La Sapienza
Università degli Studi di Roma

Automatic Composition of *e*-Services

Daniela Berardi

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

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

e-Services

- e-Services: distributed applications that export a semantic view of their behavior:
  - input / output behavior
  - “interactive” behavior
- e-Service Description
- e-Service Publication and Notification
- e-Service Discovery, Selection and Invocation
- (manual and automatic) e-Service Composition
  - e-Service Orchestration
  - e-Service Compatibility, Substitutability, Adaptation
- e-Service advertisement, e-Service negotiation, Quality of e-Services, security and privacy issues,…
- A lot of industrial and technological efforts (WSDL, BPEL4WS, UDDI, …)

Description and Automatic Composition of e-Services: Relevant Work

- (Implicit or explicit) contribution from several research areas:
  - Artificial Intelligence:
    - *e*-Services as (set of) atomic actions
    - composition by exploiting agent-based techologies and planning techniques
  - Theoretical Computer Science:
    - *e*-Services as finite state machines
    - composition as automata synthesis
  - WorkFlow, Databases, Software Engineering,...

Description and Automatic Composition of e-Services: Main Results

- McIlraith et.al.:
  - e-Services as complex actions in SitCalc, seen as atomic by the client [IEEE01, KR02]
    - e-Services as generic ConGolog procedures
    - client specification involves call to the procedures
    - (angelic) nondeterminism in client is allowed and resolved by ConGolog intepreter
  - Petri Net representation of composition of atomic e-Services [WWW02]
- [Hull et.al.: PODS03, WWW03]
  - e-Services as abstract peers that can execute certain set of actions (message exchange)
  - given a desired global behavior (in terms of action execution) it is synthesized a finite state automaton for each peer to control its actions

General Goal of my Thesis

1. General framework for *e*-Services that export their behavior in terms of an abstract program-like structure

2. Formal analysis of *e*-Service behavior

3. Automatic *e*-Service composition synthesis
   - techniques, algorithms, computational complexity results

e-Services and Community of *e*-Services: The Model used by "Roman" Results

- An *e*-Service is an interactive program that exports its behavior in terms of an abstract description
- A client selects and interacts with it according to the description exported

- 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 ...

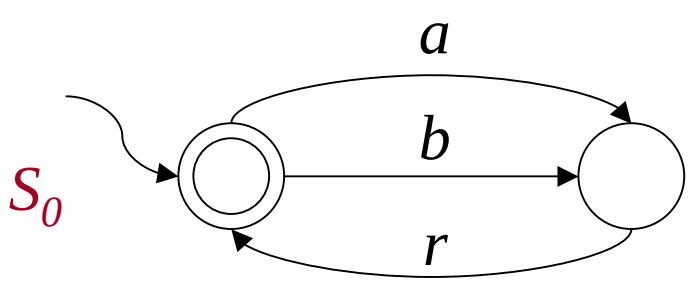
Many possible ways. Here...

- Behavior modeled by finite state machines
  core of state chart, UML state-transition diagram, etc.
  - in our FSMs, each transaction corresponds to an action *(e.g., search-by author-and-select, search-by title-and-select, listen-the-selected-song, ...)*
- 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

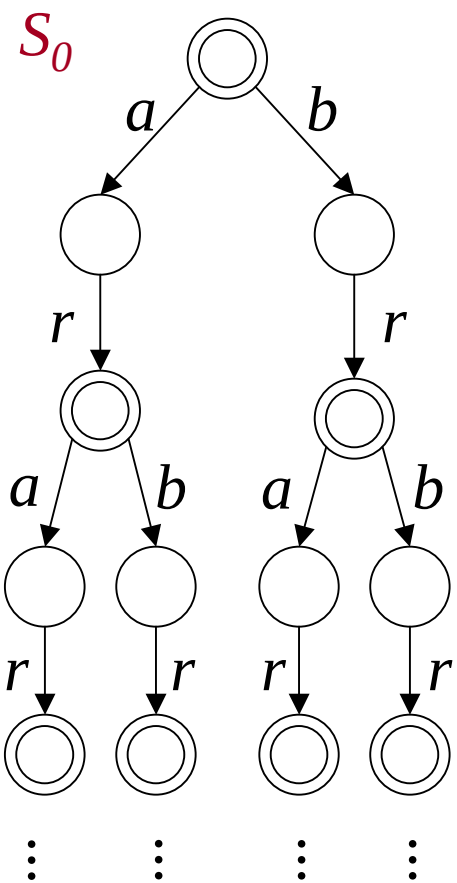
  data are used by the client to choose next action

