavior …

Many possible ways. In this talk…

- Behavior modeled by a finite state machines

  core of state chart, UML state-transition diagram, etc.

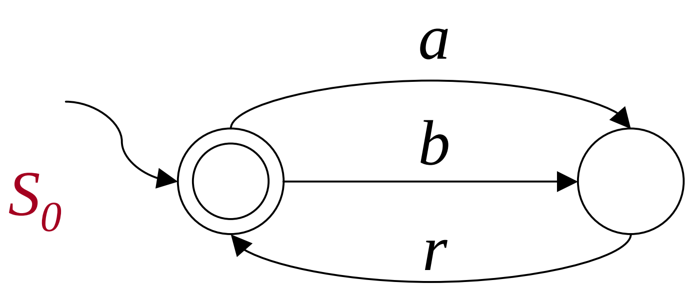
- In fact using a FSM we compactly describe all possible sequences of deterministic (atomic) actions: tree of all possible sequences of actions

- Data produced by actions not explicitly modeled

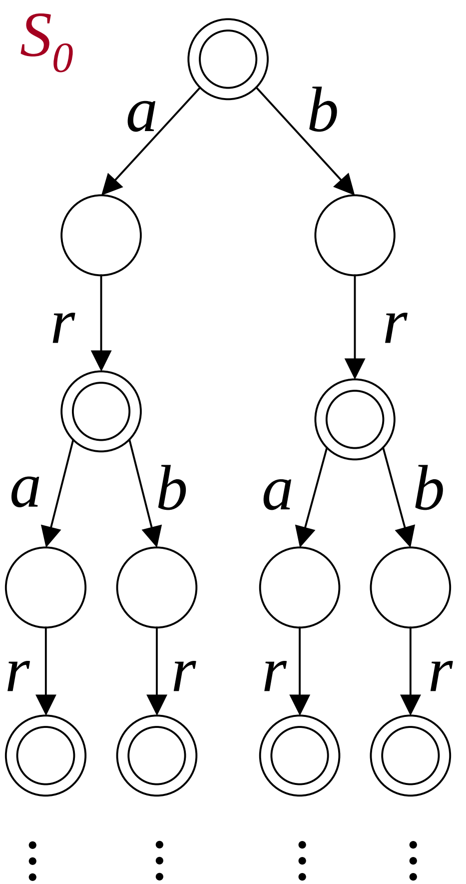
  data are used by the client to choose next action

e-Service as execution tree

Required behavior represented as a FSM (a Moore machine)



Execution tree (obtained by FSM unfolding)



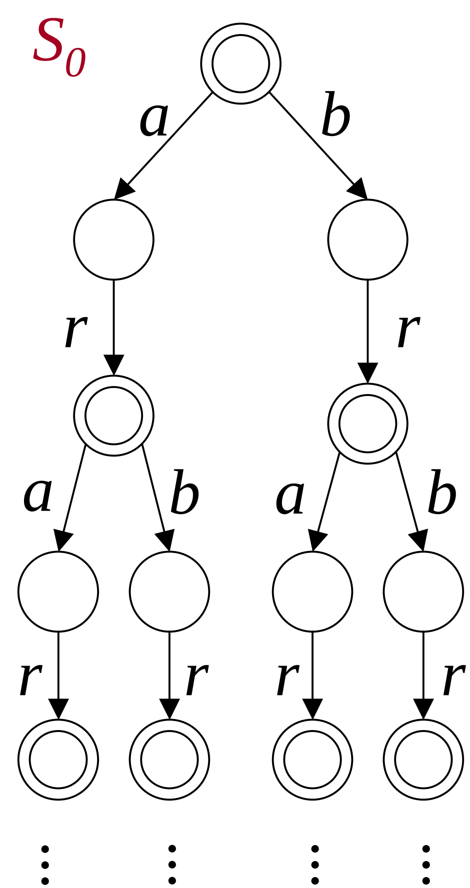
a: "search by author (and select)"
b: "search by title (and select)"
r: "listen (the selected song)"

e-Service as execution tree

- *Nodes: history (sequence) of actions executed so far*
- *Root: no action yet performed*
- *Successor node x·a of x: action a can be executed after the sequence of action x*
- *Final nodes: the e-Service can terminate*

a: "search by author (and select)"
b: "search by title (and select)"
r: "listen (the selected song)"

Execution tree (obtained by FSM unfolding)



e-Service composition

- Added value of the community…

  …when a client request cannot be satisfied by any available e-Service, it may still be possible to satisfy it by combining "pieces" of e-Services in the community

- Two issues arise:
  - support for synthesizing composition:
    - automatic synthesis of a coordinating program (composition) …
    - … that realizes the target e-Service (client request) …
    - … by suitably coordinating available e-Services

  addressed here

  - support for orchestration: execution of the coordinating program

  not addressed here

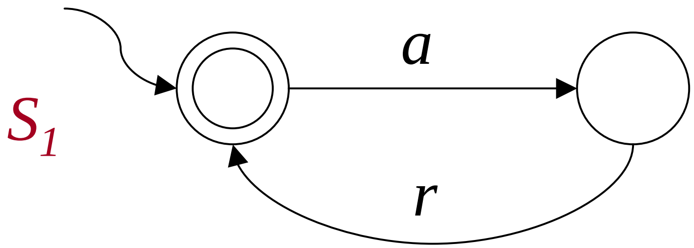
Formalizing *e*-Service composition

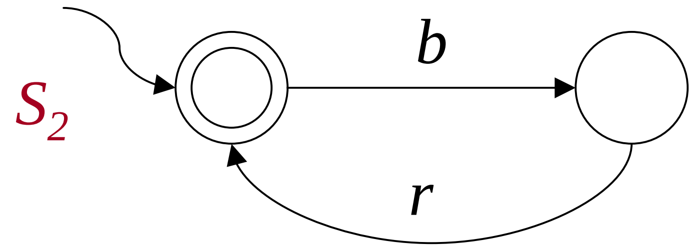
Composition:
- coordinating program …
- … that realizes the target e-Service …
- … by suitably coordinating available e-Services

- Composition can be formalized as:
  - a labeling of the execution tree of the target *e*-Service such that …
  - … each action in the execution tree is labeled by the community *e*-Service that executes it …
  - … and each possible sequence of actions on the target *e*-Service execution tree corresponds to possible sequences of actions on the community *e*-Service execution trees, suitably interleaved.

