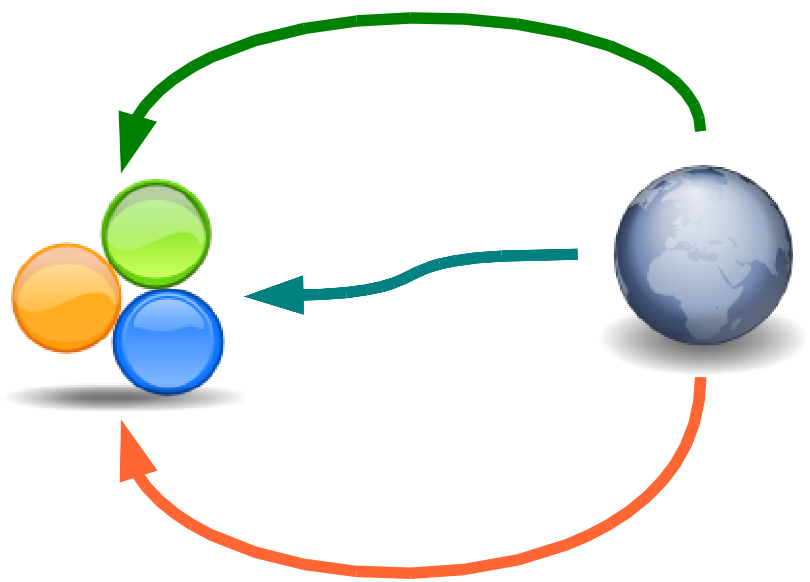
Alessandria, January. 17th

Reasoning on web services with choreographies and capabilities

Matteo Baldoni, Cristina Baroglio, Alberto Martelli, Viviana Patti, Claudio Schifanella

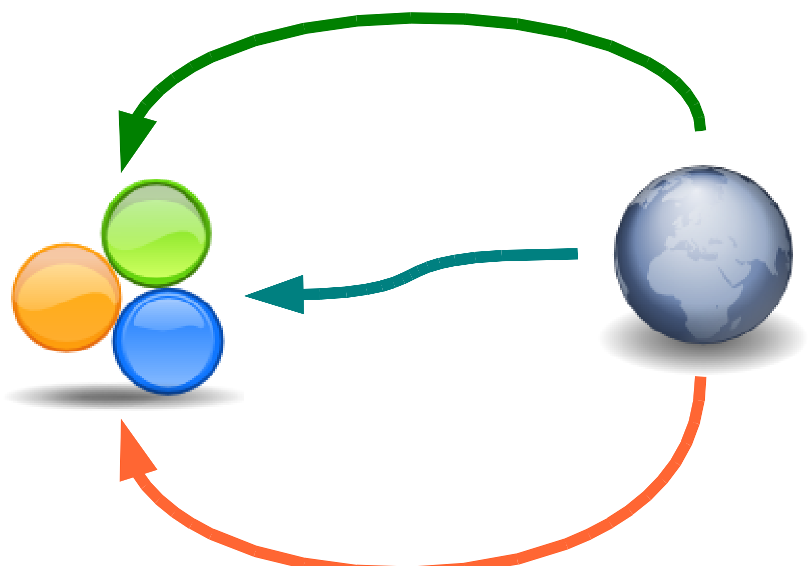
Introduction

- The platform-independent nature of services is a stimulus to develop new business processes by combining existing entities, enabling component *re-use*



Introduction

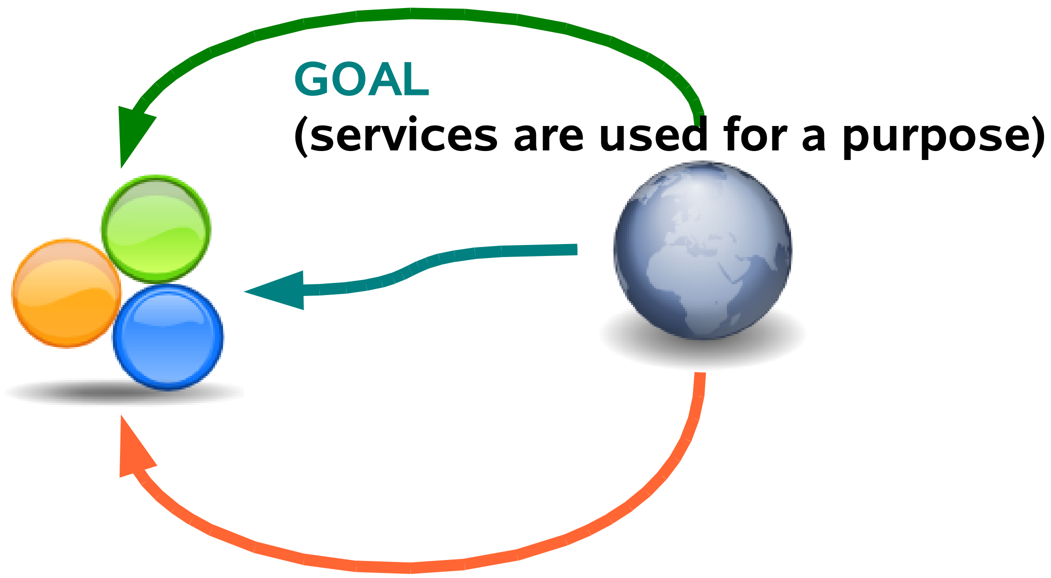
- The platform-independent nature of services is a stimulus to develop new business processes by combining existing entities, enabling component *re-use*



CONFORMANCE
(ability to interact according to a given schema)
[*ICSOC'06*]

Introduction

- The platform-independent nature of services is a stimulus to develop new business processes by combining existing entities, enabling component *re-use*



Goal-driven decision

- Participation to an interaction is often the result of a decision process driven by a goal condition:

a service provider allows the participation of a service to the interaction specified by a choreography given that, after the execution of its service, a goal condition holds

- More simply, we say that the service decides to take a role if it can achieve a goal

Perspective

- It would be nice to have
  - an abstract specification of the desired composite system plus
  - algorithms to decide if some existing services correspond to this specification
- Choreographies have opened new perspectives on the representation of abstract specifications but *the idea of using an abstract specification as a model for guiding the selection and composition of services is still embryonic*

⬇

- Indeed a choreography is a sort of contract; given that it contains sufficient information, by **reasoning** on it, it is possible to decide if a service should play a role

Perspective

- It would be nice to have
  - an abstract specification of the desired composite system plus
  - algorithms to decide if some existing services correspond to this specification
- Choreographies have opened new perspectives on the representation of abstract specifications but *the idea of using an abstract specification as a model for guiding the selection and composition of services is still embryonic*

⬇

- Indeed a choreography is a sort of contract; given that it contains sufficient information, by **reasoning** on it, it is possible to decide if a service should play a role

Perspective

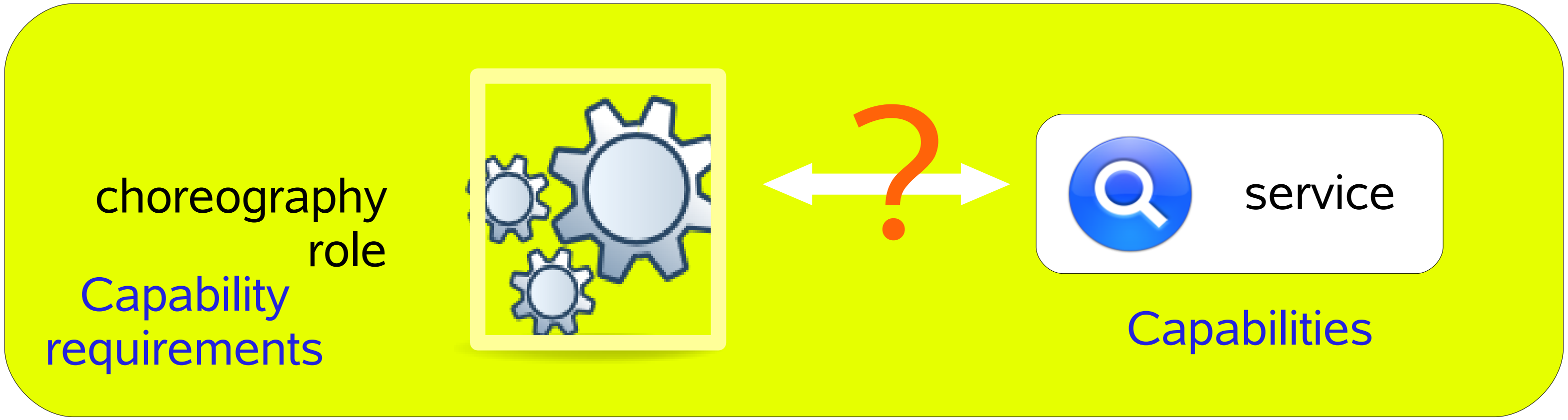
- It would be nice to have
  - an abstract specification of the desired composite system plus
  - algorithms to decide if some existing services correspond to this specification
- Choreographies have opened new perspectives on the representation of abstract specifications but *the idea of using an abstract specification as a model for guiding the selection and composition of services is still embryonic*
- Indeed a choreography is a sort of contract; given that it contains sufficient information, by **reasoning** on it, it is possible to decide if a service should play a role

Two results

- **Result 1**: extension of choreographies with capability requirements so to enable if and how a service can implement a role, taking into account the service goal
- **Result 2**: on the same principle, selection of an appropriate interlocutor

In both cases sophisticated forms of matching taking into account the **choreography** are needed

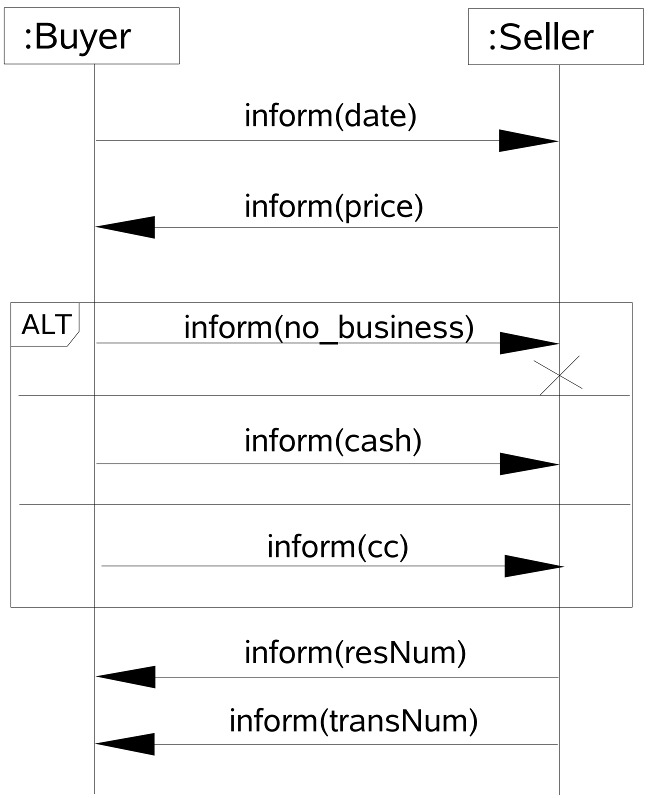
Role selection



- Let's focus on a single *choreography role* and on a *service* which might possibly play it
- In previous work [IJBPIM07] we have introduced:
  - capability requirements
  - capabilities

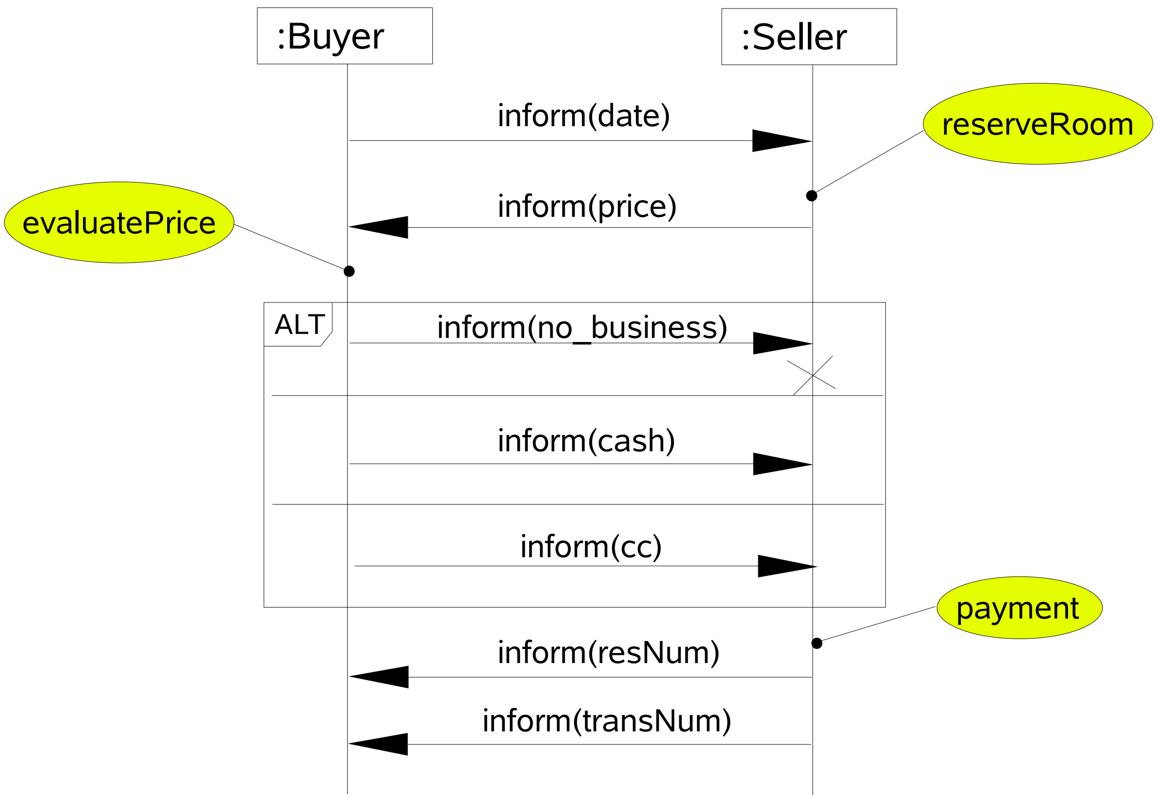
Extending choreographies: an example

A room reservation protocol

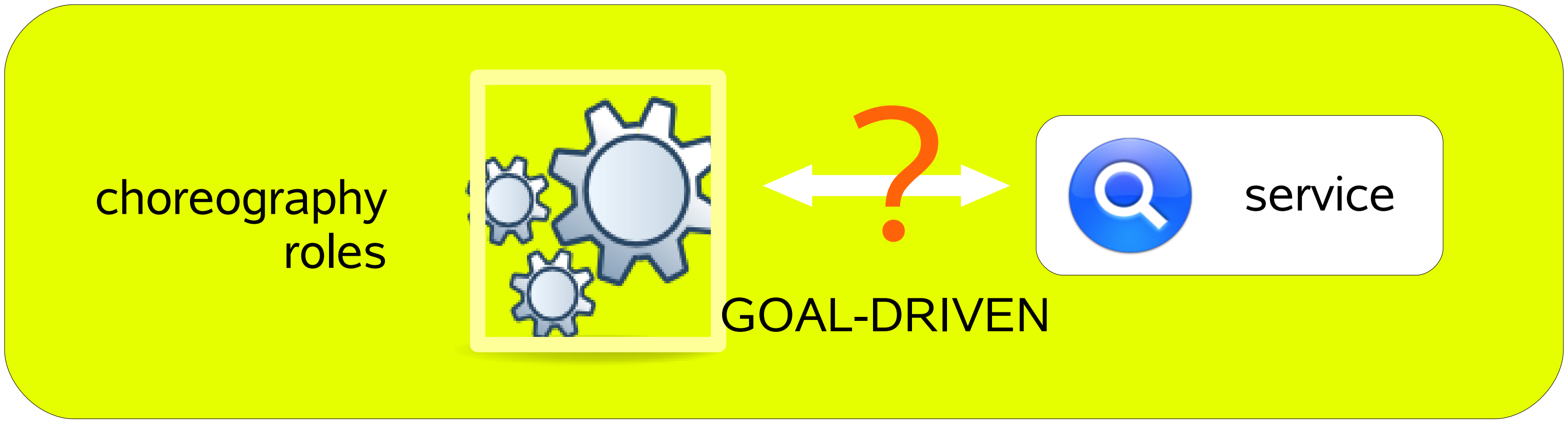


Extending choreographies: an example

A room reservation protocol

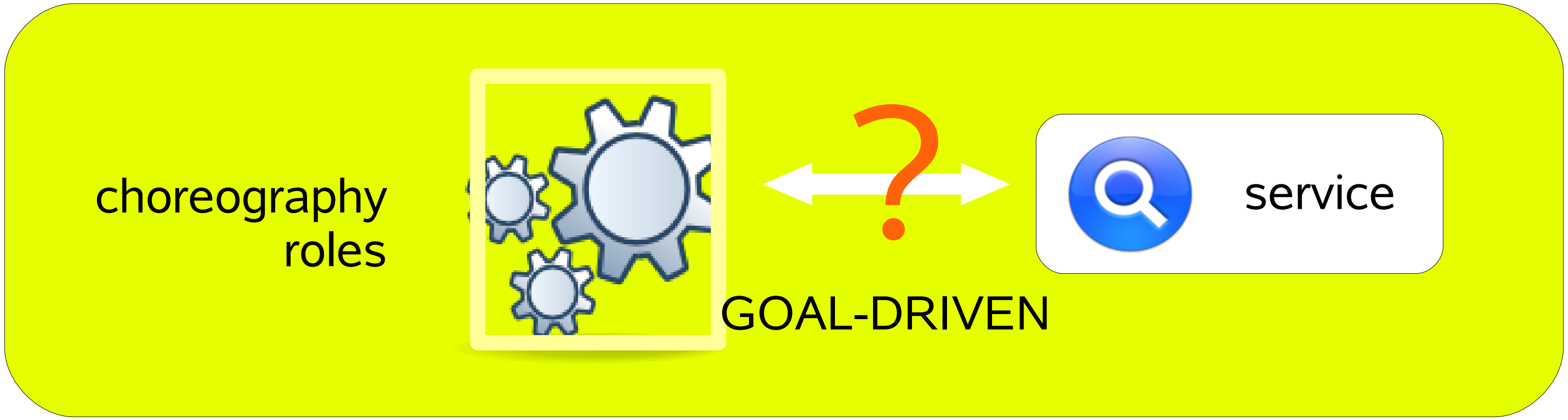


Role selection



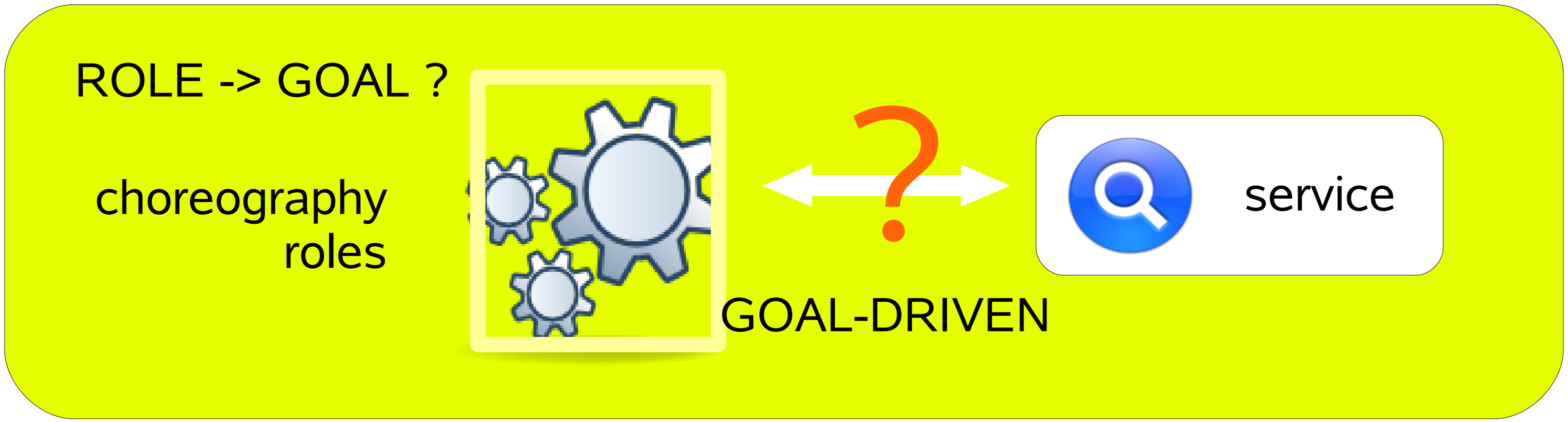
- The target is to decide if the service has the right "*capabilities*" for playing the role specification, eventually building an executable policy out of the role specification
- Goal-driven decision!