e-Service as Finite State Machine

Required behavior represented as a FSM



Execution tree (obtained by FSM unfolding)



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

The Problem of Automatic *e*-Service Composition

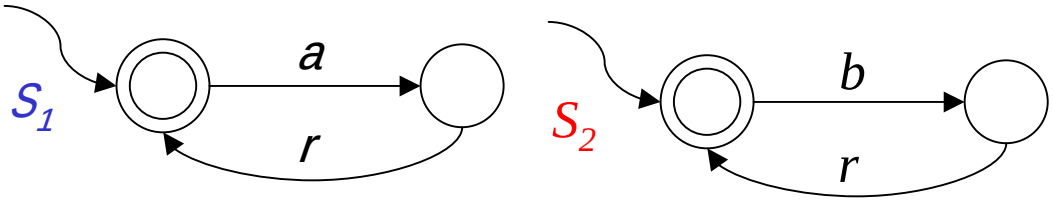
Generic statement of the problem:

- automatic synthesis of a coordinating program (composition) ...
- ... that realizes a client request ...
- ... by suitably coordinating available e-Services

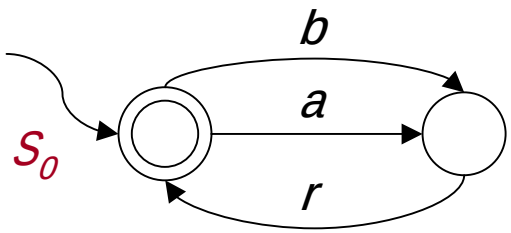
e-Service Composition in the "Roman Framework"

Given:

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

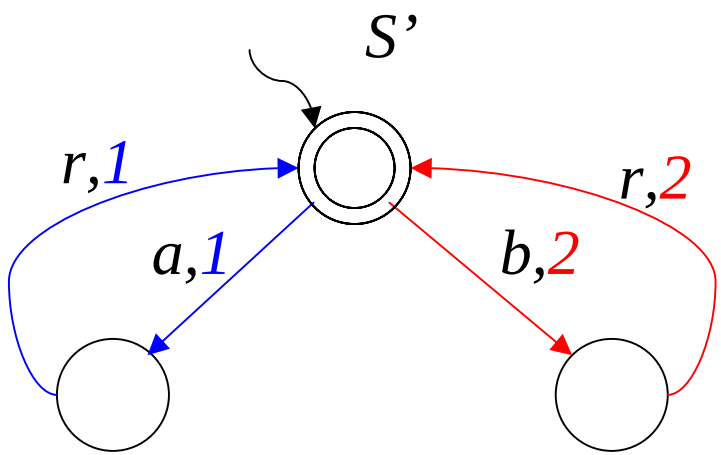


- Target *e*-Service S_0
(again expressed as FSM)



Find:

- new FSM *e*-Service S' (delegator):
  - new alphabet = actions x sets of service (identifiers)
  - "accepts" same language as S_0
  - For each accepting run of S' on word w, and for each S in C, "projection" of this run onto moves of S is an accepting computation for S



Key Idea for Finding Composition: Exploit Propositional Dynamic Logic (PDL) / Description Logics (DLs)

- Interesting properties of PDL/DL:
  - EXPTIME decidability
  - Tree model property
  - Small model property

  } We can automatically build a composition

  → We can automatically build a finite state composition

- Description Logics:
  - represent knowledge in terms of states (objects) and state transitions (links)
  - equipped with decidable reasoning
  - Here, we focus on *ALC*, seen as a simplified variant of PDL