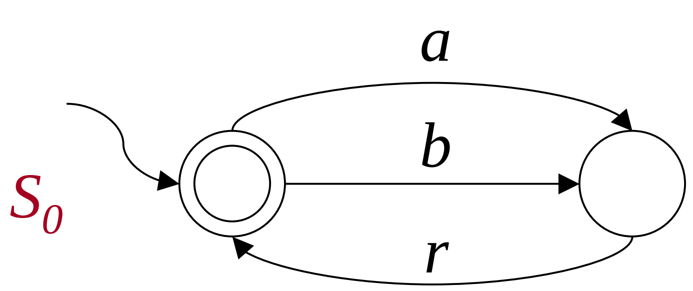
Example of composition

- Community *e*-Services *(expressed as FSMs)*

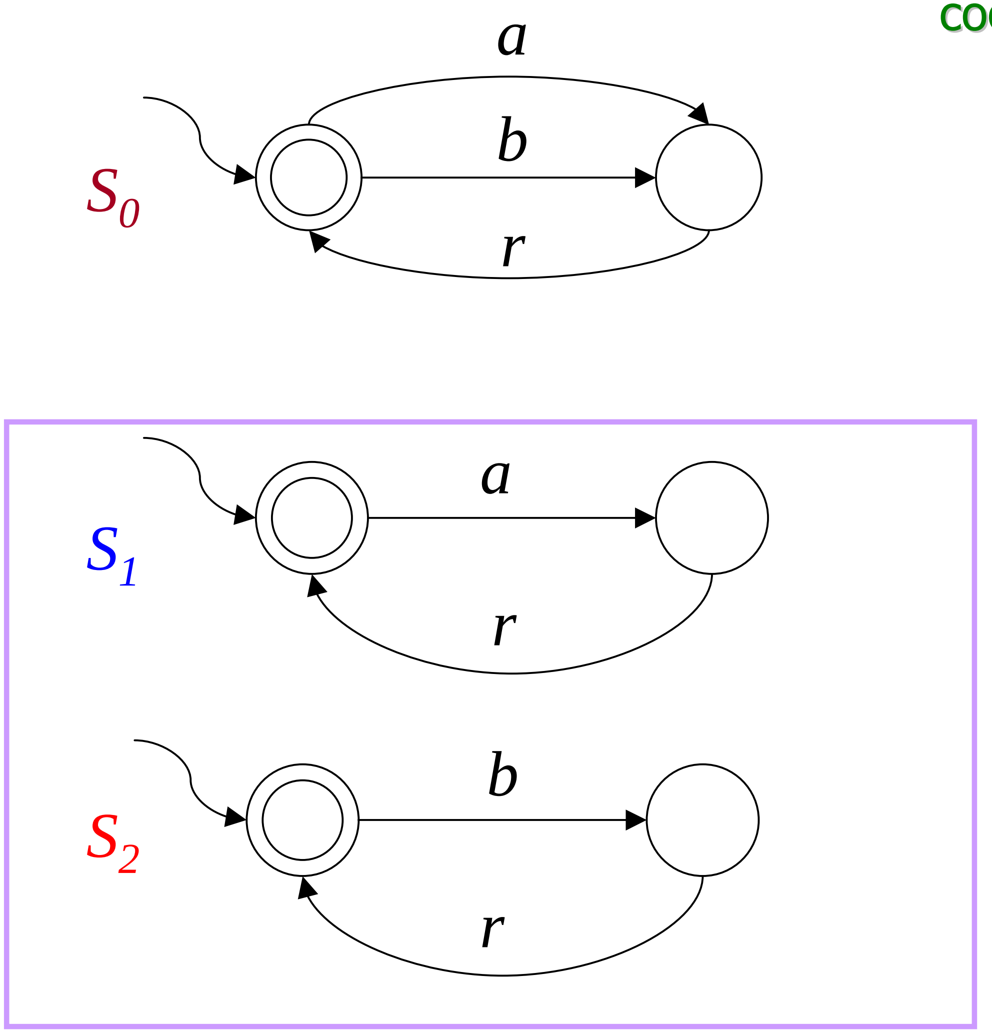




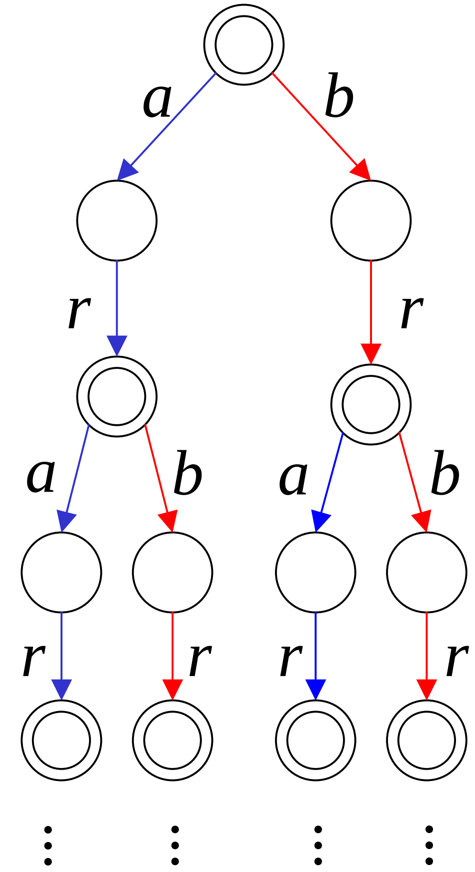
- Target *e*-Service *(again expressed as FSM)*