Role selection



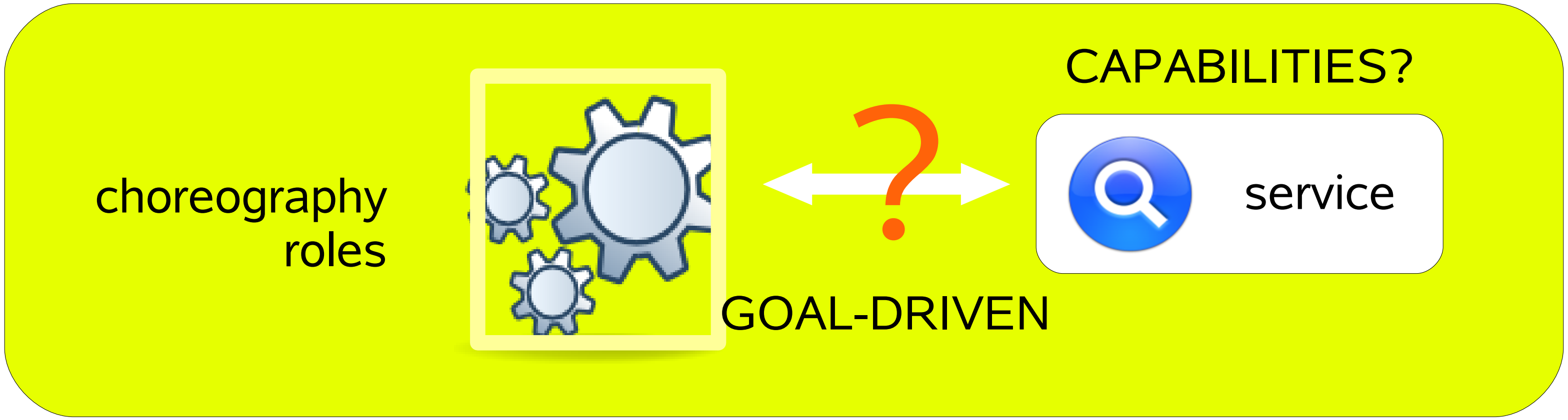
- Deliberation process:
  - verify if a choreography role allows the achievement of a goal
  - check if the service has the capabilities for playing the role
  - capabilities must be selected in a way that preserves the goal

Role selection



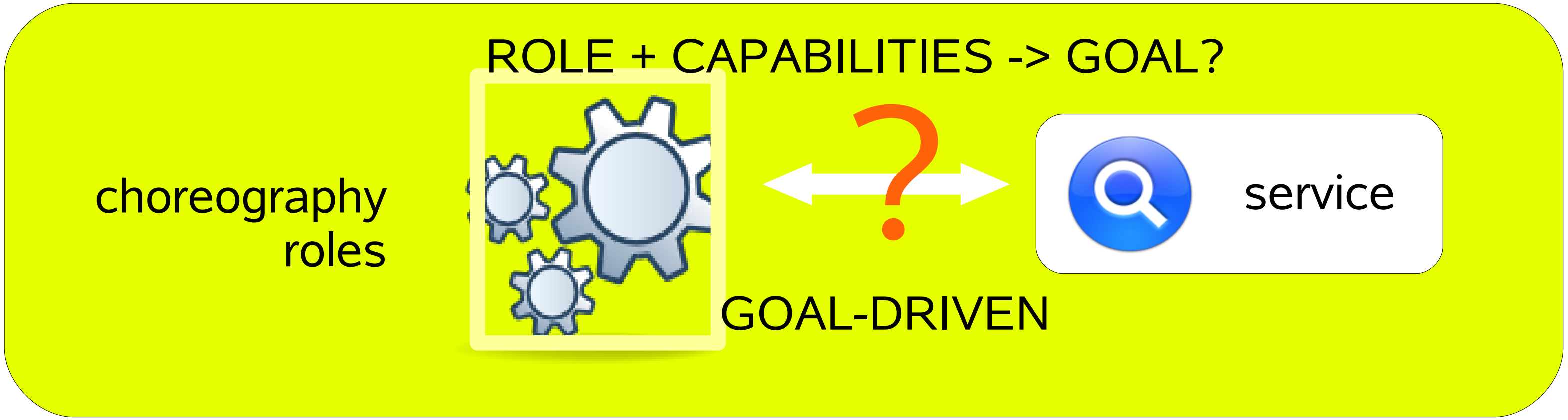
- Deliberation process:
  - verify if a choreography role allows the achievement of a goal
  - check if the service has the capabilities for playing the role
  - capabilities must be selected in a way that preserves the goal

Role selection



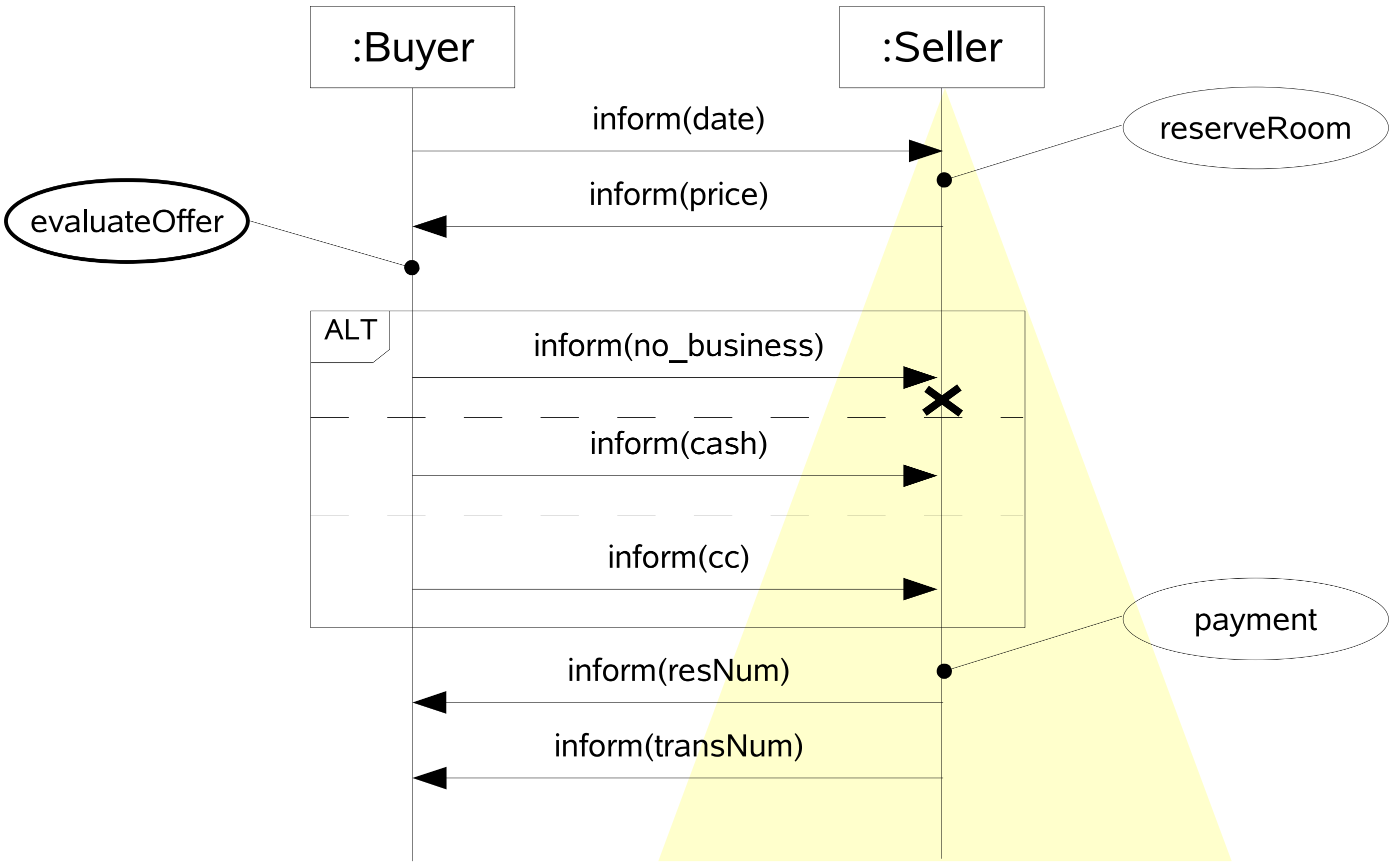
- Deliberation process:
  - verify if a choreography role allows the achievement of a goal
  - check if the service has the capabilities for playing the role
  - capabilities must be selected in a way that preserves the goal

Role selection



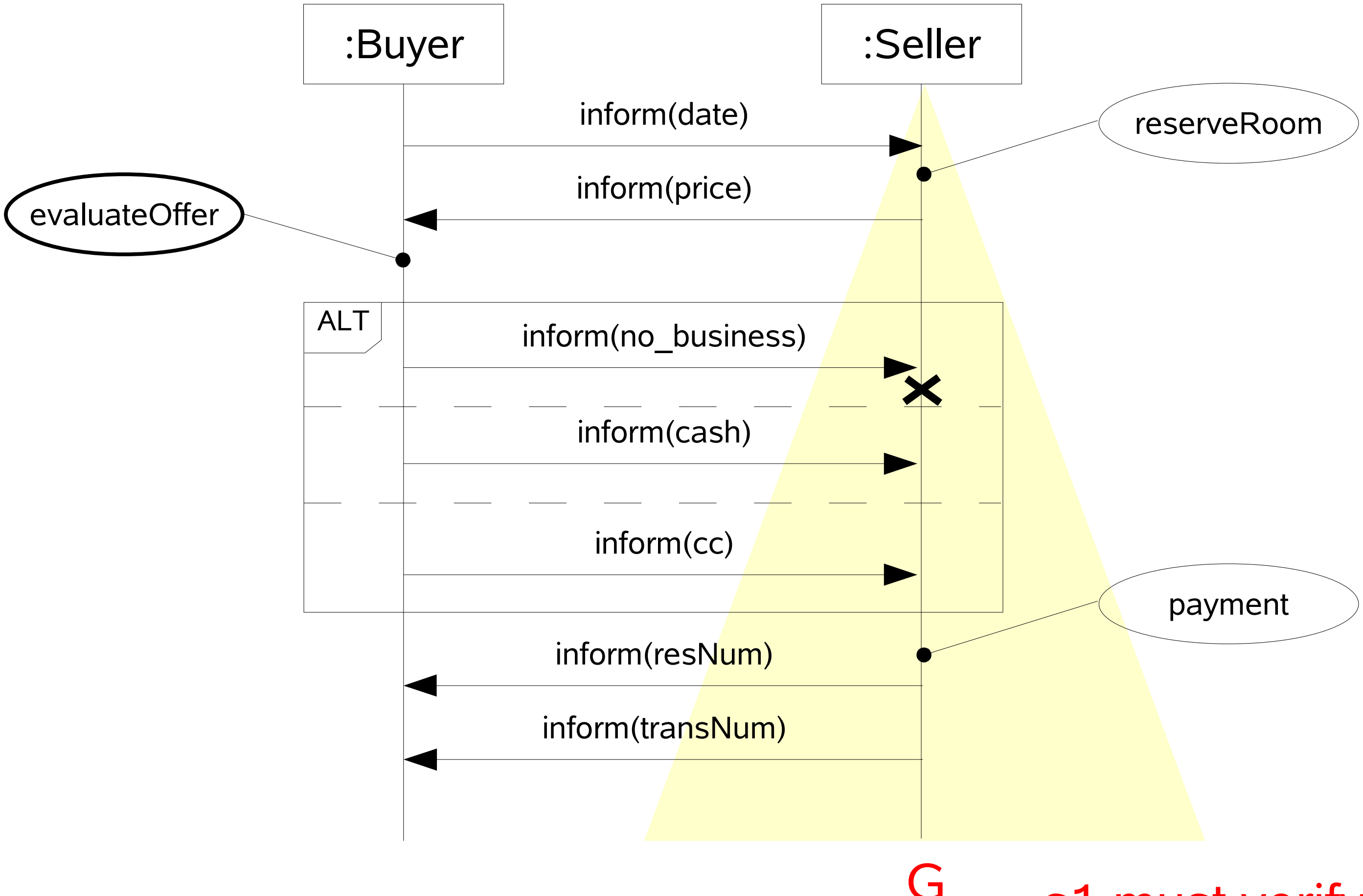
- Deliberation process:
  - verify if a choreography role allows the achievement of a goal
  - check if the service has the capabilities for playing the role
  - capabilities must be selected in a way that preserves the goal

The room reservation example



s1 wills to play as a Seller

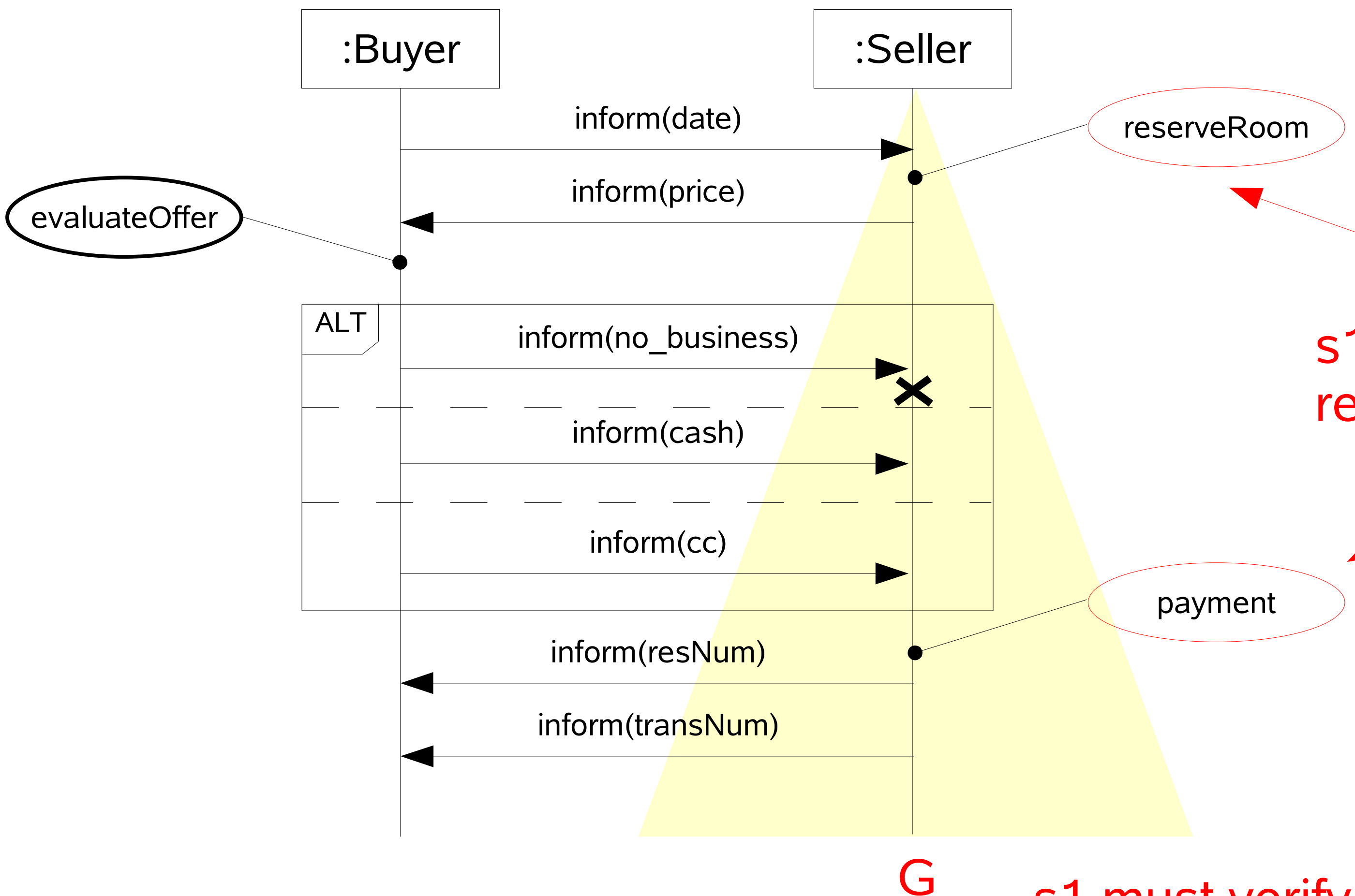
The room reservation example



s1 wills to play as a Seller

s1 must verify if the role specification allows to satisfy the goal G

The room reservation example

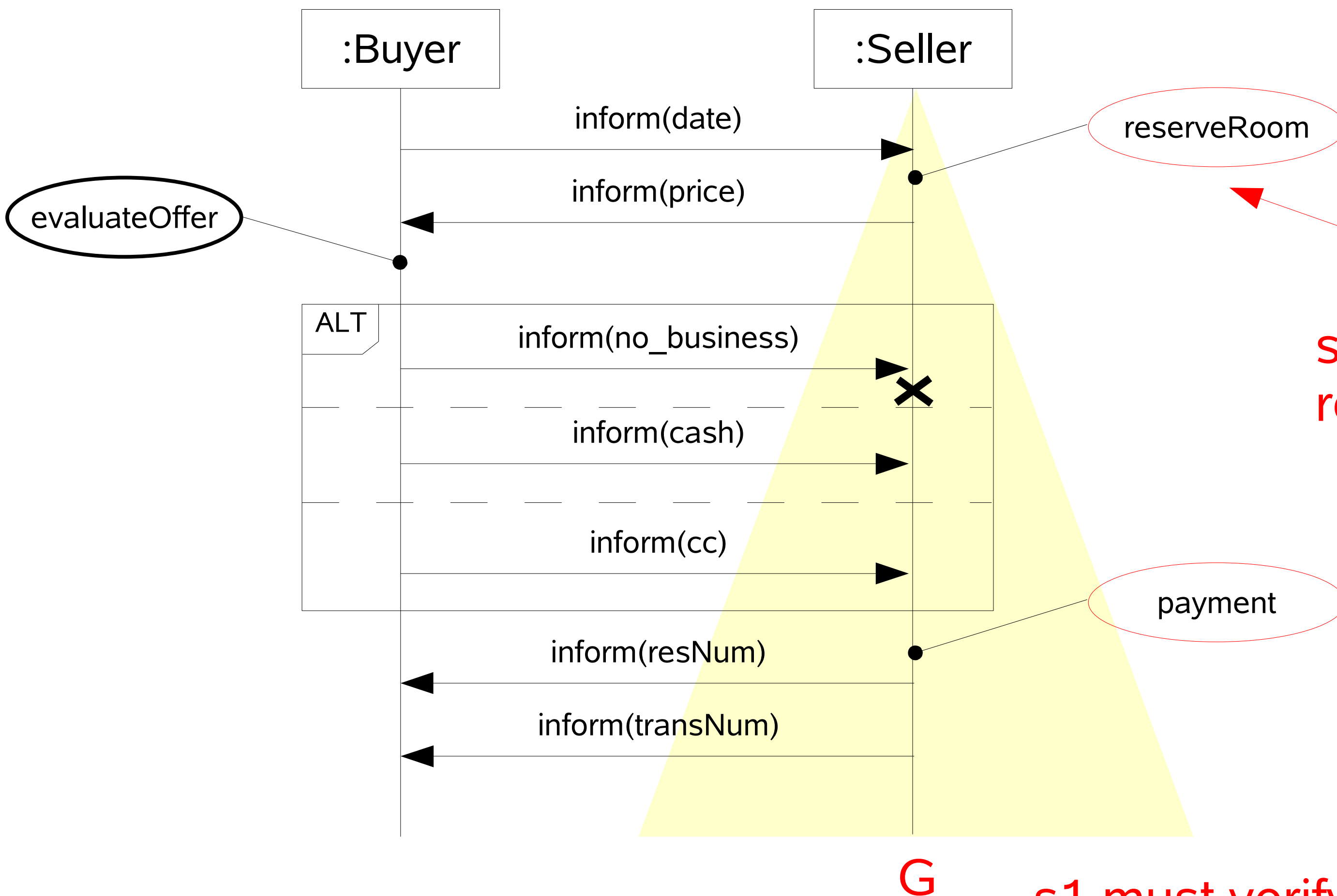


s1 wills to play as a Seller

s1 must substitute the required capabilities

s1 must verify if the role specification allows to satisfy the goal G

The room reservation example



s1 wills to play as a Seller

s1 must substitute the required capabilities

s1 must verify if the role specification allows to satisfy the goal G

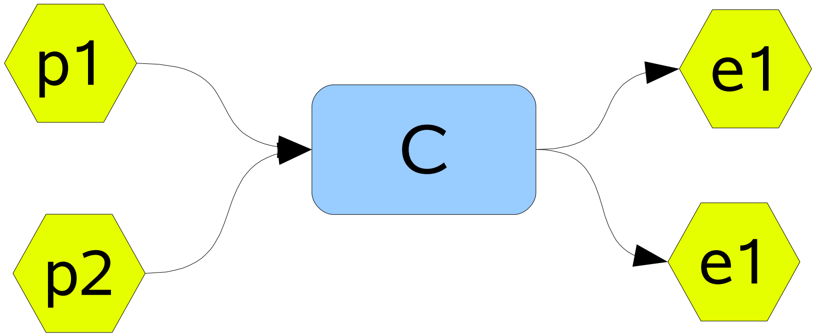
G will be preserved after the substitution?