How we Automatically Build Finite State e-Service Composition

ALC encoding of target *e*-Service

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

ALC additional domain-independent conditions

Initially all *e*-Services are in their initial states

ALC Knowledge Base:

Δ_0

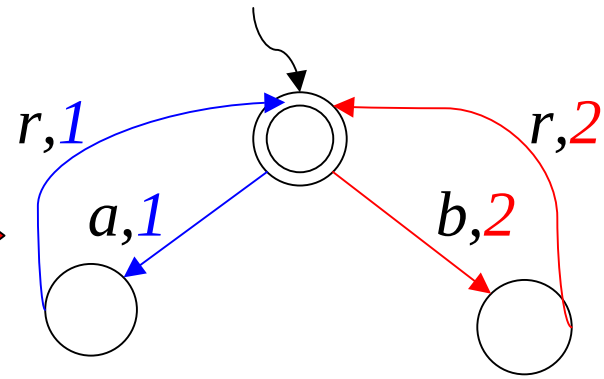
Δ_i

Δ_{aux}

Δ_{Init}

Check satisfiability (and build a model)

e-Service composition



Results

Thm 1: Composition exists iff DL Knowledge Base satisfiable

From composition labeling of the target e-Service one can build a tree model for the Knowledge Base, and vice-versa

Cor 1: Composition existence of *e*-Services, expressible as FSMs, is decidable in EXPTIME

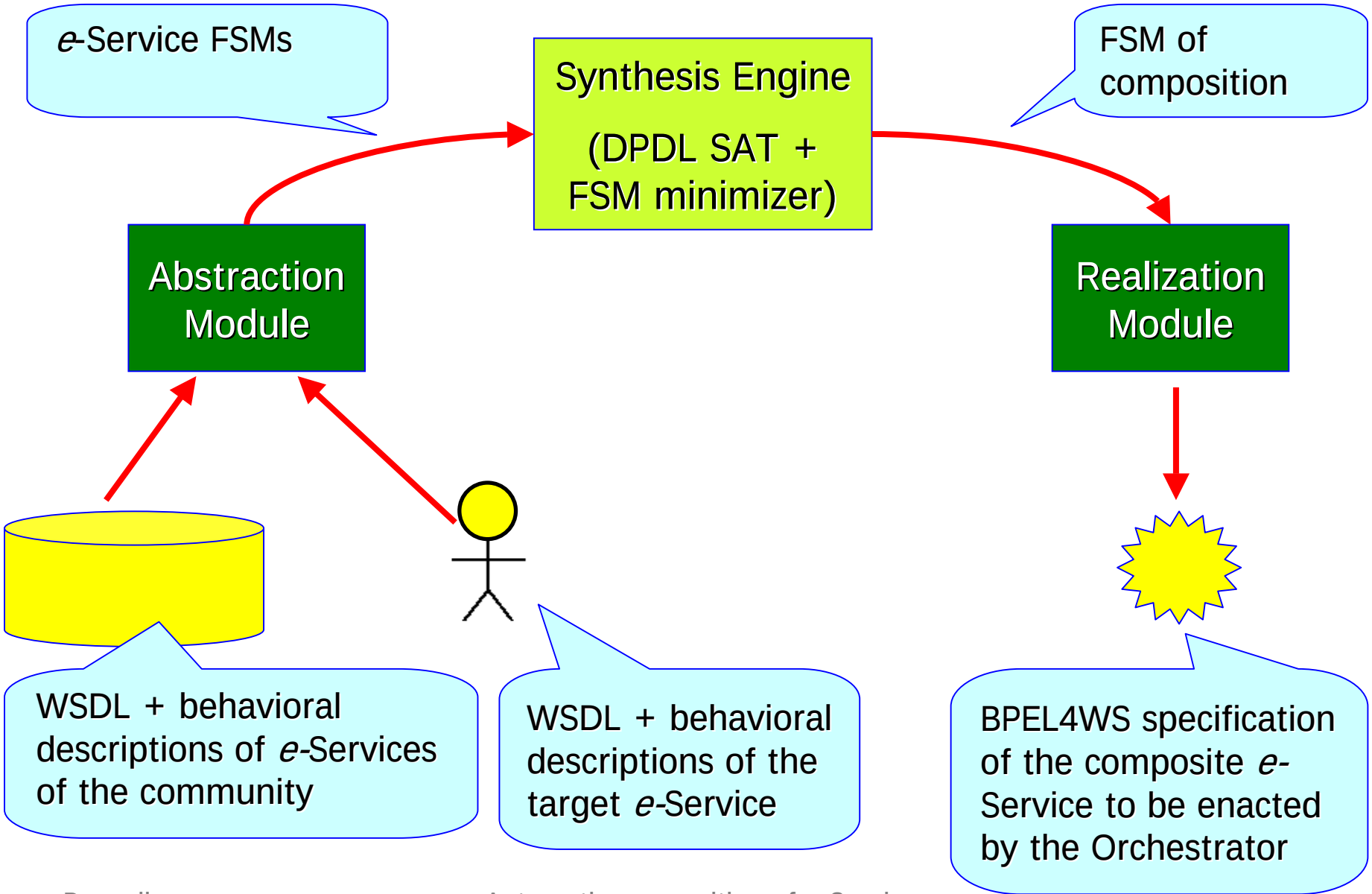
Thm 2: If composition exists then finite state composition exists.