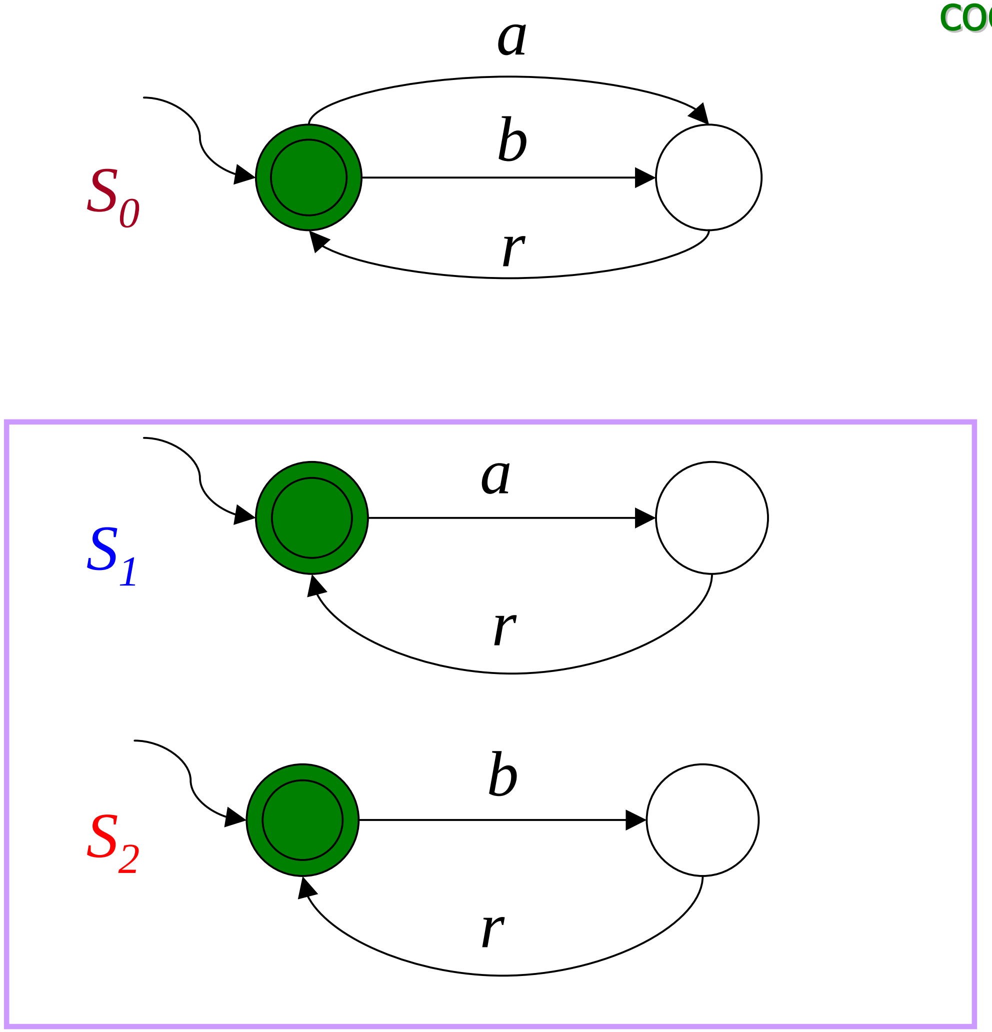
Example of composition



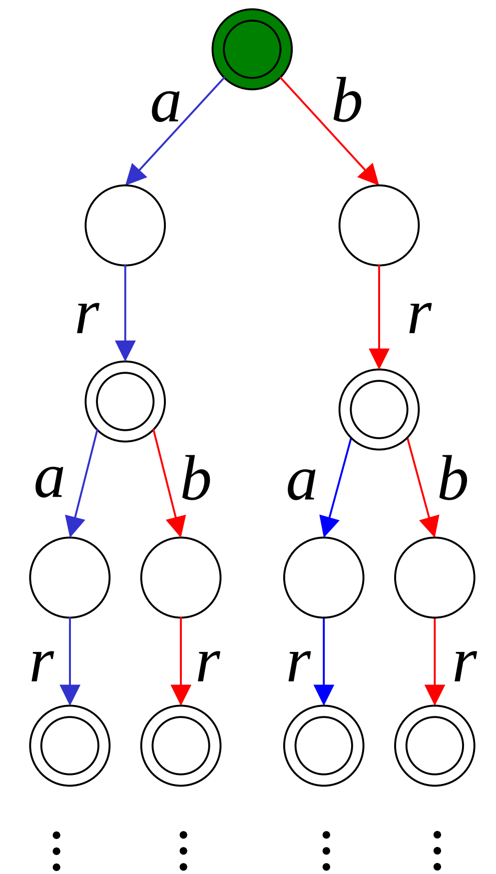
coordinating program (composition)



Example of composition

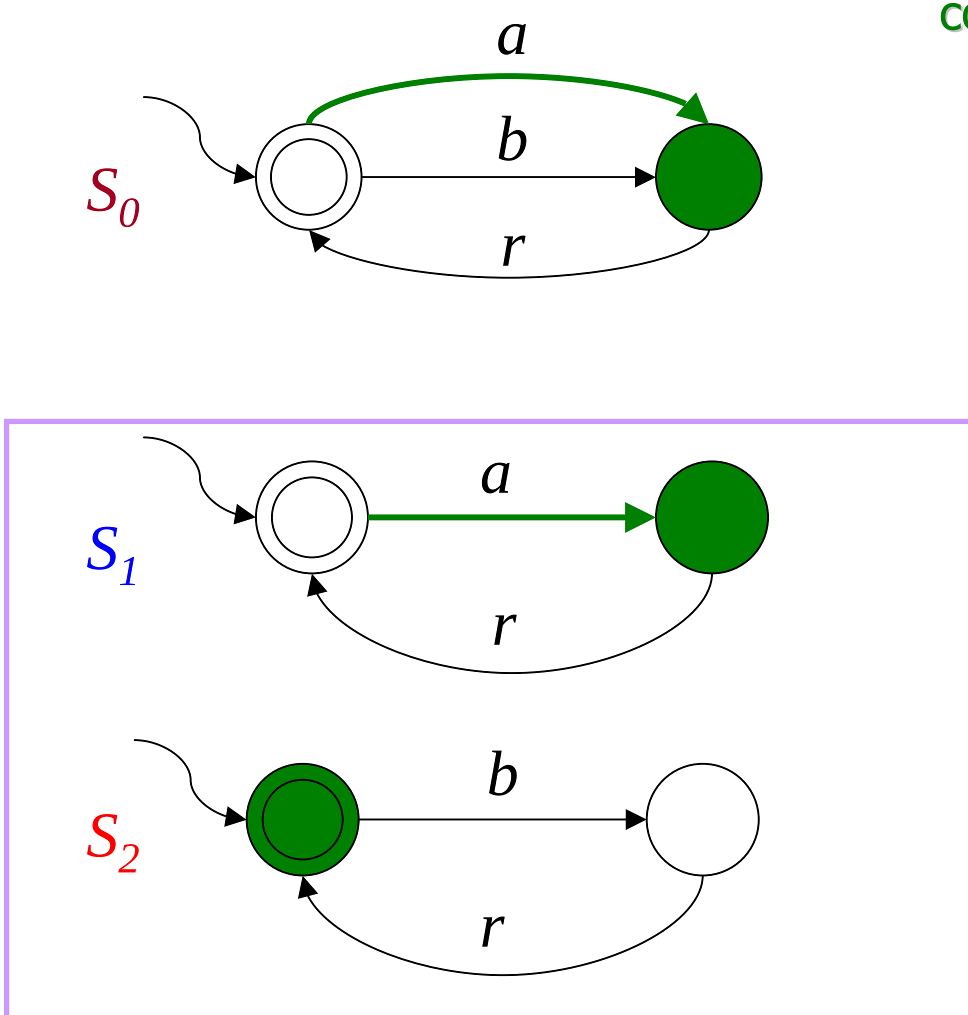


coordinating program (composition)