How to specify capabilities?

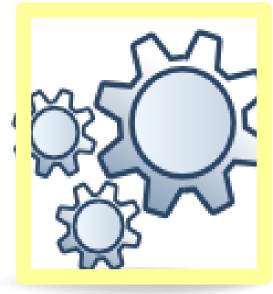
- Many alternatives! We follow an action metaphor:

  Capability: action
  - preconditions
  - effects

- Motivation: reasoning about the consequences of actions is adequate for a goal-driven deliberation process

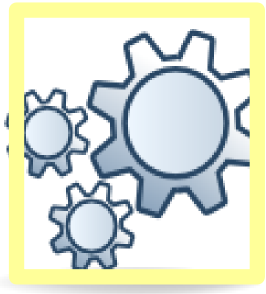


Formalization: role description



$$R_d = \langle S_A, G_A, CR, P \rangle$$

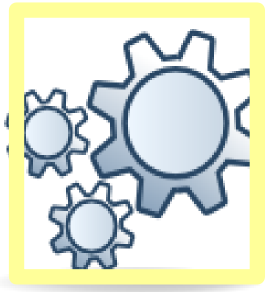
Formalization: role description



$$R_d = \langle S_A, G_A, CR, P \rangle$$

communicative
actions

Formalization: role description

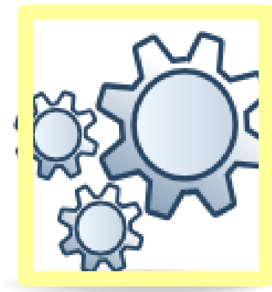


$$R_d = \langle S_A, G_A, CR, P \rangle$$

communicative actions

receive msg actions

Formalization: role description



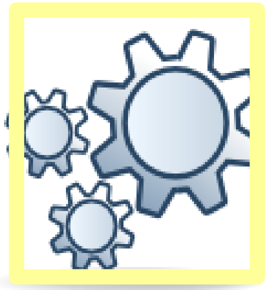
$$R_d = \langle S_A, G_A, CR, P \rangle$$

communicative actions

receive msg actions

Capability Requirements

Formalization: role description



$$R_d = \langle S_A, G_A, CR, P \rangle$$

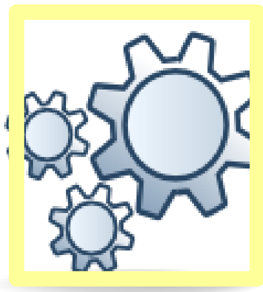
communicative actions

receive msg actions

Capability Requirements

procedure clauses

Formalization: role description



$$R_d = \langle S_A, G_A, CR, P \rangle$$

Every capability requirement *cr* is defined by its preconditions and effects:

cr **causes** $\{E_1, \ldots, E_n\}$

cr **possible if** $\{P_1, \ldots, P_t\}$

Reaching a goal

- Given a role description and an initial state, it is possible to verify if it is possible to reach a state where a goal G holds
- If the answer is positive, an execution trace leading to such a state is returned:

$$(\langle S_A, G_A, CR, P\rangle, S_0) \vdash G \; w.a. \; \sigma$$

ROLE DESCRIPTION

GOAL

EXEC. TRACE

INITIAL STATE

Reaching a goal

- Given a role description and an initial state, it is possible to verify if it is possible to reach a state where a goal G holds
- If the answer is positive, an execution trace leading to such a state is returned:

$$(\langle S_A, G_A, CR, P \rangle, S_0) \vdash G\, w.a.\, \sigma$$

- Useful to decide whether taking a role in a given choreography

Formalization: policy description

service

$$P_d = \langle S_A, G_A, C, P \rangle$$

Analogous to a role description: the actual capabilities of a service substitute the capability requirements

Capabilities are defined as:

c **causes** $\{E_1, \ldots, E_n\}$

c **possible if** $\{P_1, \ldots, P_t\}$

Substitutions and goals

- Given a set C of capabilities we can turn a role description into a policy description by applying a substitution $\theta = [C / CR]$

$$R_d = \langle S_A, G_A, CR, P \rangle \xrightarrow{\theta} P_d = \langle S_A, G_A, C, P\theta \rangle$$

- Given that

$$(\langle S_A, G_A, CR, P \rangle, S_0) \vdash G \text{ w.a. } \sigma$$

Substitutions and goals

- Given a set *C* of capabilities we can turn a role description into a policy description by applying a substitution $\theta=[C/CR]$

$$R_d=\langle S_A, G_A, CR, P\rangle \xrightarrow{\theta} P_d=\langle S_A, G_A, C, P\theta\rangle$$

- Given that

$$(\langle S_A, G_A, CR, P\rangle, S_0) \vdash G \text{ w.a. } \sigma$$

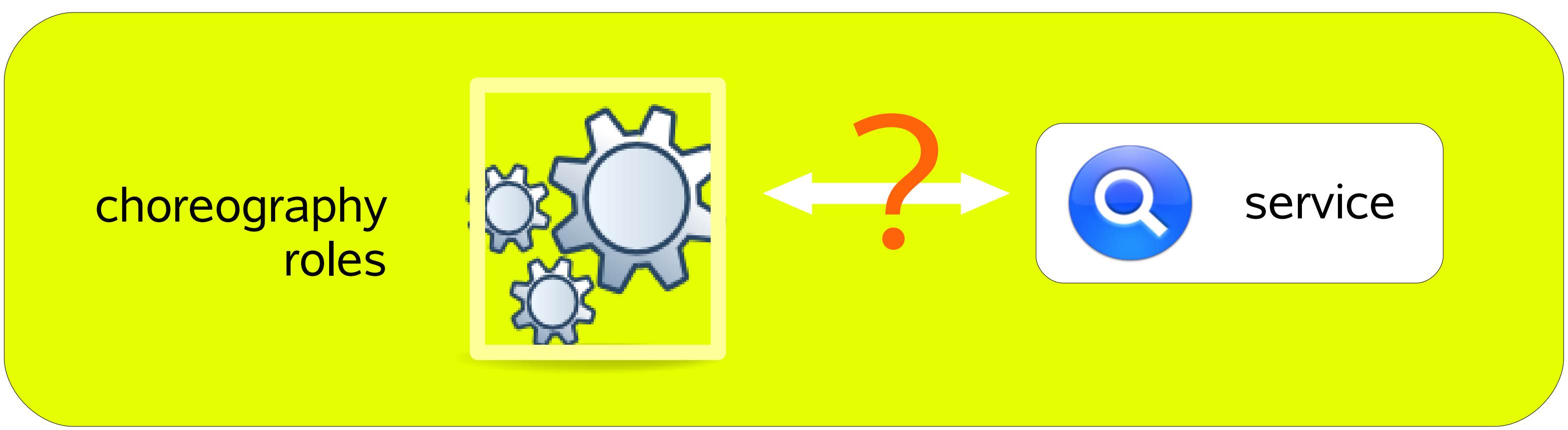
- Can the goal be achieved also after the substitution?

$$(\langle S_A, G_A, C, P\theta\rangle, S_0) \vdash G \text{ w.a. } \sigma\theta \quad ?$$

Substitutions and goals

- Given a set C of capabilities we can turn a role description into a policy description by applying a substitution $\theta = [C / CR]$

$$R_d = \langle S_A, G_A, CR, P \rangle \xrightarrow{\theta} P_d = \langle S_A, G_A, C, P\theta \rangle$$



ROLE -> GOAL $\Rightarrow$ ROLE + CAPABILITIES -> GOAL?

Reasoning on capabilities

- More in general:

$$\exists \theta = [C/CR] \, s.t. \, (\langle S_A, G_A, CR, P \rangle, S_0) \vdash G \Rightarrow$$

$$(\langle S_A, G_A, C, P\theta \rangle, S_0) \vdash G$$

- Must an entity have **all** the capabilities required for a role?

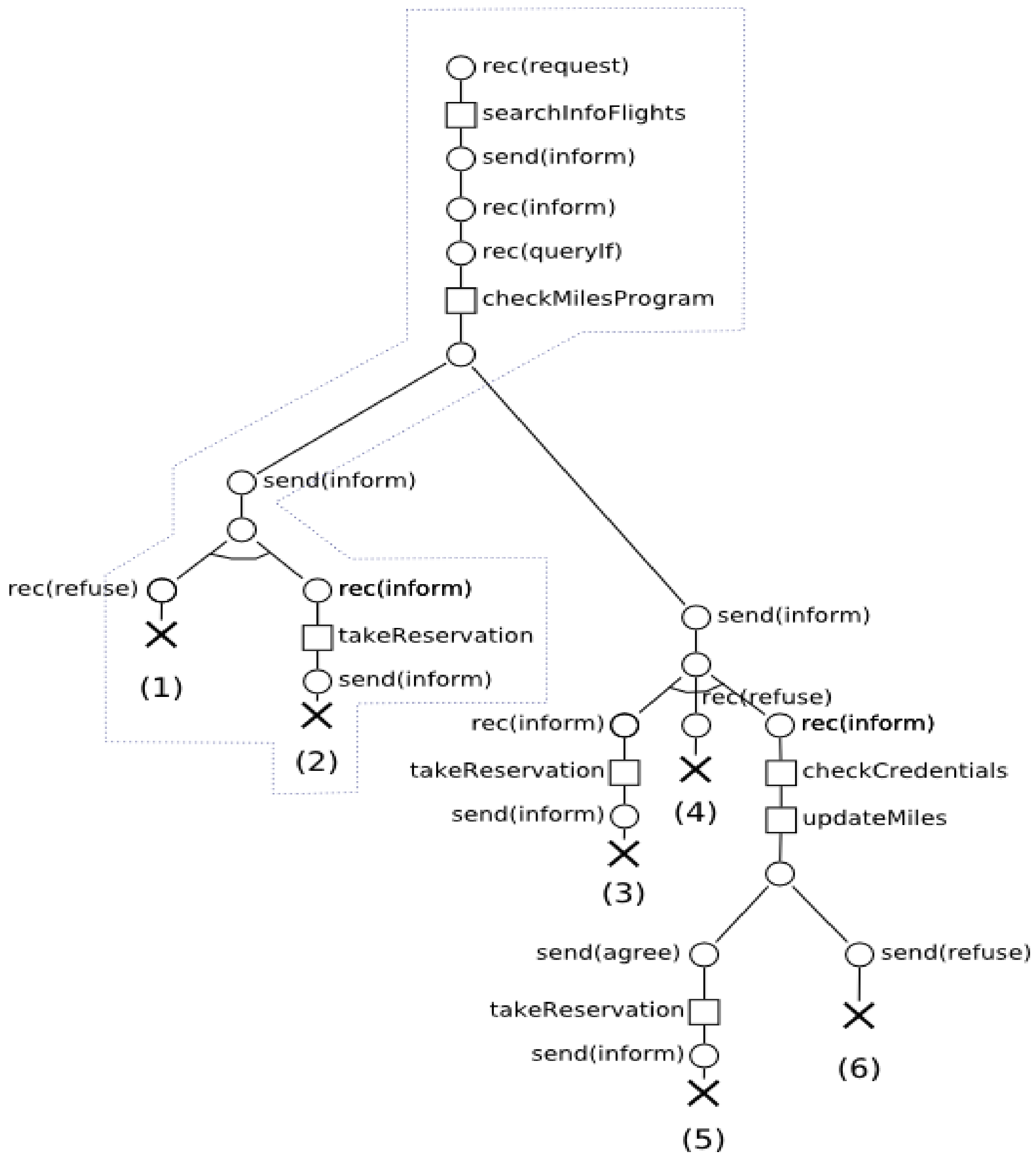
$$\exists \sigma, \theta' = [C/CR_\sigma], CR_\sigma \subseteq CR \, s.t.$$

$$(\langle S_A, G_A, CR, P \rangle, S_0) \vdash G \, w.a. \, \sigma \Rightarrow$$

$$(\langle S_A, G_A, C, P\theta' \rangle, S_0) \vdash G \, w.a. \, \sigma\theta'$$

Reasoning on capabilities

- A seller wants to sell some tickets
- But it can only handle credit card payments



Reasoning on capabilities

- Moreover, the set of capabilities of an entity could depends on the context

$$\exists \sigma, \theta'' = [C'/CR_\sigma], C' \subseteq C, CR_\sigma \subseteq CR \; s.t.$$

$$(\langle S_A, G_A, CR, P \rangle, S_0) \vdash G \; w.a. \; \sigma \Rightarrow$$

$$(\langle S_A, G_A, C', P\theta'' \rangle, S_0) \vdash G \; w.a. \; \sigma\theta''$$

Related publications

- M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. *“Interaction Protocols and Capabilities: A Preliminary Report”*. In J. J. Alferes, J. Bailey, W. May, and U. Schwertel, editors, Post-Proc. of the Fourth Workshop on Principles and Practice of Semantic Web Reasoning, PPSWR 2006, volume 4187 of LNCS, pages 63-77. Springer, 2006
- M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. *“The Need of Capability Requirements Inside Choreographies and Interaction Protocols”*. In Y. Yan and L. Zhang, editors, Proc. of the 2006 International Workshop on Service Oriented Techniques (SOT06), pages 17-24, August 2006
- M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. *“Reasoning on choreographies and capability requirements”*. International Journal of Business Process Integration and Management, IJBPIM, 2(4), 2007

Flexibility

- θ can be any kind of association between the operations provided by a service with the operations described in the choreography
- It is the result of a matching process
- It is unlikely that an existing operation perfectly matches a specification it wasn't designed for
- Some degree of flexibility in the matching process is needed (same perspective as in semantic matchmaking)
- Flexible match is required in order to enhance software reuse

Zaremski and Wing Match rules

- Zaremski and Wing propose a formal specification to describe the behavior of software components [*Specification matching of software components, ACM 1997*]
- This work allows to determine if two software components match
- Software components are described by means of preconditions and effects
- Five degrees of relaxed matches

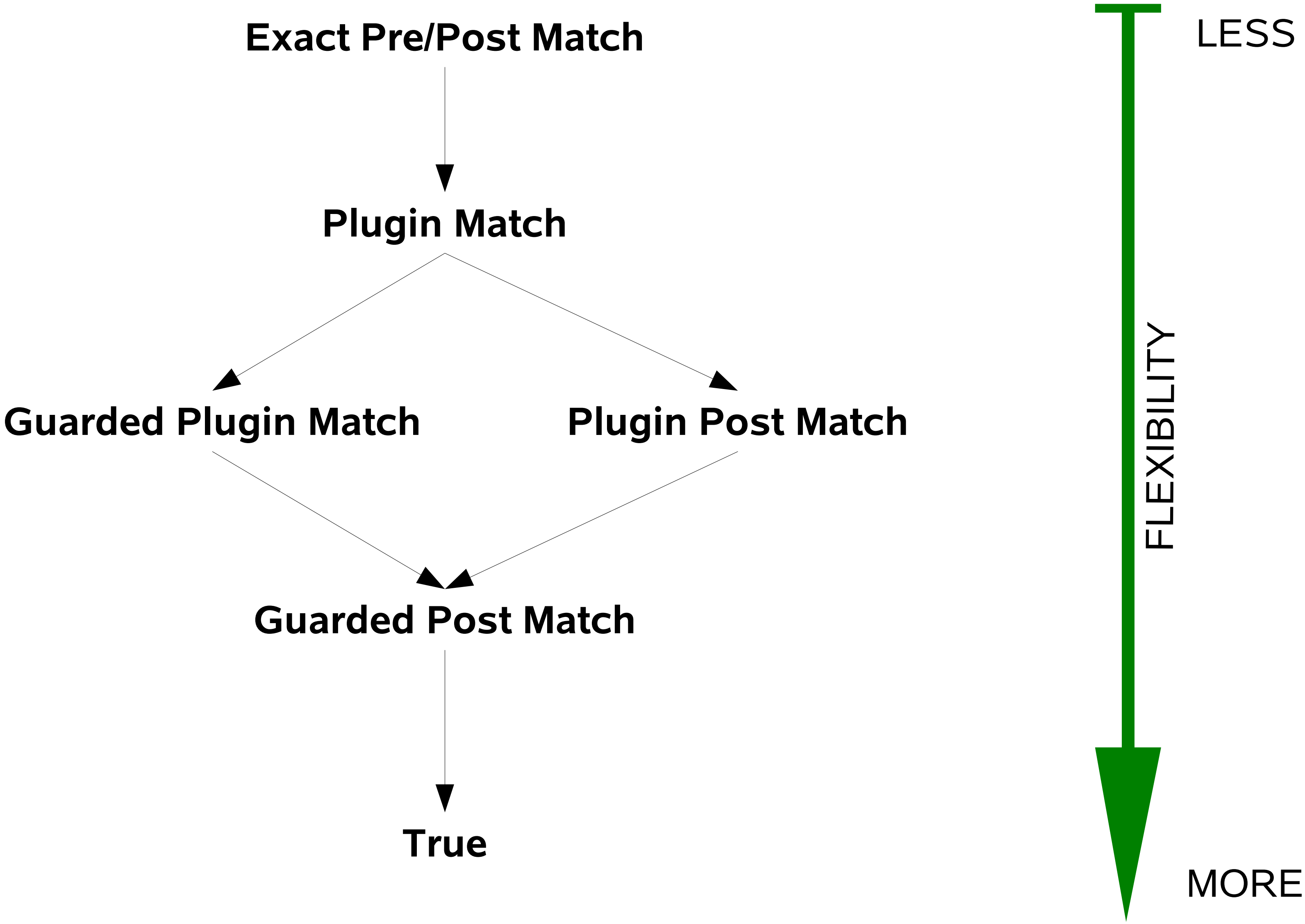
Zaremski and Wing Match rules

EM	• Exact pre/post match:	$R_{pre} \Leftrightarrow S_{pre} \wedge R_{post} \Leftrightarrow S_{post}$
PIM	• Plugin match:	$R_{pre} \Rightarrow S_{pre} \wedge S_{post} \Rightarrow R_{post}$
POM	• Plugin Post match:	$S_{post} \Rightarrow R_{post}$
GPIM	• Guarded Plugin match:	$R_{pre} \Rightarrow S_{pre} \wedge (S_{pre} \wedge S_{post}) \Rightarrow R_{post}$
GPOM	• Guarded Post match:	$(S_{pre} \wedge S_{post}) \Rightarrow R_{post}$

R = Requirement
S = Software component

pre and post are logic formulae

Zaremski and Wing lattice



Zaremski and Wing Match rules

EM
- Exact pre/post match: $Pr(c_r) = Pr(c) \wedge Ef(c_r) = Ef(c)$

PIM
- Plugin match: $Pr(c_r) \supseteq Pr(c) \wedge Ef(c) \supseteq Ef(c_r)$

POM
- Plugin Post match: $Ef(c) \supseteq Ef(c_r)$

GPIM
- Guarded Plugin match: $Pr(c_r) \supseteq Pr(c) \wedge ((Pr(c) \cup Ef(c)) \supseteq Ef(c_r))$

GPOM
- Guarded Post match: $((Pr(c) \cup Ef(c)) \supseteq Pr(c_r))$

cr => Capability Requirement
c => Capabilities!