From a small model of a DL Knowledge Base, one can build a finite state composition

Cor 2: Finite state composition existence of *e*-Services, expressible as FSMs, is decidable in EXPTIME

$\Rightarrow$ We can automatically build finite state composition

The *e*-Service Composition System



PARIDE Open Source Project

- We have developed a prototype tool that implements our technique
- The behavioral description of e-Services are expressed in WSTL (Web Service Transition Language):
  - it integrates well with existing standards
  - it has a clear conceptual model based on FSM
- The PARIDE (Process-based frAmewoRk for composItion and orchestration of Dinamyc *E*-Services) Open Source Project:

http://sourceforge.net/projects/paride/

- On this site we intend to release the various prototypes produced by our research.
- Tool developed within a master thesis project by Alessandro Iuliani

Some Remarks on the Framework

1. at each step the client chooses the next action
2. determinism on automata
3. the *e*-Services involved in the composition do not communicate one with the other

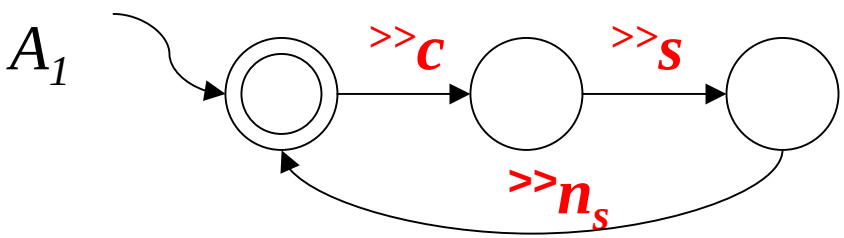
<u>Enhancing the Framework:</u> main ideas

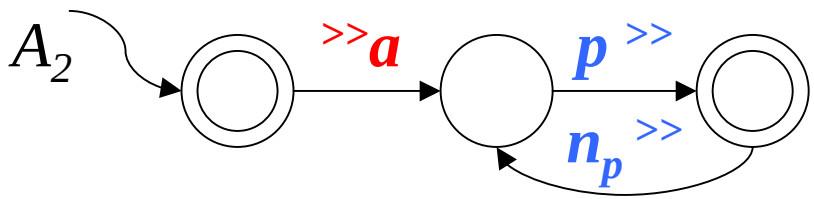
1. "sometimes" the client can leave the choice about the next action to the composition system
2. angelic nondeterminism: nondeterminism as don't care conditions on the next action
3. communication between component *e*-Services