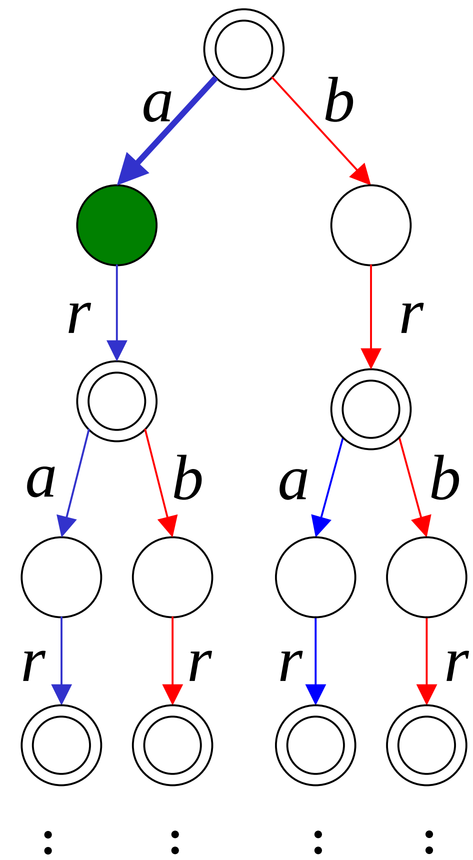


All e-Services start from their starting state

Example of composition

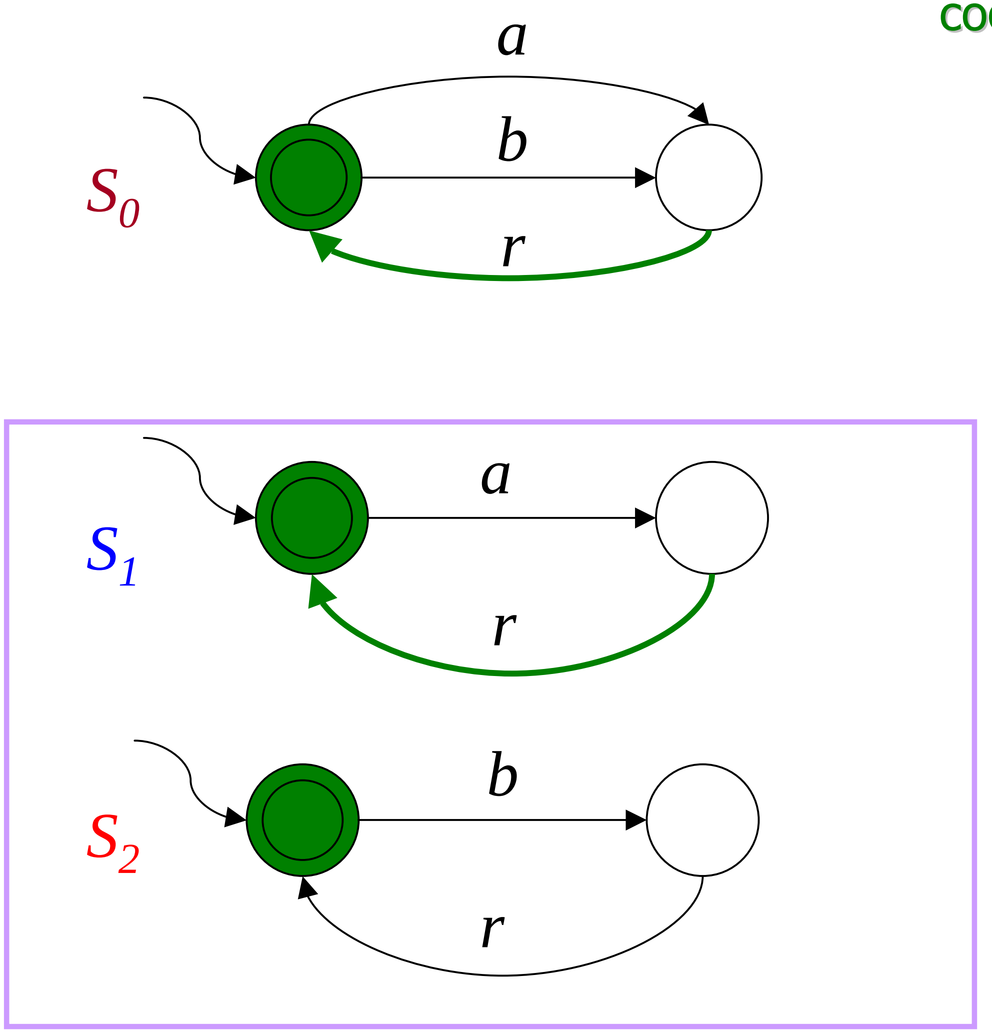


coordinating program (composition)

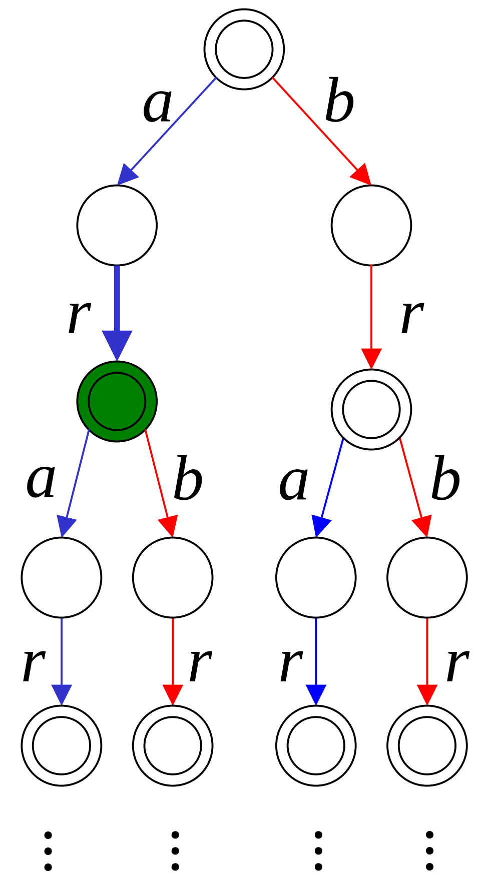


Each action of the target e-Service is executed by at least one of the component e-Services

Example of composition

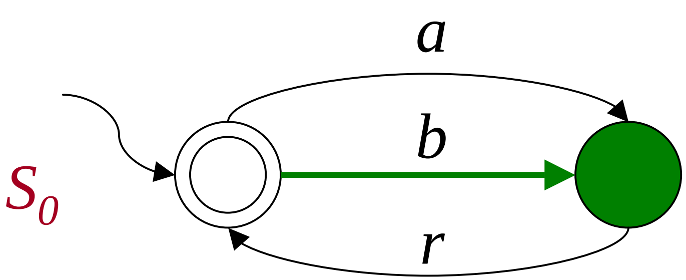


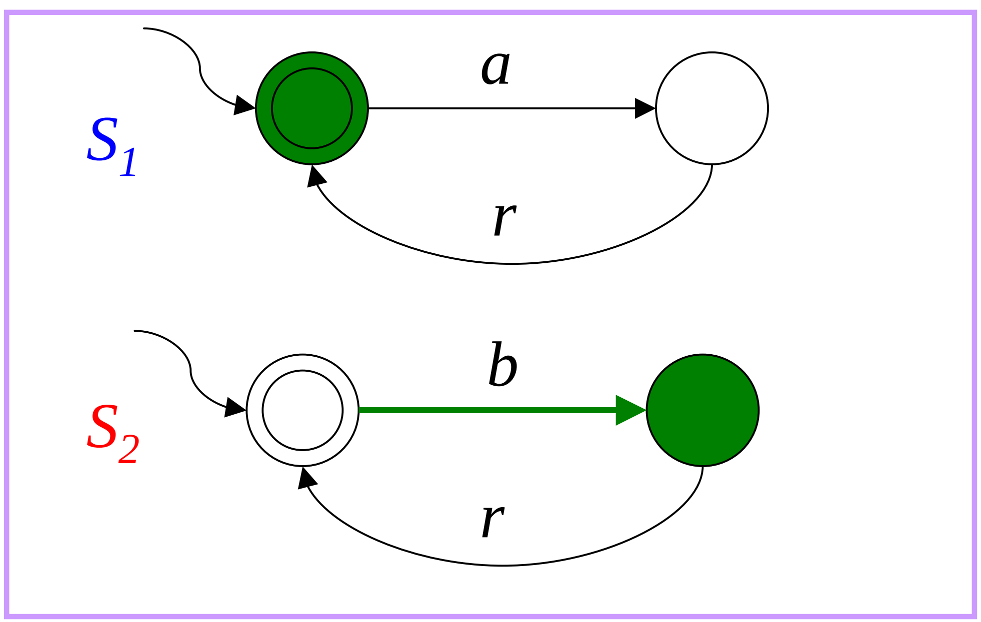
coordinating program (composition)



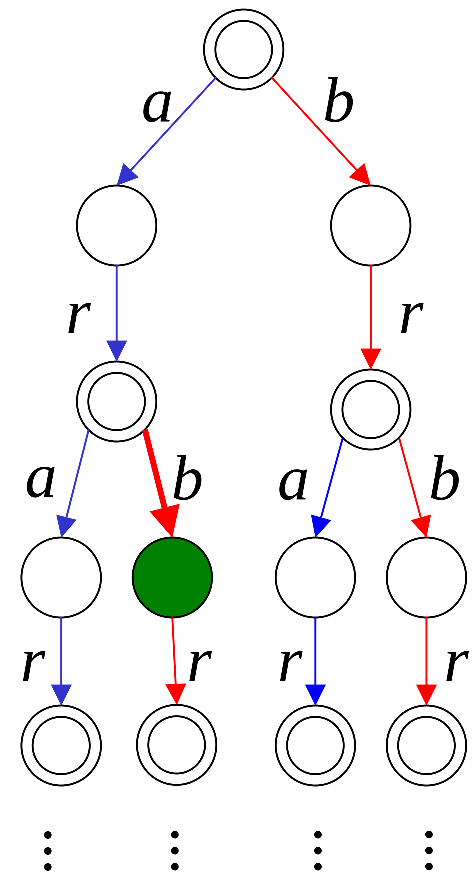
When the target e-Service can be left, then all component e-Services must be in a final state