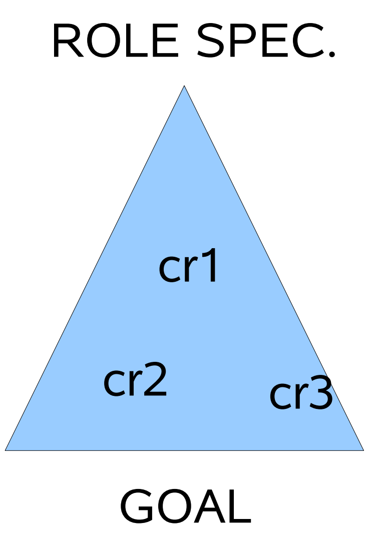
pre and post are logic formulae

Flexibility + goal

- Problem: these matches compare a single operation description to a single requirement, they are *local*
- When a service adopts a goal to achieve a goal of interest, these matches, but EM, do not guarantee that after the substitution (of capabilities to capability requirements) the goal will be preserved

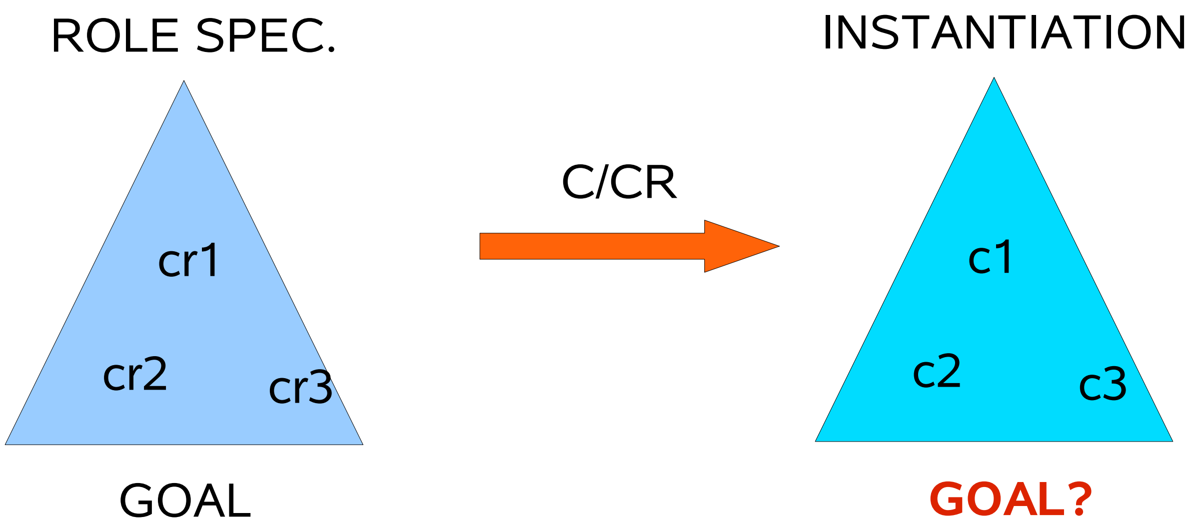
Flexibility + goal

- Problem: these matches compare a single operation description to a single requirement, they are local
- When a service adopts a role to achieve a goal of interest, these matches, but EM, *do not* guarantee that after the substitution (of capabilities to capability requirements) the goal will be preserved



Flexibility + goal

- Problem: these matches compare a single operation description to a single requirement, they are local
- When a service adopts a role to achieve a goal of interest, these matches, but EM, *do not* guarantee that after the substitution (of capabilities to capability requirements) the goal will be preserved



Flexibility + goal

- Problem: these matches compare a single operation description to a single requirement, they are local
- When a service adopts a role to achieve a goal of interest, these matches, but EM, *do not* guarantee that after the substitution (of capabilities to capability requirements) the goal will be preserved

Idea: to use constraints derived from the choreography, which defines the global execution context

Def - Conservative substitution

- Let $\langle S_A, G_A, CR, P \rangle$ be a role description, S_0 the initial state, and *G* the goal of interest. When the following relation holds:

$$\exists \sigma, \theta = [C / CR_\sigma], CR_\sigma \subseteq CR, s.t.$$

$$(\langle S_A, G_A, CR, P \rangle, S_0) \vdash G \text{ w.a. } \sigma \Rightarrow (\langle S_A, G_A, C, P\theta \rangle, S_0) \vdash G \text{ w.a. } \sigma\theta$$

where CR_σ is the set of *CR* that appear in σ, the substitution is *conservative*.

Theorem

The substitutions

$$\theta_{PIM}$$

$$\theta_{POM}$$

$$\theta_{GPIM}$$

$$\theta_{GPOM}$$

are not conservative

Theorem

The substitutions

θ_{PIM}

θ_{POM}

θ_{GPIM}

θ_{GPOM}

are not conservative

PROOF BY COUNTER-EXAMPLE

capability requ.	cr1 causes {*B*l1} cr1 possible if true
speech act	a causes {*B*l2} a possible if {*B*l1, *B*l3}
policy:	p isp cr1, a

ROLE DESCR.

Theorem

The substitutions

θ_{PIM}

θ_{POM}

θ_{GPIM}

θ_{GPOM}

are not conservative

PROOF BY COUNTER-EXAMPLE

capability requ.	cr1 causes {*B*l1} cr1 possible if true
speech act	a causes {*B*l2} a possible if {*B*l1, *B*l3}
policy:	p isp cr1, a

ROLE DESCR.

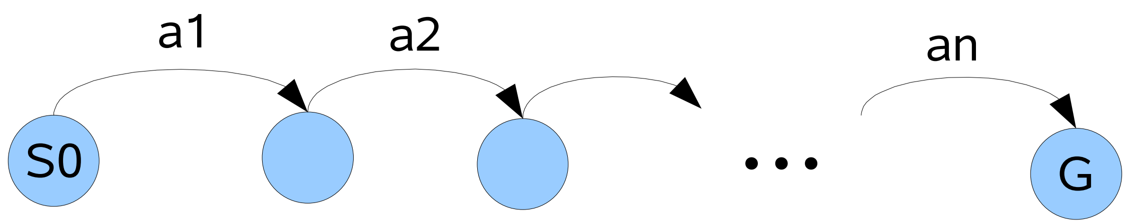
capability	c1 causes {*B*l1, ¬ *B*l3} c1 possible if true
initial state goal	S_0=Bl3 G=Bl2

c1 matches cr1 according to any of the listed flexible matches !!!

'a' cannot be applied anymore!!
Preconditions not satisfied
due to the **additional effect** of c1

Towards a conservative *plugin* match

- To overcome the problem, let's consider also the overall structure of the solution, and focus on causal chains

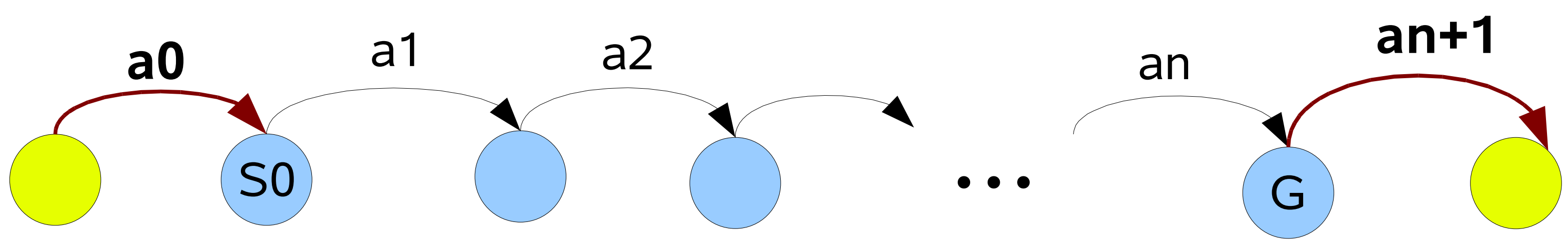


$$\sigma = a_1 ; a_2 ; \ldots ; a_n$$

SUCCESSFUL EXECUTION TRACE
OBTAINED BY REASONING ON THE ROLE DESCR.

Towards a conservative *plugin* match

- To overcome the problem, let's consider also the overall structure of the solution, and focus on causal chains



a_0 possible if true
a_0 causes S_0

FICTITIOUS ACTION

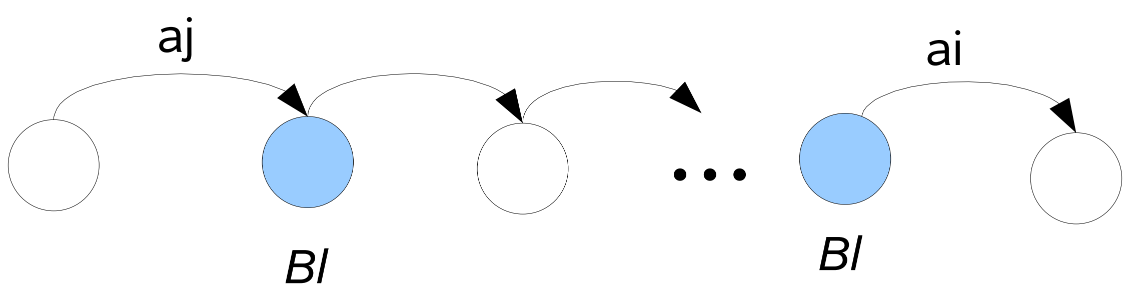
$$\bar{\sigma} = a_0 ; a_1 ; a_2 ; \ldots ; a_n ; a_{n+1}$$

a_0 possible if G
a_0 causes true

FICTITIOUS ACTION

Def - action dependency

- Consider two indexes *i* and *j*, *j* < *i*, we say that a_i depends on a_j for the fluent *Bl* iff *Bl* is an effect of a_j, Bl is a precondition to a_i, and there is no *k*, j<*k*<*i*, s.t. *Bl* is an effect of a_k



aj causes ... *Bl* ...

ai possible if ... *Bl* ...

$$a_j \rightsquigarrow_{\langle Bl, \bar{\sigma} \rangle} a_i$$

Def – dependency set

- Dependency set of *Bl*

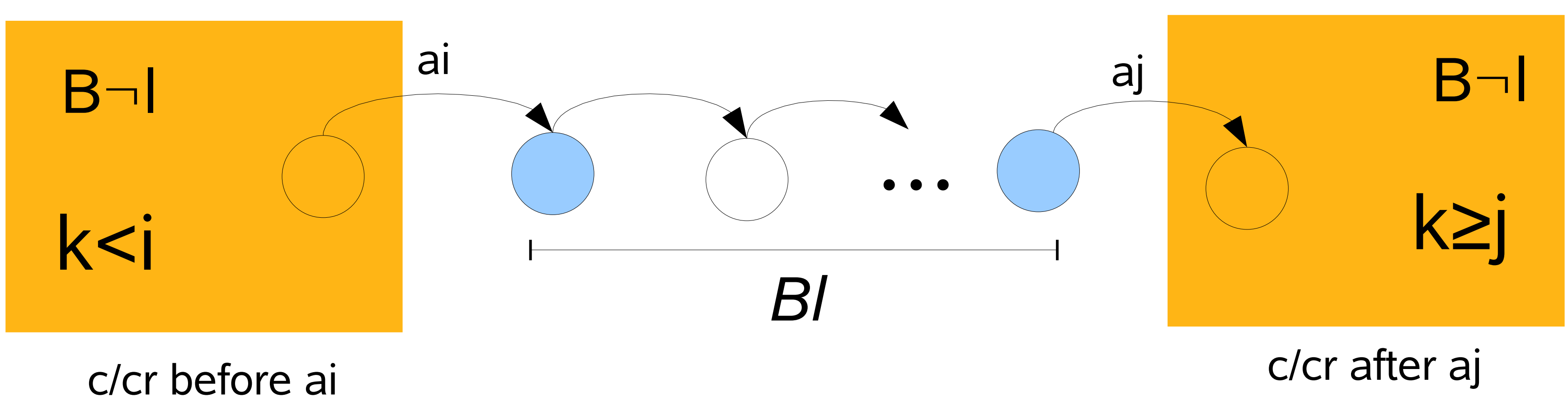
$$Deps(Bl) = \{(j, i) \mid a_j \rightsquigarrow_{<Bl, \overline{\sigma}>} a_i\}$$

The dependency set of a fluent is the set of pairs of indexes of actions, such that the second depends on the first for what concerns its precondition Bl

Uninfluential additional effect

- Consider a [*c*/*cr*] in a substitution θ_{PIM} and an effect *B¬l* of *c* which is *not* an effect of *cr*: *B¬l ∈ Effs(c) – Effs(cr)*

B¬l is an *uninfluential fluent*, *w.r.t.* $\sigma\theta_{PIM}$, iff:
∀ *(i, j)* ∈ *Deps*(Bl, σ), given that *k* is the position of *cr* in the execution trace, either *k<i* or *k ≥ j*

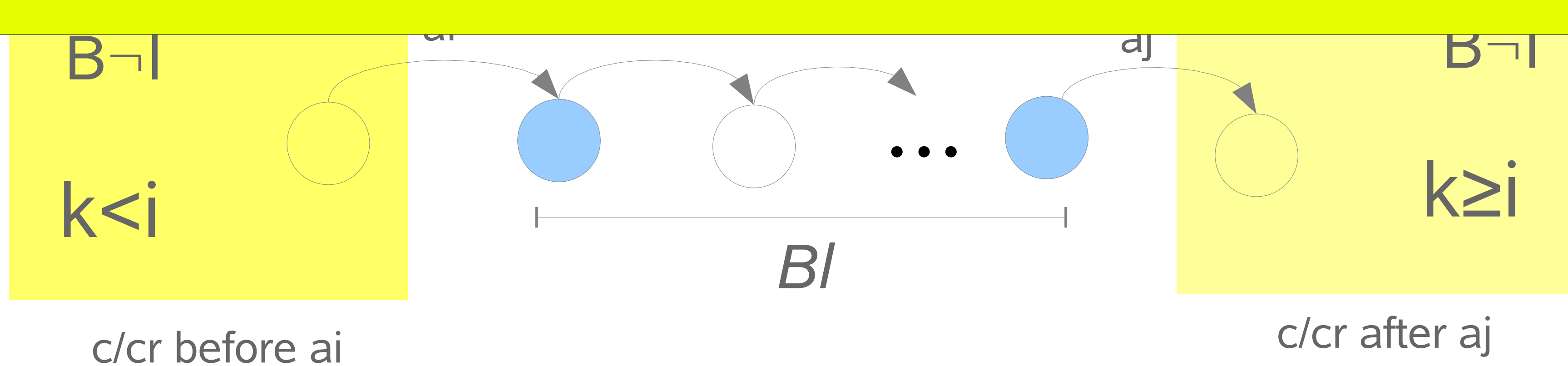


Def - Uninfluential substitution

- Consider a [*c*/*cr*] in a substitution θ_{PIM} and an effect *B*¬*l* of *c* which is *not* an effect of *cr*: *B*¬*l* ∈ *Effs(c) – Effs(cr)*

Bl is an *uninfluential fluent*, *w.r.t.* $\sigma\theta_{PIM}$, iff:

A **PIM-substitution** is **uninfluential** iff all the additional effects of the capabilities, that are substituted to capability requirements, are *uninfluential fluents*



Theorem (proof by absurd)

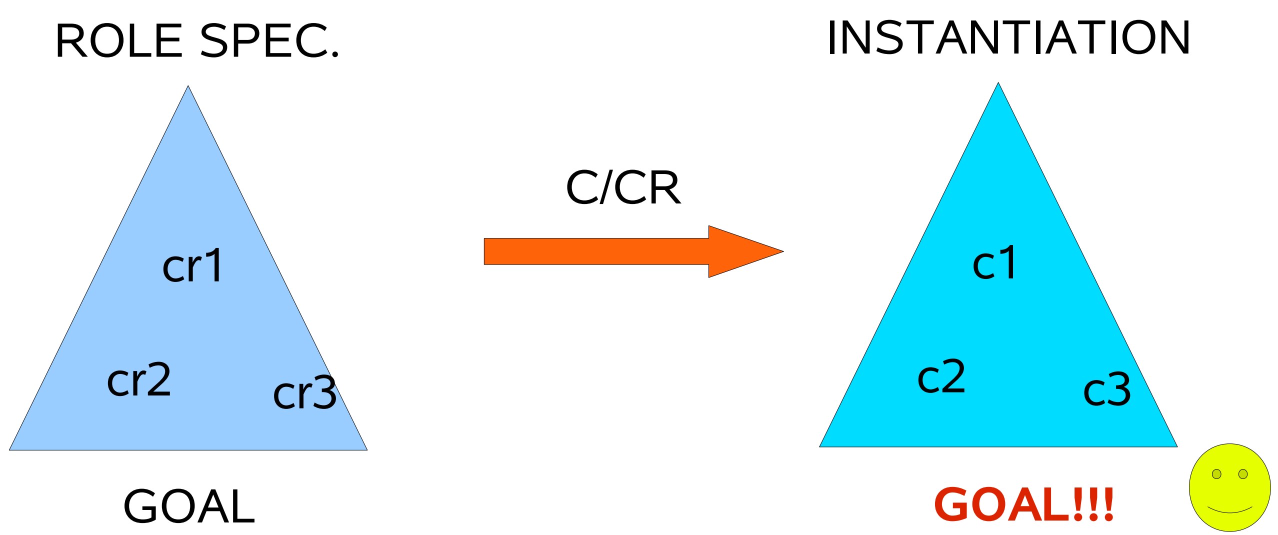
- Let G be a goal and $(\langle S_A, G_A, CR, P\rangle, S_0)$ a role description. If $(\langle S_A, G_A, CR, P\rangle, S_0) \vdash G$ *w.a.* σ and there is an uninfluential substitution $\theta_{PIM} = [C/CR_\sigma], CR_\sigma \subseteq CR$ then

$$(\langle S_A, G_A, C, P\theta_{PIM}\rangle, S_0) \vdash G \text{ w.a. } \sigma\theta_{PIM}$$