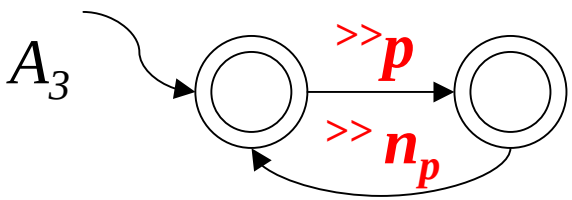
Enhancing the Framework: new roles

- Initiator : who requests the execution of an action
  - the client is always an initiator
  - each action has exactly one initiator
- Servant: who executes the requested action
  - each action has one or more servants

e-card Example: e-Services in the Community





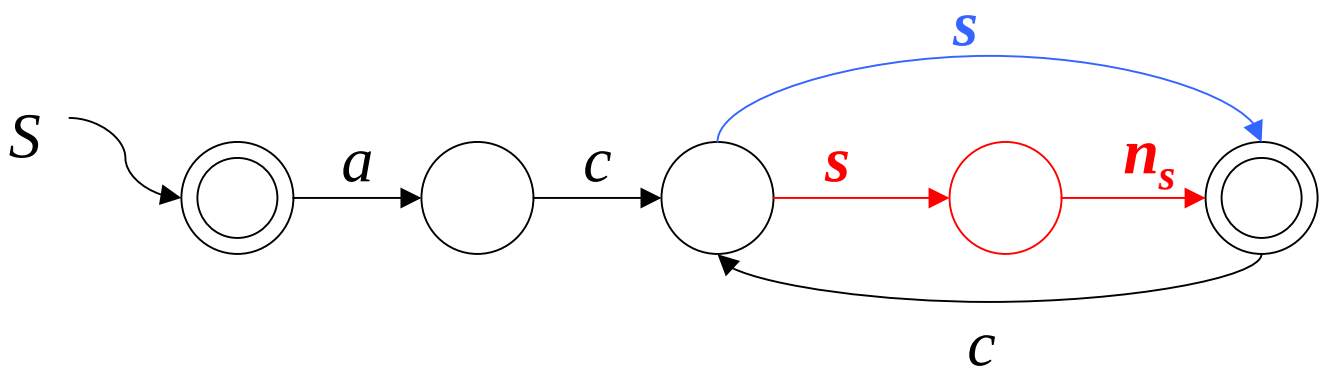


c= search_greeting_card_&_select
s = compose_&_send
n_s = notification_send
a = user_authentication
p = payment
n_p = notification_payment

Enhancing the framework: angelic nondeterminism

e-card Example: Client specification of desired e-Service

a = user_authentication
c= search_greeting_card_&_select
s = compose_&_send
n_s = notification_send

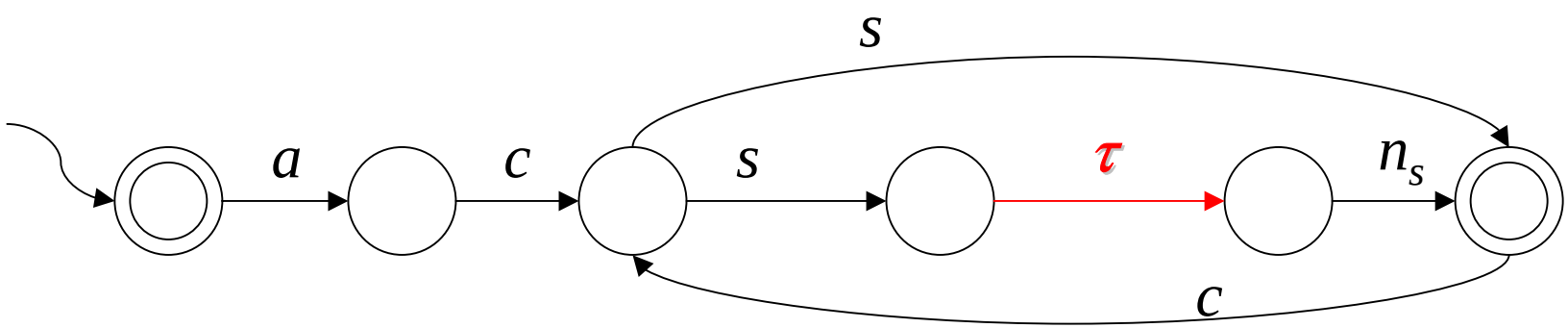


The client “doesn’t care” whether the blue or the red transition is taken (i.e., whether s/he receives a confirmation after sending the e-card or not)