Example of composition

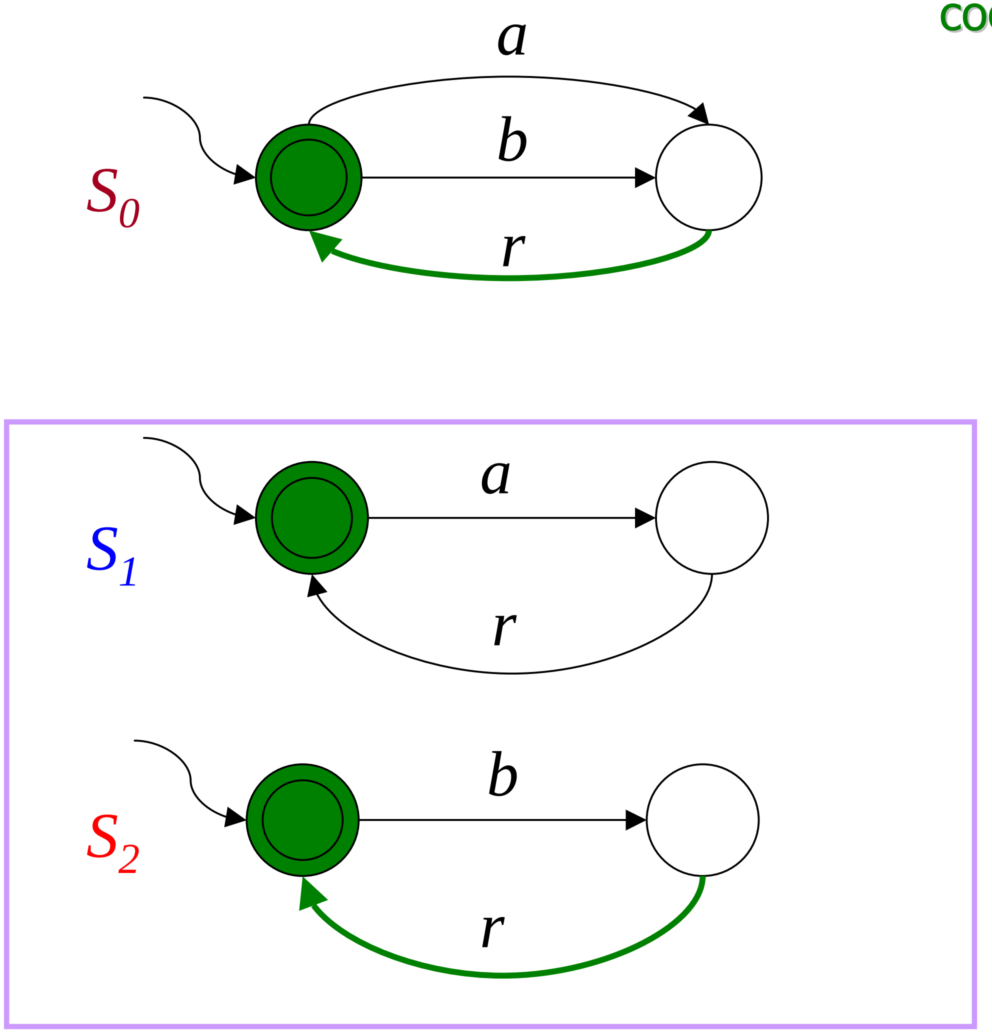




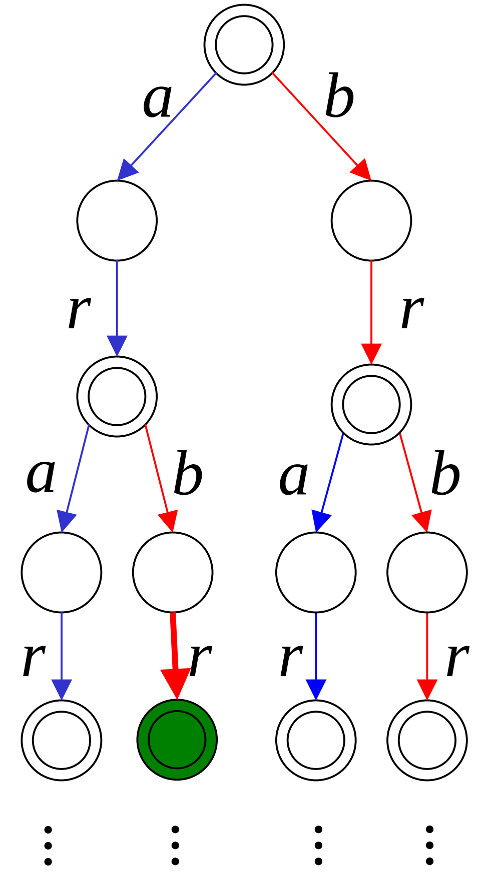
coordinating program (composition)



Example of composition

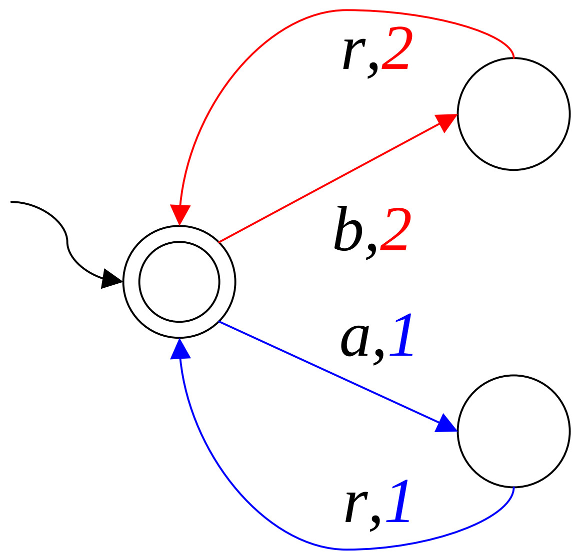


coordinating program (composition)



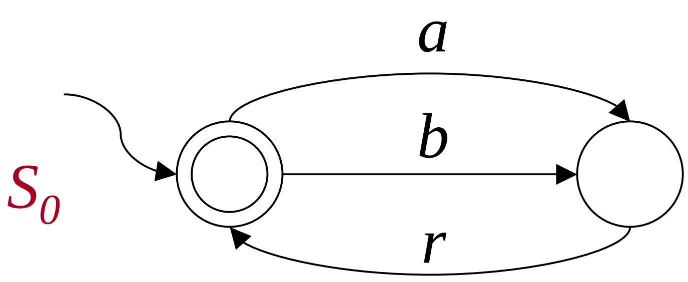
Observation

- This labeled execution tree has a finite representation as a FSM, a Mealy machine