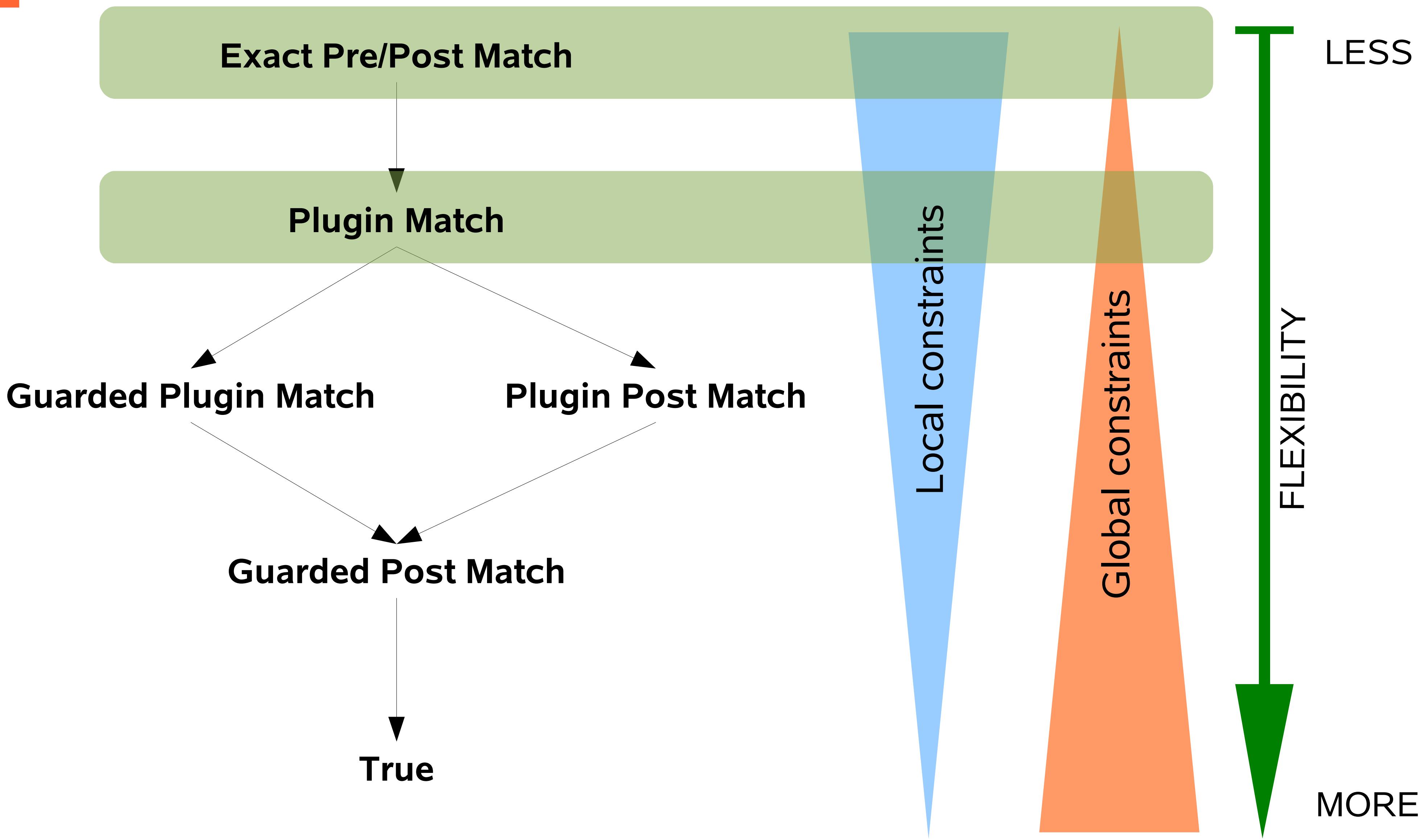
Theorem (proof by absurd)

- Let G be a goal and $(\langle S_A, G_A, CR, P\rangle, S_0)$ a role description. If $(\langle S_A, G_A, CR, P\rangle, S_0) \vdash G$ *w.a.* σ and there is an uninfluential substitution $\theta_{PIM} = [C/CR_\sigma], CR_\sigma \subseteq CR$ then

$$(\langle S_A, G_A, C, P\theta_{PIM}\rangle, S_0) \vdash G \text{ w.a. } \sigma\theta_{PIM}$$



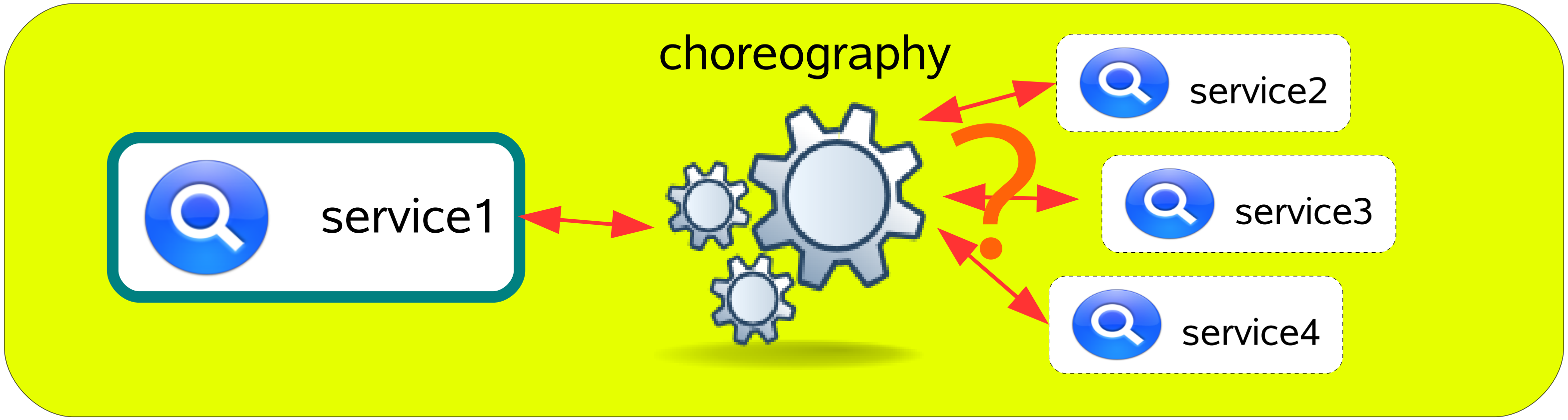
Zaremski and Wing lattice



Related publications

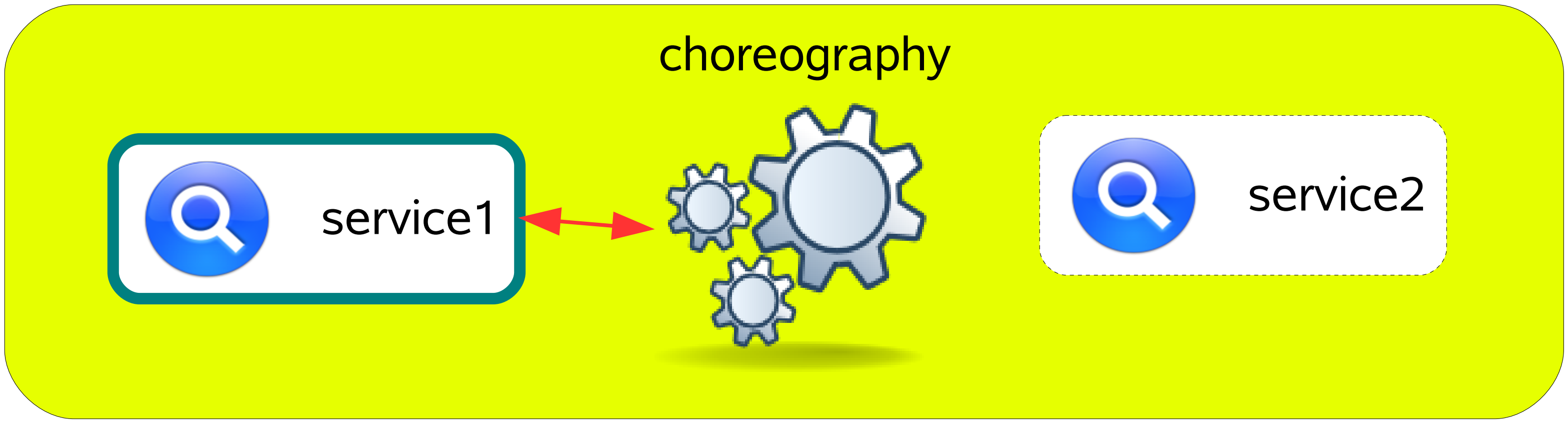
- M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. *"Goal preservation by choreography-driven matchmaking"*. In Proc. of the Third International Workshop on Engineering Service-Oriented Applications: Analysis, Design and Composition, WESOA 2007, in conjuction with ICSOC 2007, pages 77-88, Vienna, Austria, Sept. 2007.
- M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. *"Preserving player's goals: a choreography-driven matchmaking approach"*. In Proc. of WOA 2007, pages 132-139, Genova, Italy, Sept. 2007. Seneca Edizioni.

Service selection



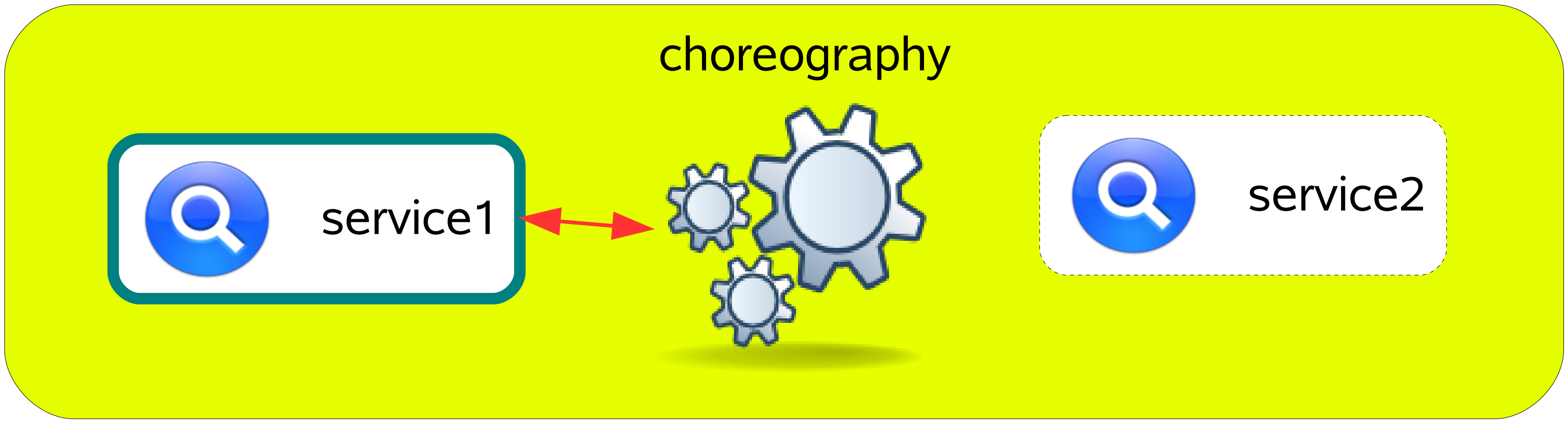
- The target is to retrieve a service interlocutor, which can play a given choreography role, preserving at the same time a condition of interest
- Goal-driven decision

Service selection



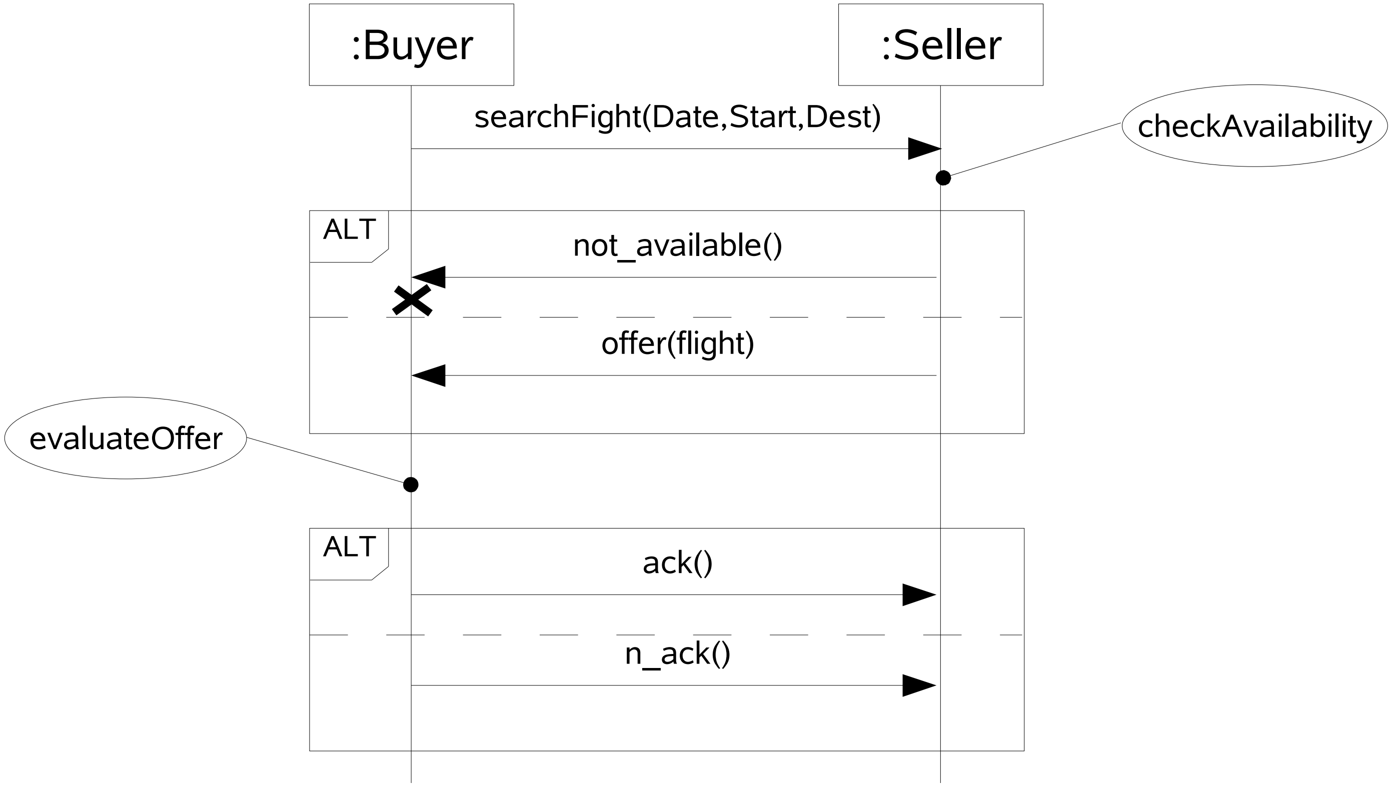
- service1 has verified that by playing role1 it can achieve its goal
- for animating the choreography it is necessary to find a partner that will play the other role
  - partner must conform to role2
  - partner offers operations that will be invoked by service1 that must (flexibly) match the specifications in the choreography

Service selection



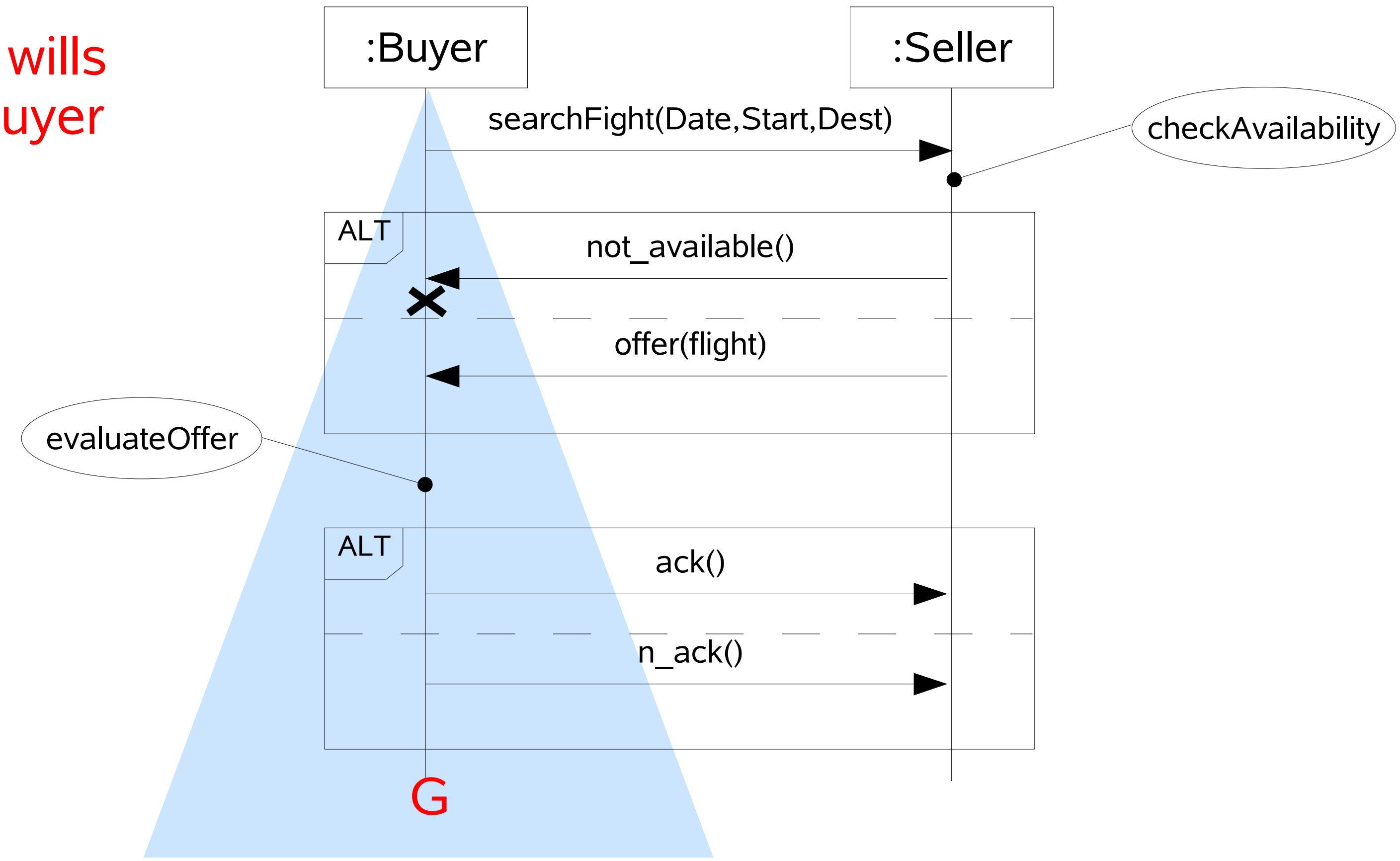
- let's suppose that service2 conforms to role2 (it is a candidate partner for service1)
- the operations offered by service2 match with the requirements of the choreography in a flexible way
- will the choice of service2 as partner invalidate the achievement of service1 goals?