Enhancing the framework: the τ action

e-card Example: Client specification of desired e-Service

a = user_authentication
c= search_greeting_card_&_select
s = compose_&_send
n_s = notification_send



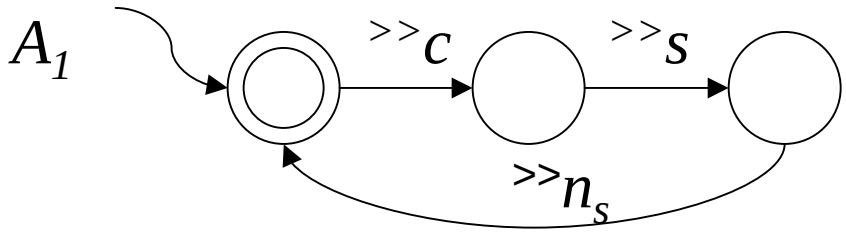
The client is not initiator (nor servant) relative to the τ transition: s/he lets the eServices involved in composition suitably communicate, without being "brought in"

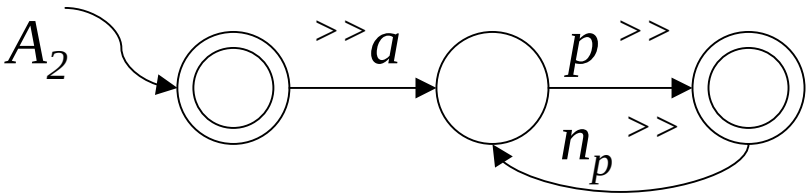
e-Service Composition in the "Roman Enhanced Framework"

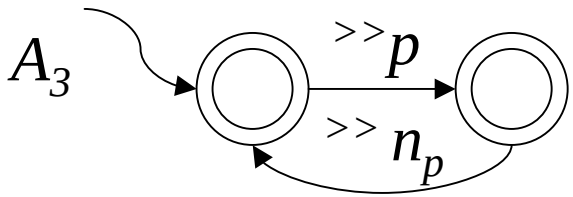
Given:

- Community C of *e*-Services

(expressed as FSMs)

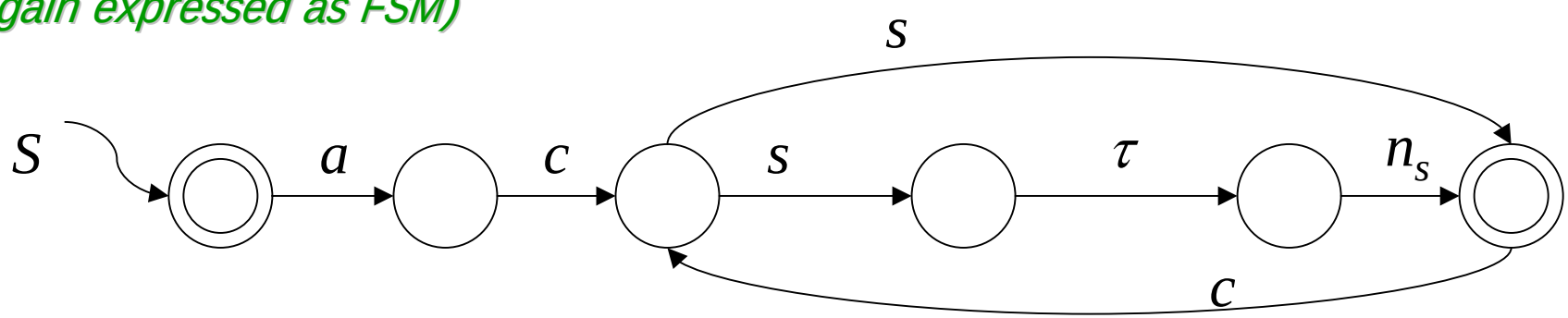






- Target *e*-Service S_0

(again expressed as FSM)