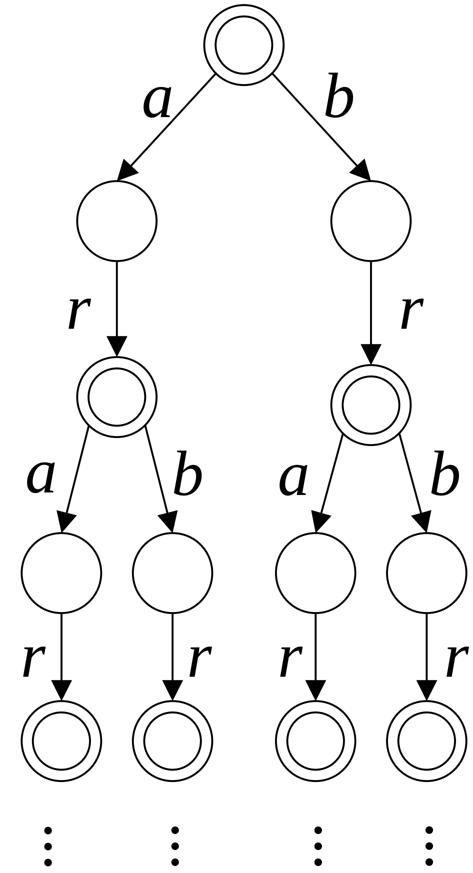


Is this always the case when we deal with e-Services expressible as FSMs? See later…

Observation



Execution tree of S_0
(obtained by FSM unfolding)



Note: we cannot label the FSM directly …

… we need to label the execution tree

Questions

Assume *e*-Services of community and target *e*-Service are FSMs

- Can we always check composition existence?
- If a composition exists there exists one which is a Mealy machine (i.e., finite state)?
- If yes, how can a Mealy machine composition by computed?

To answer we exploit DPDL (a well-known modal logic for reasoning on program schemas)

Answers

Reduce *e*-Service composition synthesis to satisfability in DPDL

- Can we always check composition existence?

  Yes, SAT in DPDL is decidable in EXPTIME

- If a composition exists there exists one which is a Mealy machine (i.e., finite state)?

  Yes, by the small model property of DPDL

- How can a Mealy machine composition by computed?

  From a (small) model of the corresponding DPDL formula

DPDL encoding

$$\Phi = \mathbf{Init} \wedge ([u]\Phi_0 \wedge_{i=1,\ldots,n} [u]\Phi_i \wedge [u]\Phi_{aux})$$

Initial states of all *e*-Services

DPDL encoding of target *e*-Service

DPDL encoding of *i*-th component *e*-Service

DPDL additional domain-independent conditions

DPDL encoding is polinomial in the size of the e-Service FSMs

DPDL encoding

- Target *e*-Service $S_0 = (\Sigma, S_0, s^0_0, \delta_0, F_0)$

  in DPDL we define Φ_0 as the conjuction of:

  – $s \rightarrow \neg s'$ for all pairs of distinct states in S_0

    e-Service states are pair-wise disjoint

  – $s \rightarrow \langle a \rangle T \wedge [a]s'$ for each $s'=\delta_0(s,a)$

    target e-Service can do an a-transition going to state s'

  – $s \rightarrow [a] \perp$ for each $\delta_0(s,a)$ undef.

    target e-Service cannot do an a-transition

  – $F_0 \equiv \bigvee_{s \in F0} s$

    denotes target e-Service final states

- …

DPDL encoding (cont.d)

- Community *e*-Services $S_i = (\Sigma, S_i, s^0{}_i, \delta_i, F_i)$

  in DPDL we define Φ_i as the conjuction of:

  - $s \rightarrow \neg s'$ for all pairs of distinct states in S_i

    e-Service states are pair-wise disjoint

  - $s \rightarrow [a](moved_i \wedge s' \vee \neg moved_i \wedge s)$ for each $s'=\delta_i(s,a)$

    if e-Service moved then new state, otherwise old state

  - $s \rightarrow [a](\neg moved_i \wedge s)$ for each $\delta_i(s,a)$ undef.

    if e-Service cannot do a, and a is performed then it did not move

  - $F_i \equiv \bigvee_{s \in Fi} s$

    denotes community e-Service final states

- …

DPDL encoding (cont.d)

- Additional assertions Φ_{aux}
  - $<a>T \rightarrow [a] \vee_{i=1,\ldots,n} moved_i$ for each action a
    at least one of the community e-Services must move at each step
  - $F_0 \rightarrow \wedge_{i=1,\ldots,n} F_i$
    when target e-Service is final all comm. e-Services are final
  - $Init \equiv s^0_0 \wedge_{i=1\ldots n} s^0_i$
    Initially all e-Services are in their initial state

DPDL encoding: $\Phi = Init \wedge [u](\Phi_0 \wedge_{i=1,\ldots,n} \Phi_i \wedge \Phi_{aux})$

Results

Thm: Composition exists iff DPDL formula Φ SAT

From composition labeling of the target e-Service one can build a <u>tree model</u> of the DPDL formula and viceversa

Information on the labeling is encoded in predicates $moved_i$

- Composition existence of *e*-Services expressible as FSMs is decidable in EXPTIME

Results on Mealy composition

Thm: If composition exists then Mealy composition exists.

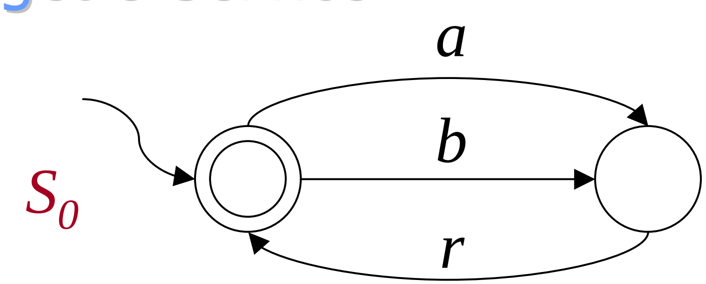
From a small model of the DPDL formula Φ,
one can build a Mealy machine

Information on the output function of the machine is encoded in predicates $moved_i$

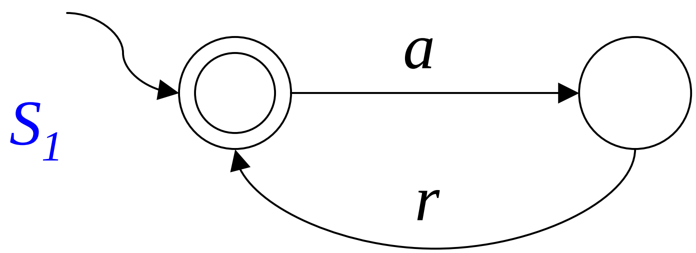
- Mealy composition existence of *e*-Services expressible as FSMs is decidable in EXPTIME

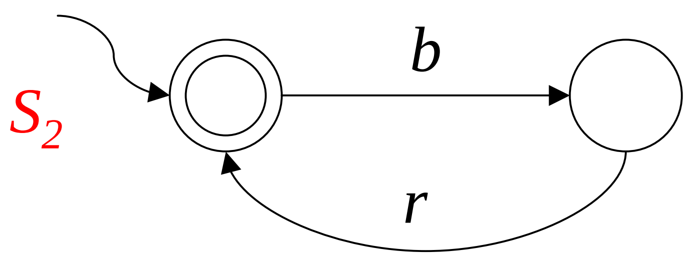
Example

Target *e*-Service



Community *e*-Services





DPDL

...

...

...

$s_0^0 \wedge s_1^0 \wedge s_2^0$

$\langle a \rangle\, T \rightarrow [a]\, (moved_1 \vee moved_2)$

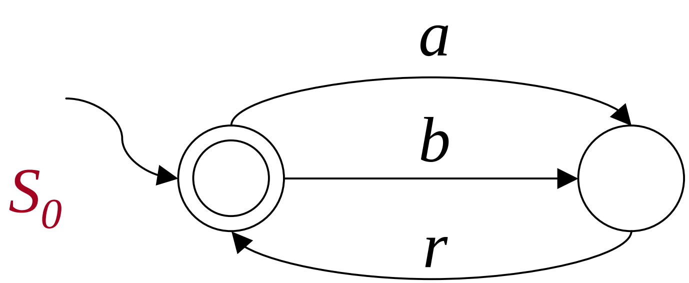
$\langle b \rangle\, T \rightarrow [b]\, (moved_1 \vee moved_2)$

$\langle r \rangle\, T \rightarrow [r]\, (moved_1 \vee moved_2)$

$F_0 \rightarrow F_1 \wedge F_2$

Example

Target *e*-Service



$s_0^0 \rightarrow \neg\, s_0^1$

$s_0^0 \rightarrow \langle a \rangle\, T \wedge [a]\, s_0^1$

$s_0^0 \rightarrow \langle b \rangle\, T \wedge [b]\, s_0^1$

$s_0^1 \rightarrow \langle r \rangle\, T \wedge [r]\, s_0^0$

$s_0^0 \rightarrow [r]\, \bot \wedge [r]\, s_0^0$

$s_0^1 \rightarrow [a]\, \bot$

$s_0^1 \rightarrow [b]\, \bot$

$F_0 \equiv s_0^0$

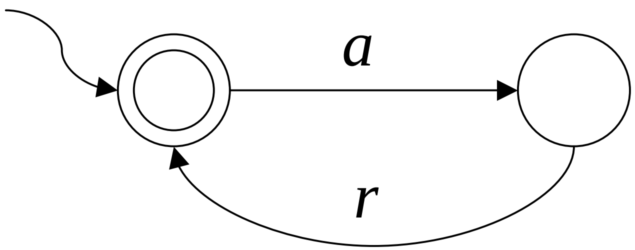
...

...

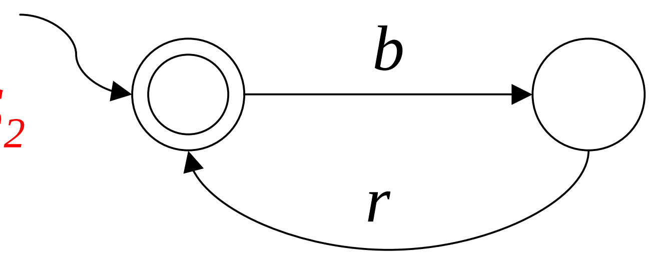
...

Example

Community *e*-Services



S_1



S_2

...

$s_1^0 \rightarrow \neg s_1^1$

$s_1^0 \rightarrow [a] (moved_1 \wedge s_1^1 \vee \neg moved_1 \wedge s_1^0)$

$s_1^0 \rightarrow [r] \neg moved_1 \wedge s_1^0$

$s_1^0 \rightarrow [b] \neg moved_1 \wedge s_1^0$

$s_1^1 \rightarrow [a] \neg moved_1 \wedge s_1^1$

$s_1^1 \rightarrow [b] \neg moved_1 \wedge s_1^1$

$s_1^1 \rightarrow [r] (moved_1 \wedge s_1^0 \vee \neg moved_1 \wedge s_1^0)$

$F_1 \equiv s_1^0$

$s_2^0 \rightarrow \neg s_2^1$

$s_2^0 \rightarrow [b] (moved_2 \wedge s_2^1 \vee \neg moved_2 \wedge s_2^0)$

$s_2^0 \rightarrow [r] \neg moved_2 \wedge s_2^0$

$s_2^0 \rightarrow [a] \neg moved_2 \wedge s_2^0$

$s_2^1 \rightarrow [b] \neg moved_2 \wedge s_2^1$

$s_2^1 \rightarrow [a] \neg moved_2 \wedge s_2^1$

$s_2^1 \rightarrow [r] (moved_2 \wedge s_2^0 \vee \neg moved_2 \wedge s_2^0)$

$F_2 \equiv s_2^0$

...