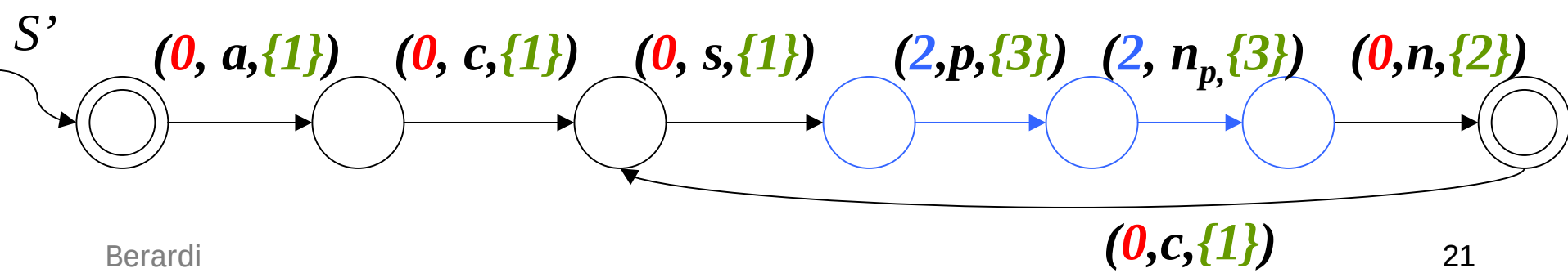


e-Service Composition in the "Roman Enhanced Framework" (cont.)

Find:

- new FSM *e*-Service S' (delegator):

- new alphabet = service initiator x actions x sets of service servants
- nondeterminism resolved by choosing a single successor state for each transition (including τ transitions) $\Rightarrow$ S' is a deterministic FSM
- each a transition add 0 if the client is the initiator of *a*
- each τ transition replaced by a finite sequence of transitions where client is NOT the initiator
- choosing for each transition a set of servants
- For each accepting run of S' on word w, and for each S in C, "projection" of this run onto moves of S is an accepting computation for S



Automatically Building e-Service Composition in the "Roman Enhanced Framework"

- As before, we exploit DLs
  - $ALCQ_{reg}$: a τ transition is realized through a *single sequence* of actions

$\Rightarrow$ We can automatically build finite state composition in the "Roman Enhanced Framework"

Future work

- Hardness of FSM *e*-Service composition?

  ...at least PSPACE-hard! EXPTIME-hard?
- Incomplete information on e-Services:
  - *e*-Services export partial description of their behavior to the community

    diabolic nondeterminism
- On-the-fly dynamic reconfiguration of composite service
  - *what about if one component service becomes unavailable (and new services become available) during composite service execution?*
  - "fixed" vs dynamic service community
- Enriching the language for describing services:
  - not only operational semantics
  - coping with non-functional features
  - Adding Data:
    - lower level of abstraction
    - new problems, e.g. how to deal with intrinsic nondeterminism ?

Future work:
Unified Framework for *e*-Service: a PSL based approach

- Joint work with Michael Gruninger, Rick Hull, Sheila McIlraith, within SWSL working group
- PSL (Process Specification Language): FOL ontology for describing process
- Aims:
  - to give a uniform conceptual view of SWS results from different approaches (e.g., automata-based, DL-based, Petri-net based, sitcalc-based, etc)
  - to explicitly represent messages and dataflow (cf. W3C choreography, behavioral message-based signatures, etc.)
  - to integrate with existing and emerging standards (BPEL, W3C choreography, etc.)

Info & Contacts

- Thesis dissertation scheduled for January, 2005

Daniela Berardi

e-mail: berardi@dis.uniroma1.it

home page: http://www.dis.uniroma1.it/~berardi

address: Dipartimento di Informatica e Sistemistica,
Universita' di Roma "La Sapienza"
Via Salaria, 113 (2 piano)
I-00198 Rome (Italy)

- Work done in collaboration with the Knowledge Representation and DataBase Group:

M. Lenzerini, G. De Giacomo, D. Calvanese (now in Bolzano) and M. Mecella