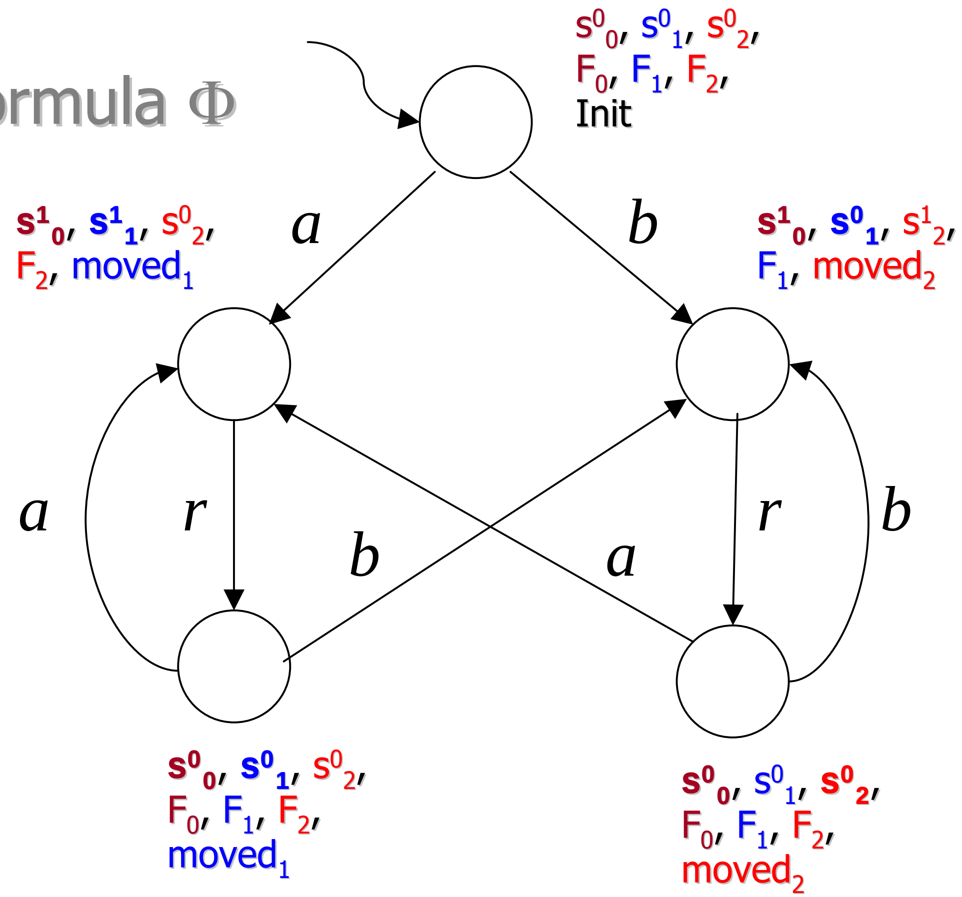
Example

Check: run SAT on DPDL formula Φ

Example

Check: run SAT on DPDL formula Φ

Yes $\Rightarrow$ (small) model

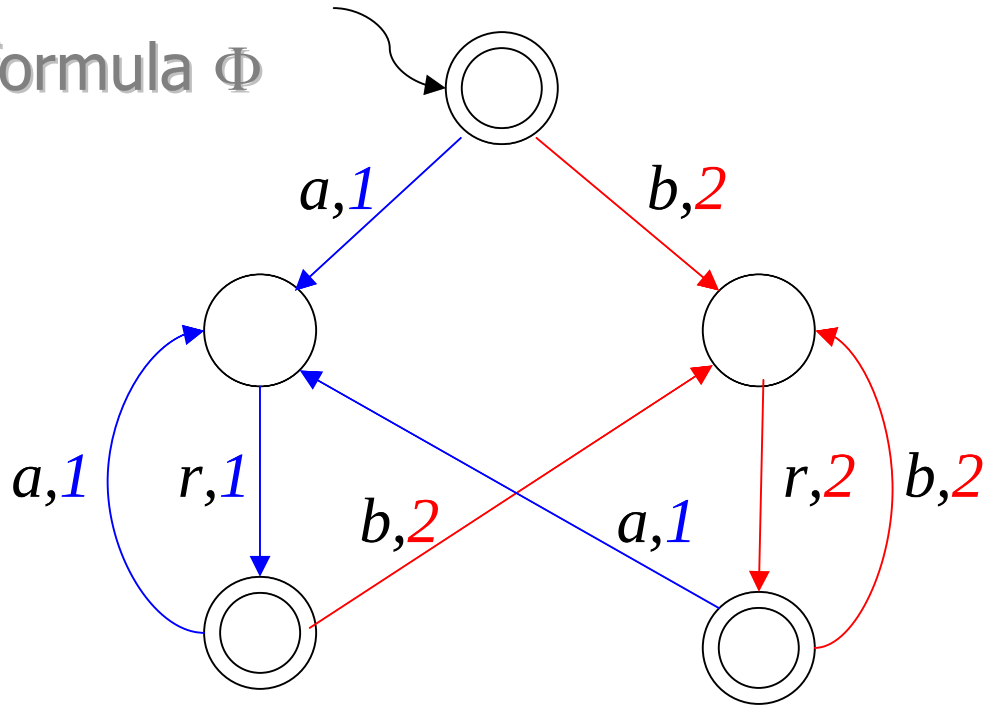


Example

Check: run SAT on DPDL formula Φ

Yes $\Rightarrow$ (small) model

$\Rightarrow$ extract Mealy machine



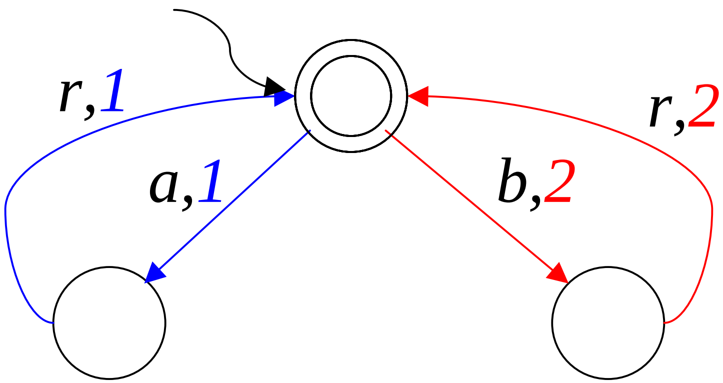
Example

Check: run SAT on DPDL formula Φ

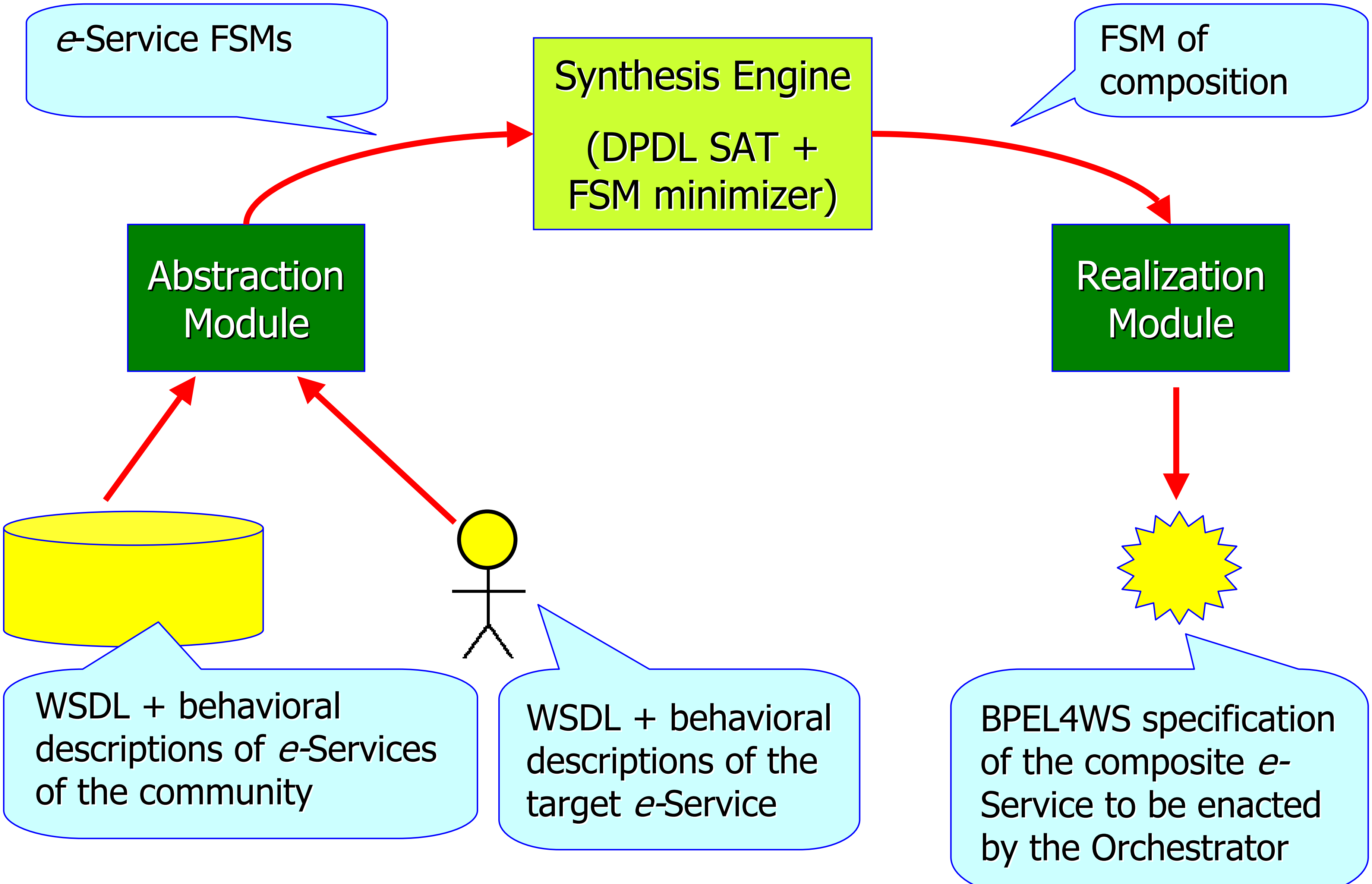
Yes $\Rightarrow$ (small) model

$\Rightarrow$ extract Mealy machine

$\Rightarrow$ minimize Mealy machine



The *e*-Service Composition System



Future work

We have only scratched the surface …

- Implementation? *We are working on it, using DPDL/DL-systems based on tableaux*

- Hardness of FSM *e*-Service composition?
  …at least PSPACE-hard! EXPTIME-hard?

- Loose specification of target e-Service?
  – target *e*-Service "under-specified" *Note: angelic nondeterminism ongoing work*

- Incomplete specification of e-Services of the community?
  – *e*-Services export partial description of their behavior to the community
  Note: synthesis with diabolic nondeterminism very hard!

- Data? …