The flight reservation example



The flight reservation example

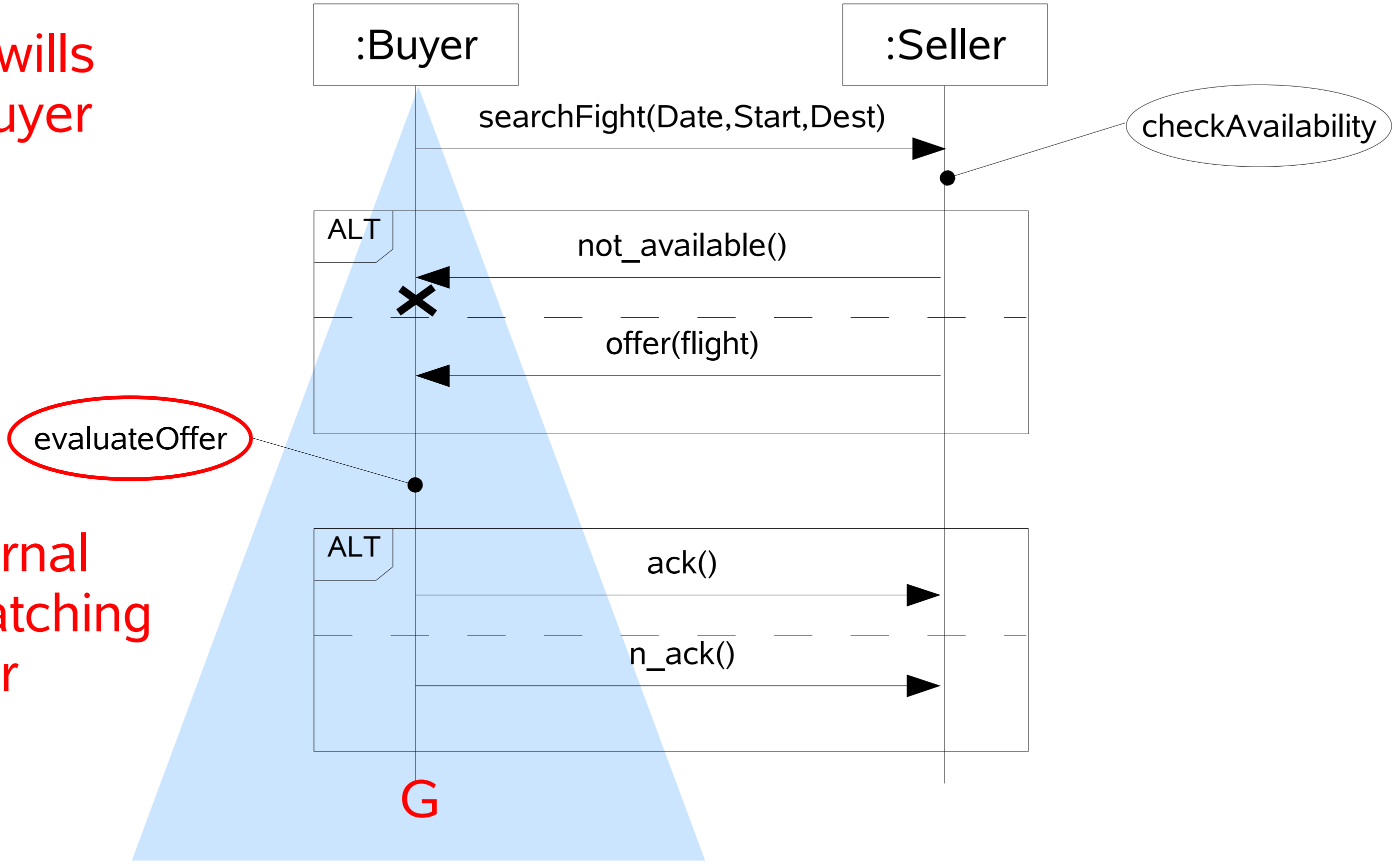
service1 wills
to play Buyer



The flight reservation example

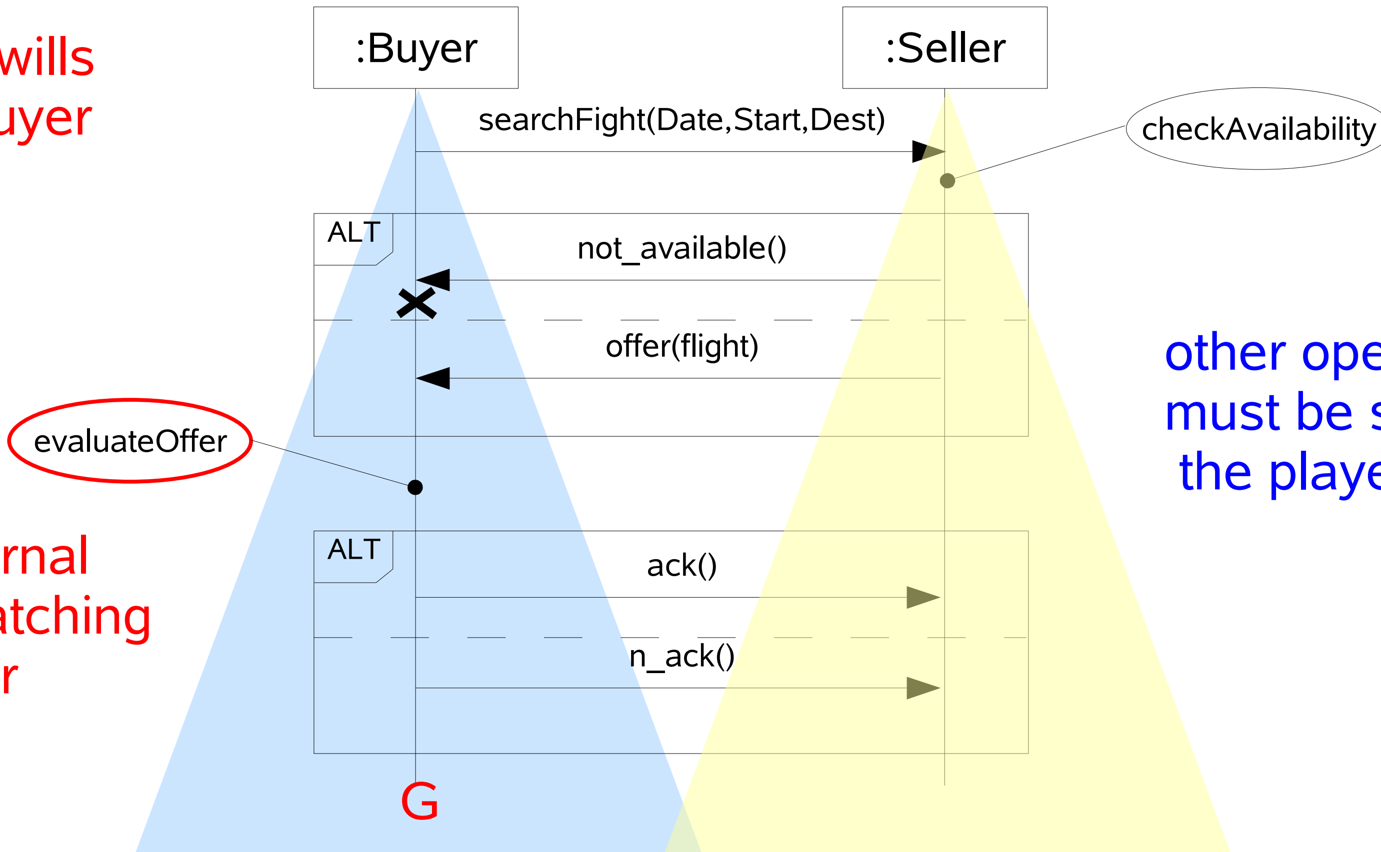
service1 wills to play Buyer

it has an internal operation matching evaluateOffer



The flight reservation example

service1 wills
to play Buyer

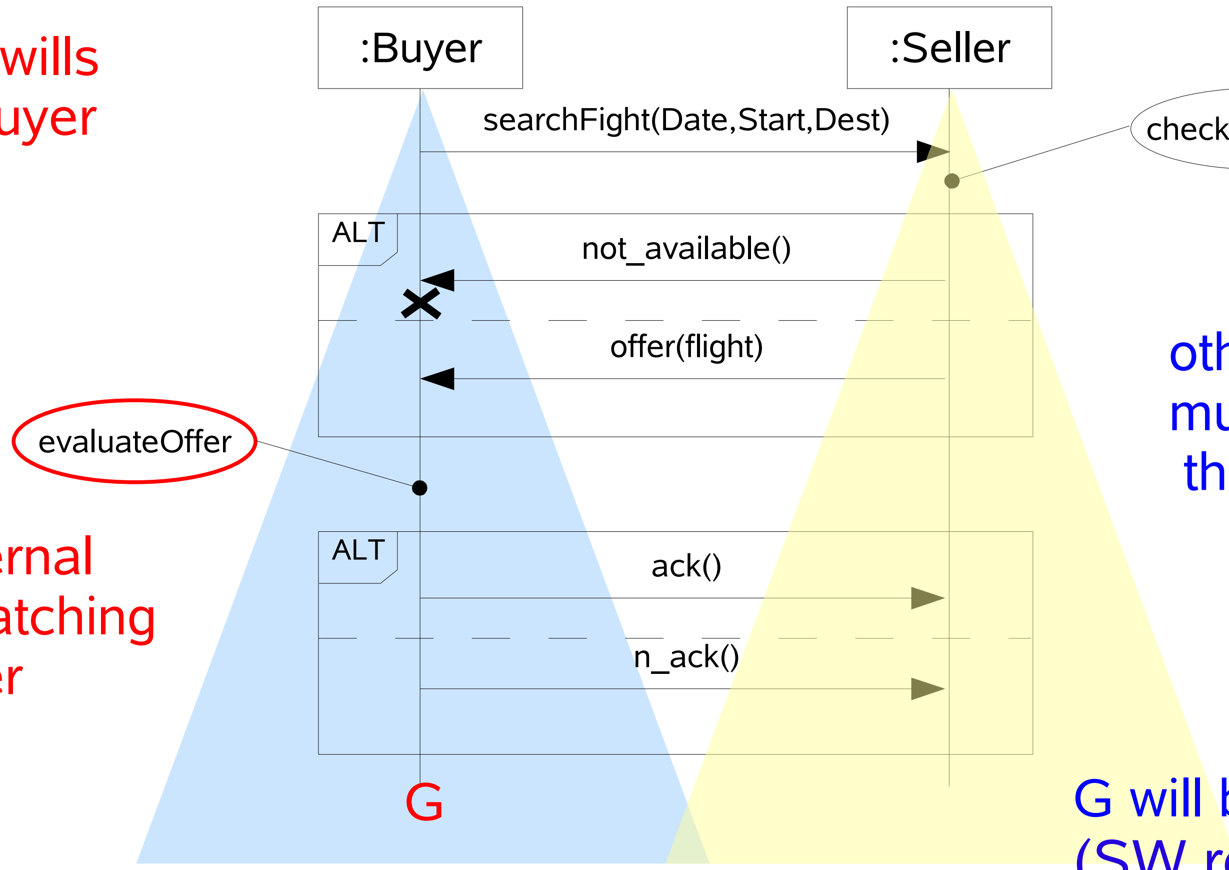


other operations
must be supplied by
the player of seller

it has an internal
operation matching
evaluateOffer

The flight reservation example

service1 wills to play Buyer

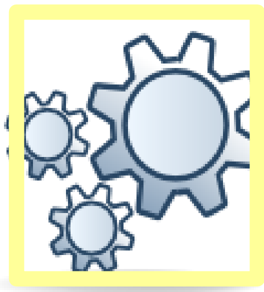


other operations must be supplied by the player of seller

it has an internal operation matching evaluateOffer

G will be preserved (SW re-use!!) by using these operations?

Slightly simpler formalization



SERVICE DESCRIPTION

$$S_d = \langle O, G, P \rangle$$

Operations

op≫ (interlocutor, content)
solicit-response

op≪ (interlocutor, content)
request-response

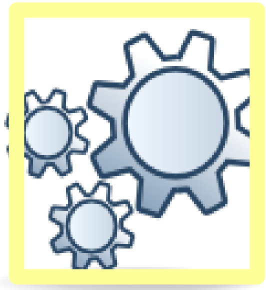
op (content)
internal operation

Get-message Actions

one-way operations

Procedures

Formalization: service description



$$S_d = \langle O, G, P \rangle$$

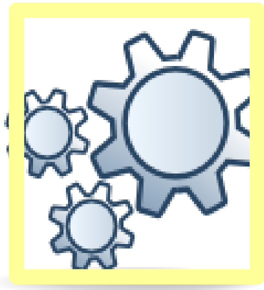
SERVICE DESCRIPTION

Every operation *op* is an atomic action, defined by its preconditions and effects:

op^d(interlocutor, content) **causes** $\{E_1, \ldots, E_n\}$

op^d(interlocutor, content) **possible if** $\{P_1, \ldots, P_t\}$

Formalization: service description



$$S_d = \langle O, G, P \rangle$$

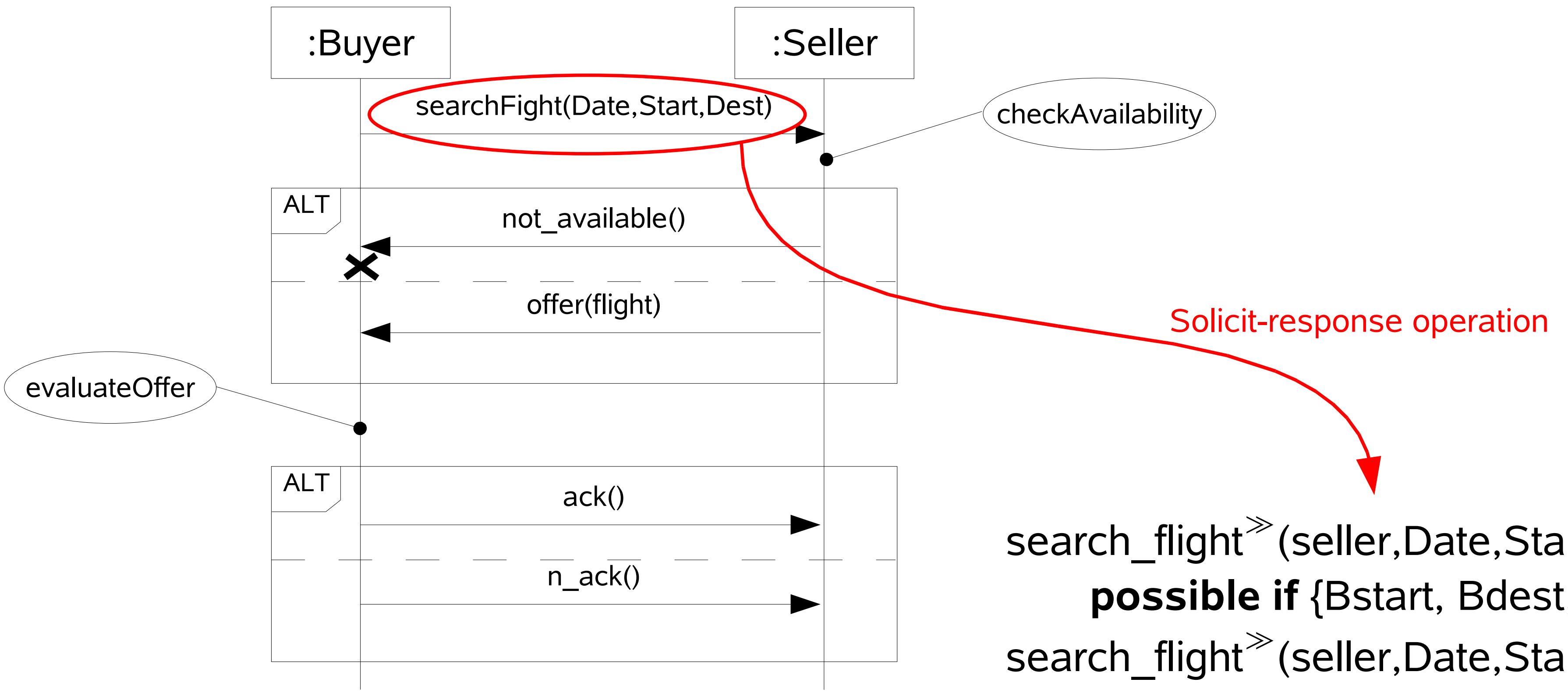
SERVICE DESCRIPTION

Internal operations are not visible from outside: they are represented also as actions:

op(content) **causes** $\{E_1, \ldots, E_n\}$

op(content) **possible if** $\{P_1, \ldots, P_t\}$

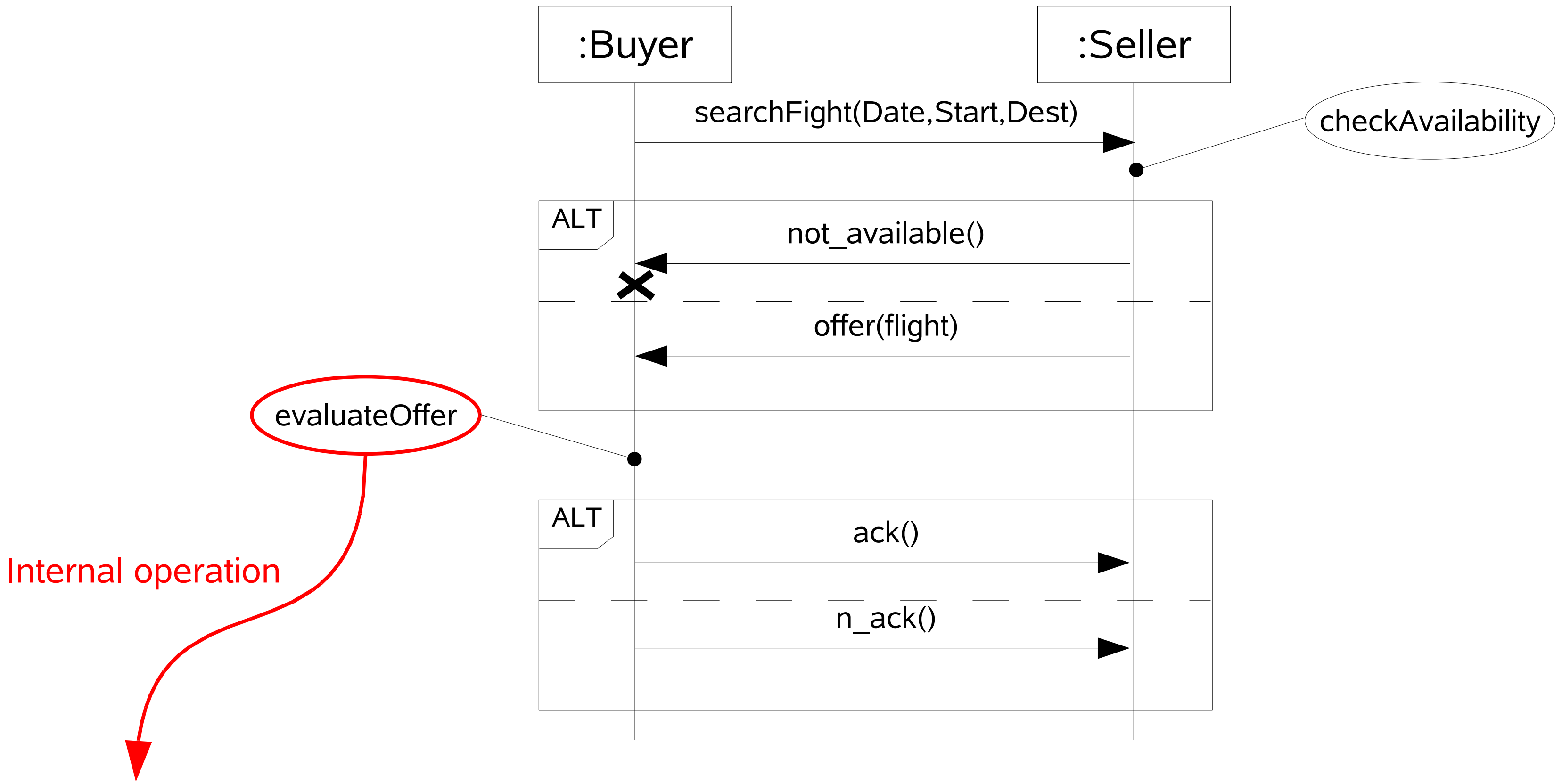
The flight reservation example



search_flight$^{\gg}$(seller,Date,Start,Dest)
possible if {Bstart, Bdest, Bdate}

search_flight$^{\gg}$(seller,Date,Start,Dest)
causes {Bwill_get_offer}

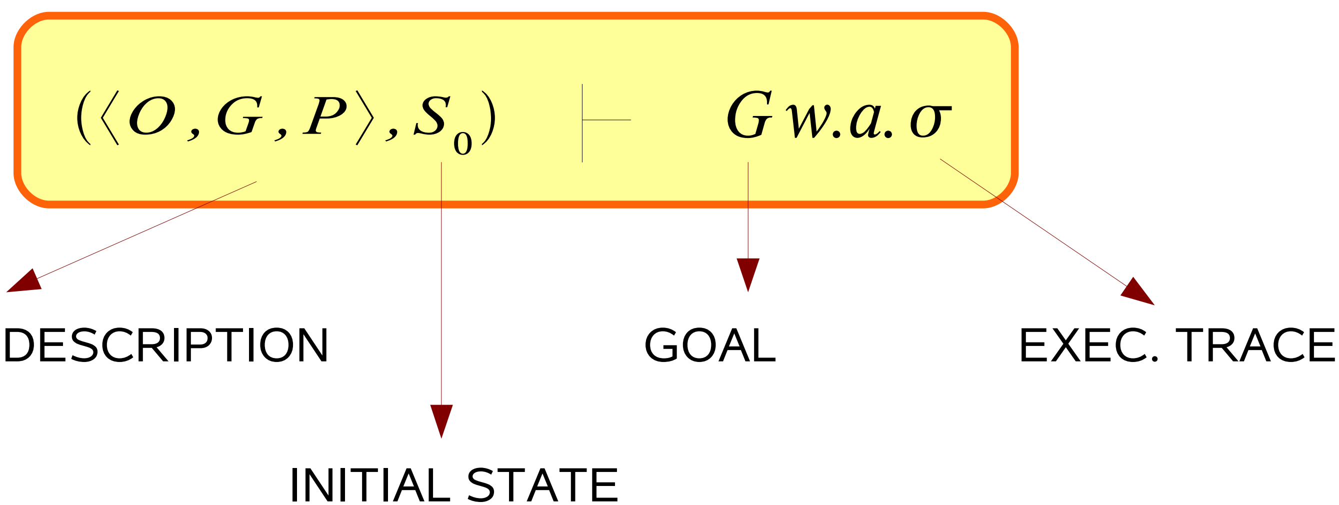
The flight reservation example



eval_offer **possible if** {Boffer(flight)}
eval_offer **causes** {Beval_rst(Y)}

Reaching a goal

- Given a service description and an initial state, it is possible to verify if it is possible to reach a state where a goal G holds
- If the answer is positive, an execution trace leading to such a state is returned:



Bound and unbound operations

- The policy of a service contains operations that belong to other roles in the choreography
- The set *O* can be also partitioned in two sets:
  - **bound** operations (internal)
  - **unbound** operations
- **Unbound** operations: $O_u \subseteq O$
  - specified in the choreography
  - substituted by those supplied by the counterparts during the service selection process

Substitutions and goals

- Given a set O_{S_i} of operations provided by a candidate interlocutor S, we can define the substitution $\theta = [O_{S_i} / O_u]$ that will produce the service description $S_d \theta$:

$$S_d = \langle O, G, P \rangle \xrightarrow{\theta} S_d \theta = \langle O\theta, G\theta, P\theta \rangle$$

Given that:

$$(\langle O, G, P \rangle, S_0) \vdash G \, w.a. \, \sigma$$

Can the goal be achieved also after the substitution?

$$(\langle O\theta, G\theta, P\theta \rangle, S_0) \vdash G \, w.a. \, \sigma$$

Zaremski and Wing Match rules

EM	• Exact pre/post match:	$Pr(o_u) = Pr(s) \wedge Ef(o_u) = Ef(s)$
PIM	• Plugin match:	$Pr(o_u) \supseteq Pr(s) \wedge Ef(s) \supseteq Ef(o_u)$
POM	• Plugin Post match:	$Ef(s) \supseteq Ef(o_u)$
GPIM	• Guarded Plugin match:	$Pr(o_u) \supseteq Pr(s) \wedge ((Pr(s) \cup Ef(s)) \supseteq Ef(o_u))$
GPOM	• Guarded Post match:	$((Pr(s) \cup Ef(s)) \supseteq Pr(o_u))$

o_u => unbounded operations
s => operations provided by a service

Def - Conservative substitution

- Let $\langle O, G, P \rangle$ be a service description S_i, S_0 the initial state, and G the goal of interest. When the following relation holds:

$$\exists \sigma, \theta = [O_{S_j} / O^{\sigma}_{u(R_j)}], O^{\sigma}_{u(R_j)} \subseteq O_{u(R_j)} \; s.t.$$

$$(\langle O, G, P \rangle, S_0) \vdash G \text{ w.a. } \sigma \Rightarrow$$

$$(\langle O\theta, G\theta, P\theta \rangle, S_0) \vdash G \text{ w.a. } \sigma\theta$$

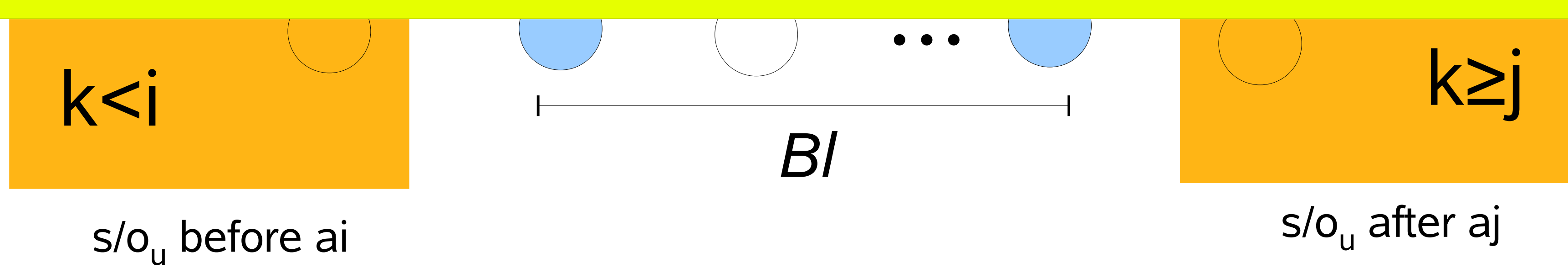
where $O^{\sigma}_{u(R_j)}$ is the set of all unbound operations in σ that refer to another role $R_i, i \neq j$, the substitution is *conservative*.

Uninfluential additional effect

- Consider a [s/o_u] in a substitution θ_{PIM} and an effect $B\neg l$ of s which is *not* an effect of o_u: $B\neg l \in Effs(s) - Effs(o_u)$

$B\neg l$ is an *uninfluential fluent*, *w.r.t.* $\sigma\theta_{PIM}$, iff:

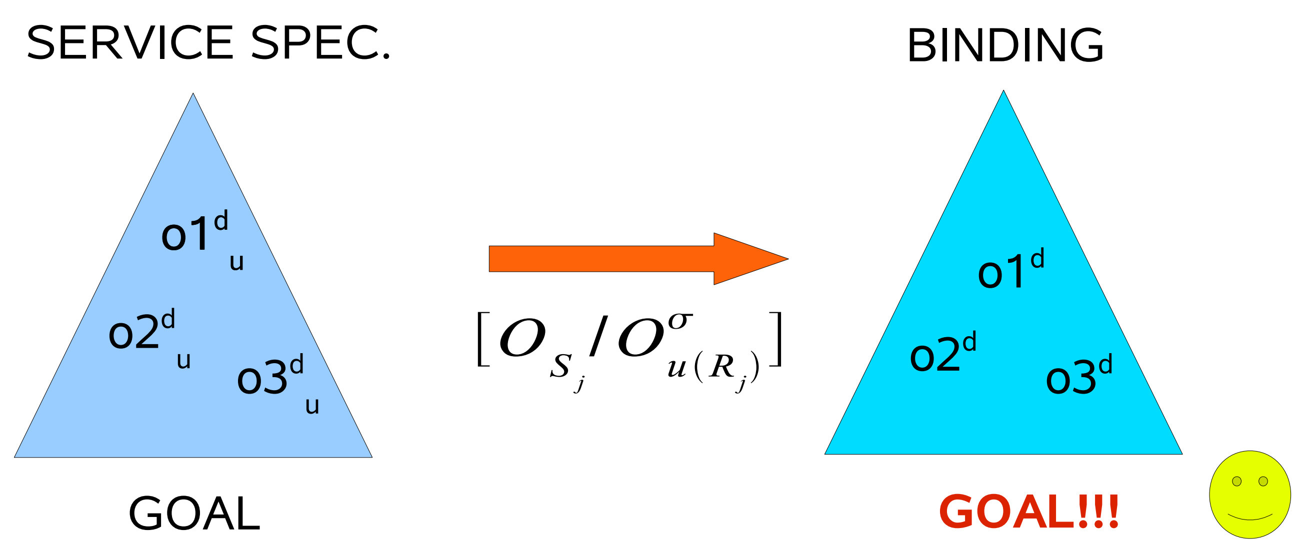
A **PIM-substitution** is **uninfluential** iff all the additional effects of the service operations, that are substituted to unbounded operations, are *uninfluential fluents*



Theorem (proof by absurd)

- Let G be a goal and $\langle O, G, P \rangle$ a service description. If $(\langle O, G, P \rangle, S_0) \vdash G$ w.a. σ and there is an uninfluential substitution $\theta_{PIM} = [O_{S_j} / O^{\sigma}_{u(R_j)}], O^{\sigma}_{u(R_j)} \subseteq O_{u(R_j)}$ then

$$(\langle O\theta_{PIM}, G\theta_{PIM}, P\theta_{PIM} \rangle, S_0) \vdash G \text{ w.a. } \sigma\theta_{PIM}$$



Related publications

- M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. *“Service selection by choreography-driven matching”*.
  In T. Gschwind and C. Pautasso, editors, Proc. of the 2nd ECOWS Workshop on Emerging Web Services Technology, WEWST 2007, Halle (Saale), Germany, Nov. 2007

Conclusions

- This work
  - has studied the relation between local matchmaking and the global goal achievement in the context of a choreography
  - has studied the selection of an appropriate interlocutor driven by the global goal and the choreography
- To this aim it
  - introduced the concepts of capability and capability requirement
  - introduced the notion of conservative match, and of uninfluential substitution
  - proved that Zaremski and Wing's matches are not conservative
  - proposed a conservative variant of the plugin match

Thanks

The room reservation example

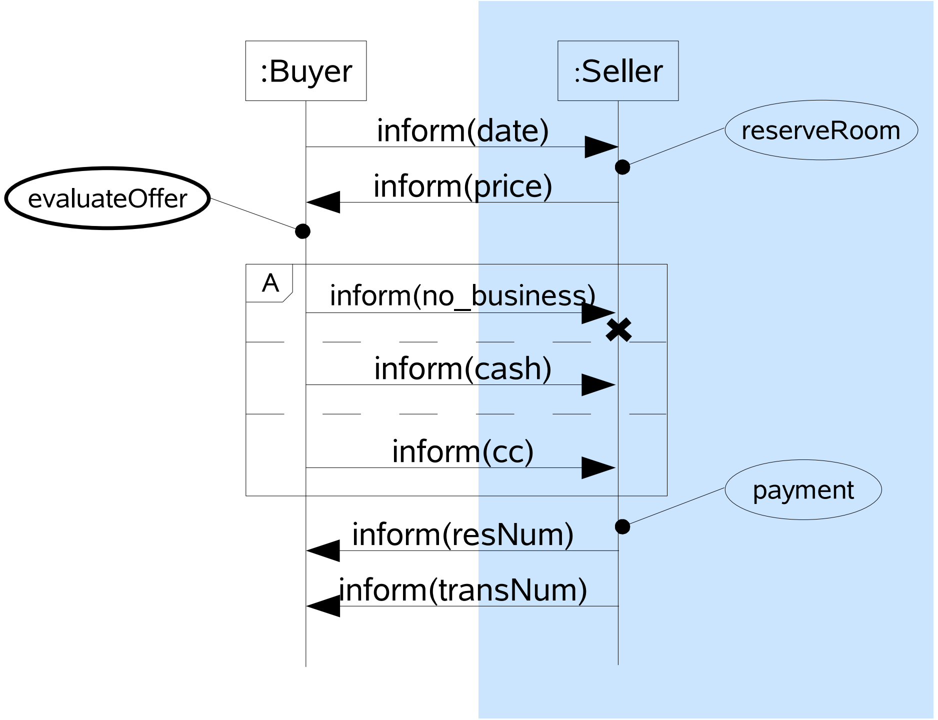
$$R_{seller} = \langle S_A, G_A, CR, P \rangle$$

S_A={inform(s,b,price), inform(s,b,resNum), inform(s,b,transNum)}

G_A={receive_date(s,b,date), receive_evaluation(s,b,[no_business,cash,cc])}

CR={reserve_room$_{CR}$,payment$_{CR}$}

P={booking,finalize_reservation}



The room reservation example

The procedures in P are described by:

booking **is** receive_date(s,b,date), reserve_room$_{CR}$,
receive_evaluation(s,b,[no_business,cash,cc]),
finalize_reservation

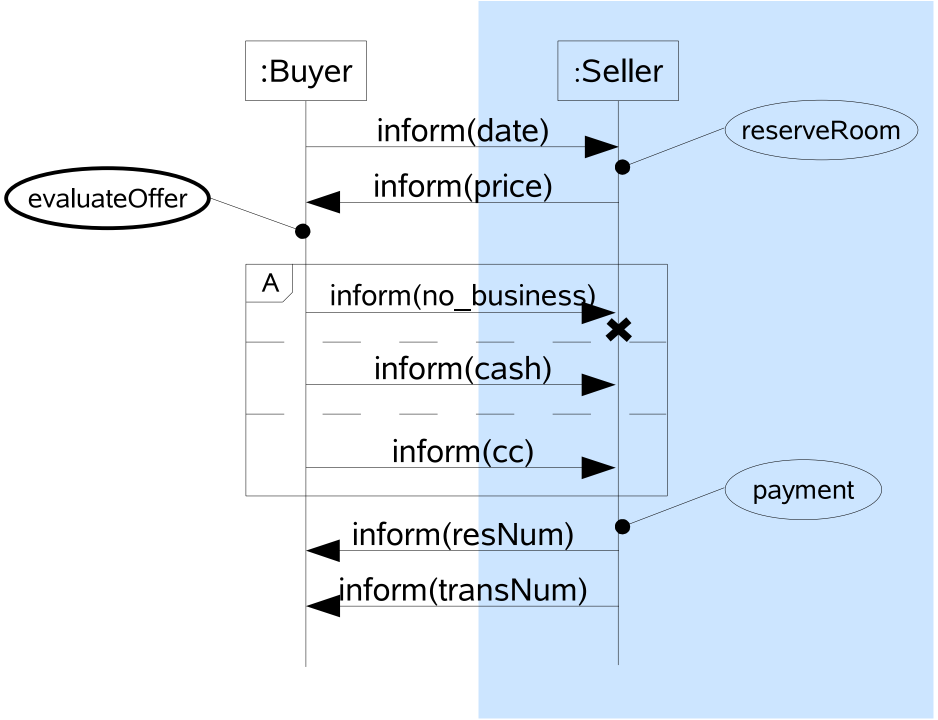
finalize_reservation **is** Bno_business?

finalize_reservation **is** payment$_{CR}$,
inform(s,b,resNum), inform(s,b,transNum)

The capability requirements in CR:

reserve_room$_{CR}$ **causes** {Bprice}
reserve_room$_{CR}$ **possible if** {Bdate}

payment$_{CR}$ **causes** {BtransNum,BresNum}
payment$_{CR}$ **possible if** {BpcashSupported,
BPccSupported



The room reservation example

$$(\langle S_A, G_A, CR, P \rangle, S_0) \vdash G$$

The initial state S_0 are described by:

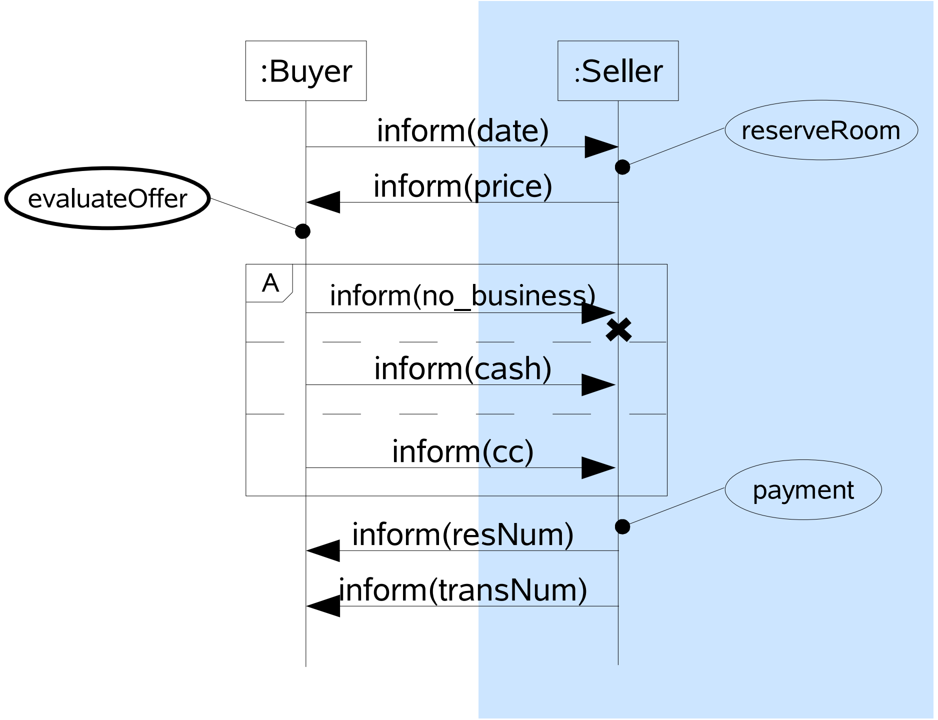
S_0={BPcashSupported, BPccSupported}

The goal G is:

G={Btransnum, BresNum} **after** booking

After the reasoning process we have as result the execution trace:

σ=inform(b,s,date); reserve_room$_{CR}$;
inform(s,b,price); inform(b,s,cc); payment$_{CR}$;
inform(s,b,resNum); inform(s,b,transNum)



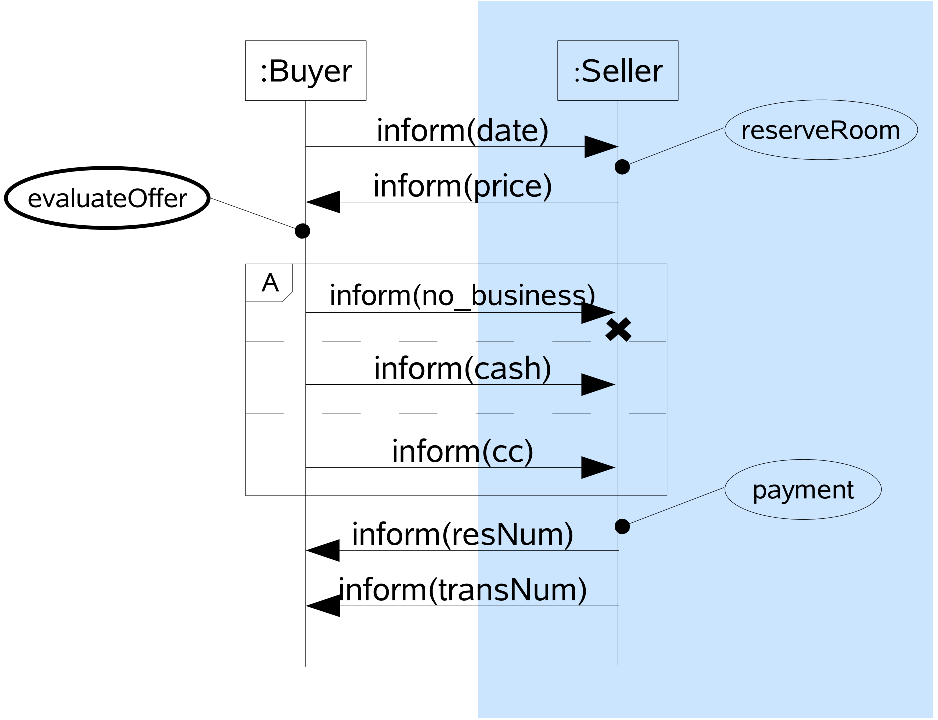
The room reservation example

The corresponding internal capabilities of the entity are:

$reserve\_room_{C1}$ **causes** {B¬PccSupported,Bprice}
$reserve\_room_{C1}$ **possible if** {Bdate}

$reserve\_room_{C2}$ **causes** {BfreeDinner,Bprice}
$reserve\_room_{C2}$ **possible if** {Bdate}

$payment_{C}$ **causes** {BtransNum, BresNum}
$payment_{C}$ **possible if** {BpcashSupported, BPccSupported}



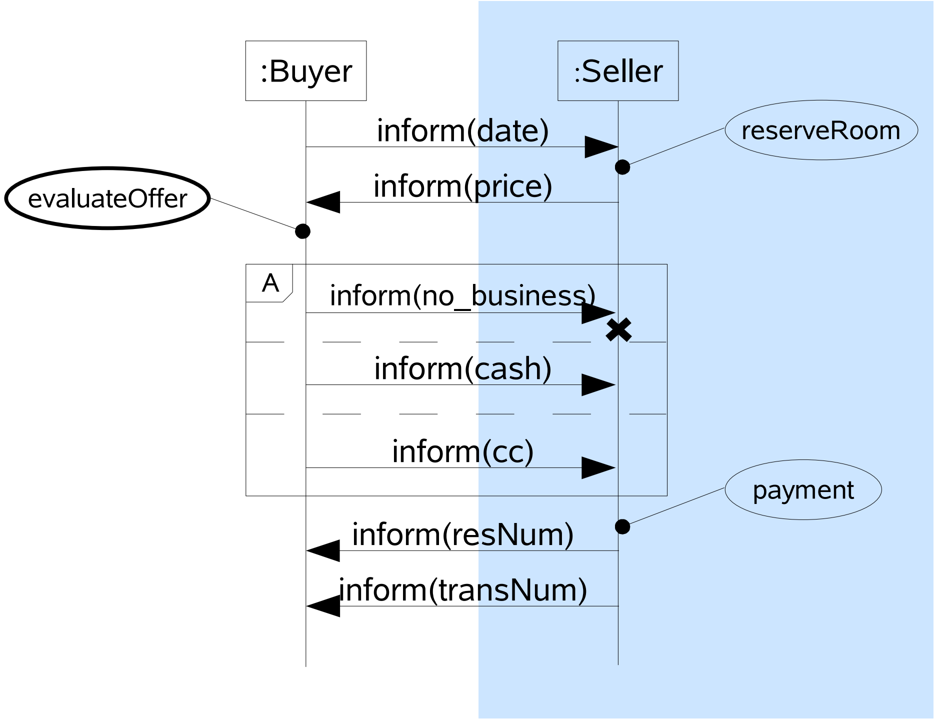
If we consider ONLY a local match ruled by the plugin level, we can find two possible substitutions:

Θ_1={[$reserve\_room_{C1}$,$reserve\_room_{CR}$],[$payment_{C}$,$payment_{CR}$]}
Θ_2={[$reserve\_room_{C2}$,$reserve\_room_{CR}$],[$payment_{C}$,$payment_{CR}$]}

The room reservation example

Θ_1 is not conservative!

The additional effect B¬PccSupported of reserve_room$_{C1}$ is not an uninfluential fluent.

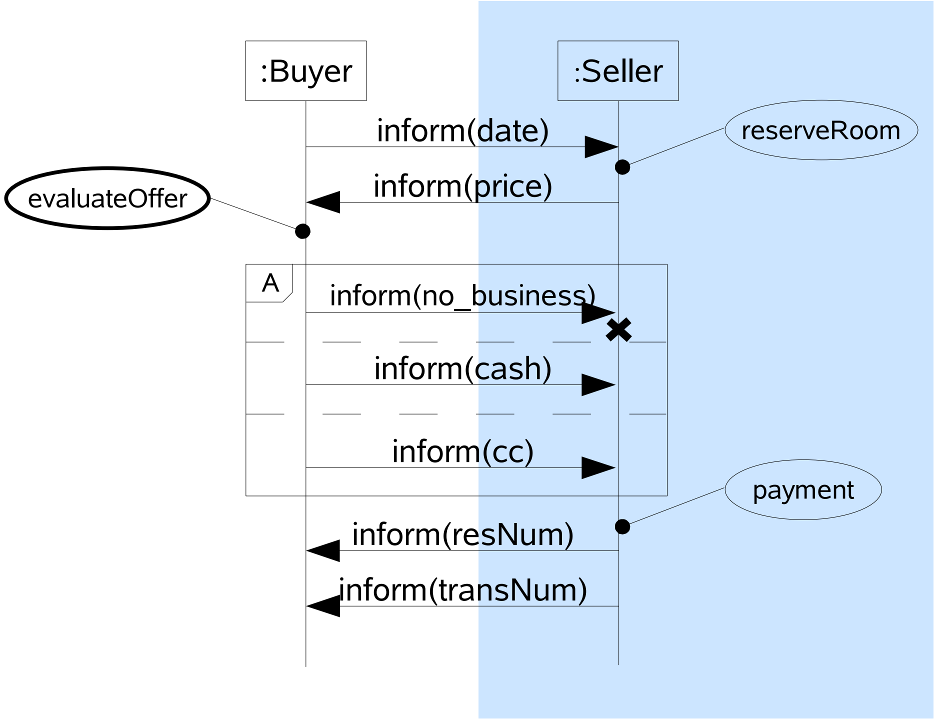


Θ_1={[reserve_room$_{C1}$,reserve_room$_{CR}$], [payment$_C$,payment$_{CR}$]}

Θ_2={[reserve_room$_{C2}$,reserve_room$_{CR}$], [payment$_C$,payment$_{CR}$]}

The room reservation example

Θ_1 is not conservative!
The additional effect B¬PccSupported of $reserve\_room_{C1}$ is not an uninfluential fluent.



Θ_1={[$reserve\_room_{C1}$,$reserve\_room_{CR}$], [$payment_C$,$payment_{CR}$]}

Θ_2={[$reserve\_room_{C2}$,$reserve\_room_{CR}$], [$payment_C$,$payment_{CR}$]}

Θ_2 is conservative!
The additional effect BfreeDinner of $reserve\_room_{C2}$ is an uninfluential fluent.

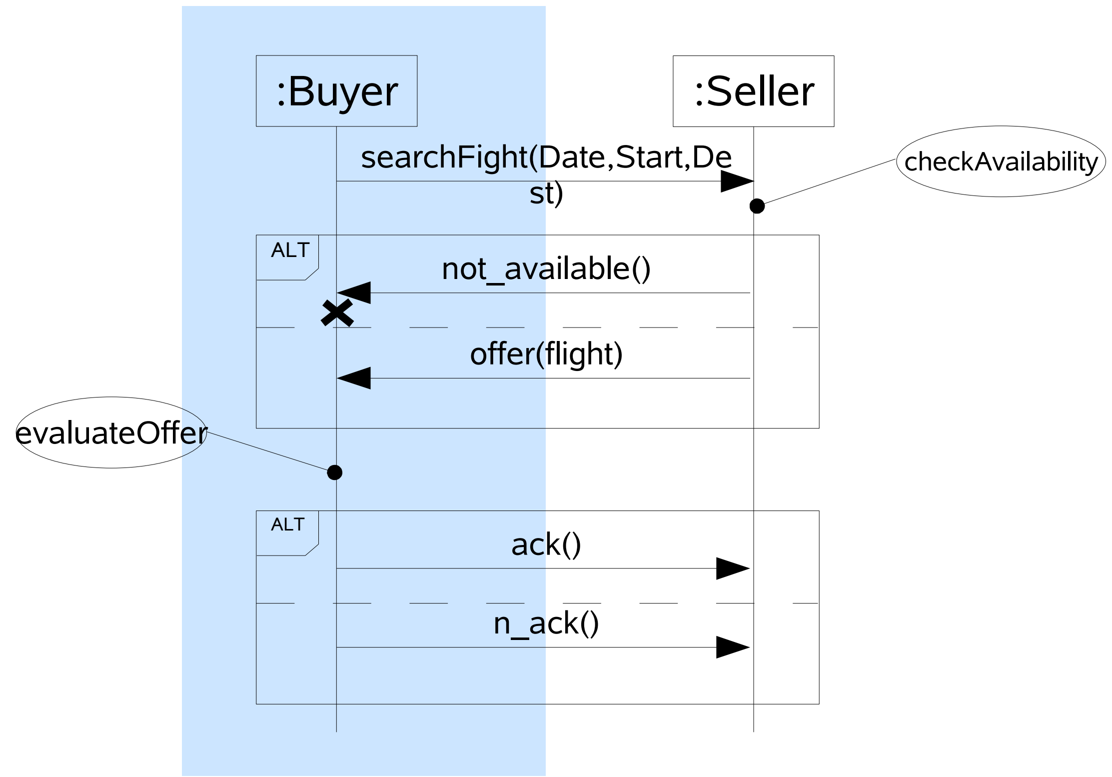
The flight reservation example

$S_{b1} = \langle O, G, P \rangle$

P={booking,finalize}

O={search_flight$^{\gg}_{u}$, not_available$^{\ll}_{u}$,
eval_offer, offer$^{\ll}_{u}$, ack$^{\gg}_{u}$, n_ack$^{\gg}_{u}$}

G={get_answer}



The flight reservation example

The procedures in *P are described by:*

booking(Seller,Date,Start,Dest) **is**

search_flight$^{\gg}_{u}$(Seller,Date,Start,Dest),
get_answer(Seller), Boffer(not_avail)?

booking(Seller,Date,Start,Dest) **is**

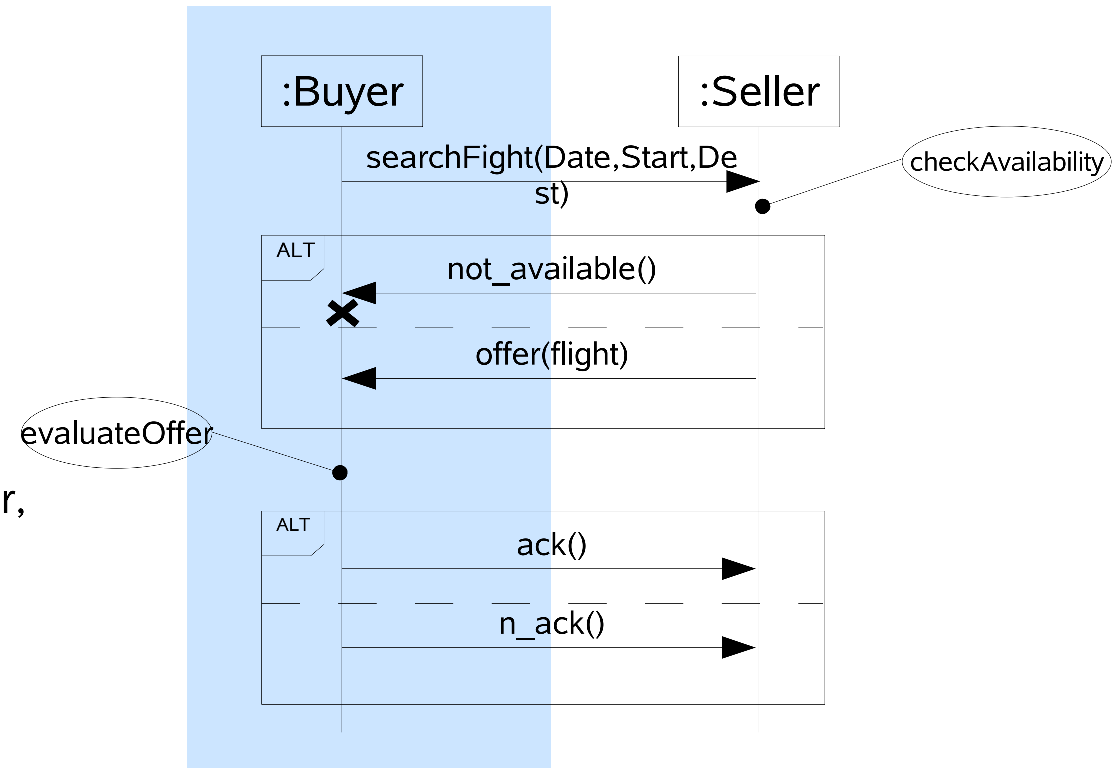
search_flight$^{\gg}_{u}$(Seller,Date,Start,Dest),
get_answer(Seller), Boffer(flight)?, eval_offer,
finalize(Seller)

finalize(Seller) **is** Beval_rst(business)?,

ack$^{\gg}_{u}$(Seller)

finalize(Seller) **is** Beval_rst(no_business)?,

n_ack$^{\gg}_{u}$(Seller)



The flight reservation example

The operations in O *are described by:*

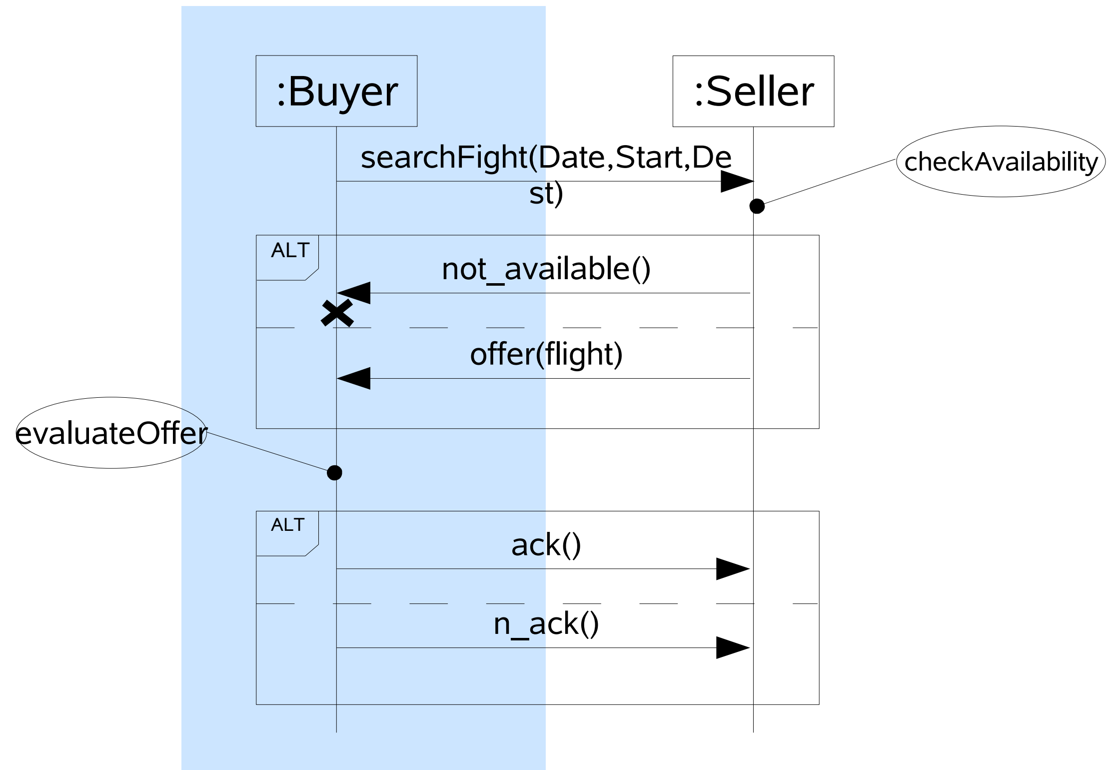
eval_offer ***causes*** *{Beval_rst(Y)}*
eval_offer **possible if** {Boffer(flight)}

search_flight$^{\gg}_{u}$ **causes** {Bwill_get_offer}
search_flight$^{\gg}_{u}$ **possible if** {Bstart, Bdest, Bdate}

not_available$^{\ll}_{u}$ **causes** {Boffer(not_available)}
not_available$^{\ll}_{u}$ **possible if** {}

offer$^{\ll}_{u}$ **causes** {Boffer(flight)}
offer$^{\ll}_{u}$ **possible if** {}

ack$^{\gg}_{u}$ **causes** {Bbooked(flight)}
ack$^{\gg}_{u}$ **possible if** {}



n_ack$^{\gg}_{u}$ causes {B¬booked(flight)}
n_ack$^{\gg}_{u}$ possible if {}

The flight reservation example

The initial state S_0 *are described by:*

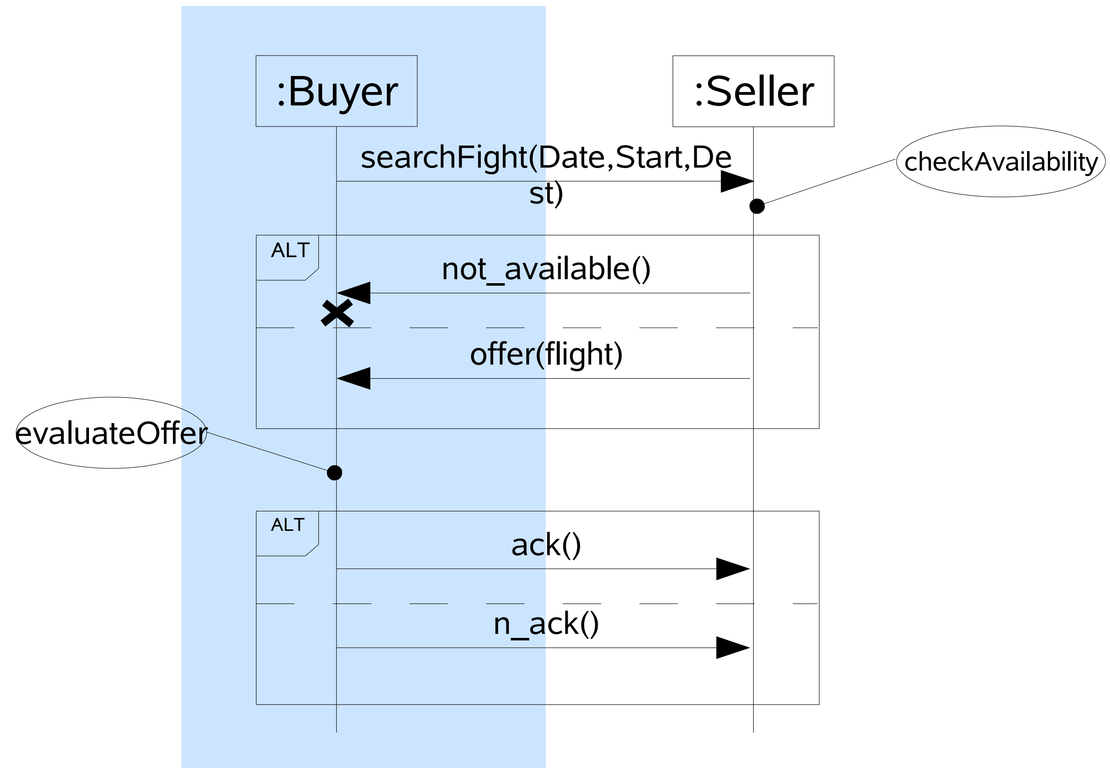
S_0={Bdate, Bstart, Bdest, Bsmoking_flight}

The goal of *b1* is that the following condition holds:

G={Bbooked(flight), Bsmoking_flight} **after**
booking(seller, date, start, dest)

After the reasoning process we have as result the execution trace:

σ=search_flight$^{\gg}_{u}$; offer$^{\ll}_{u}$; eval_offer; ack$^{\gg}_{u}$



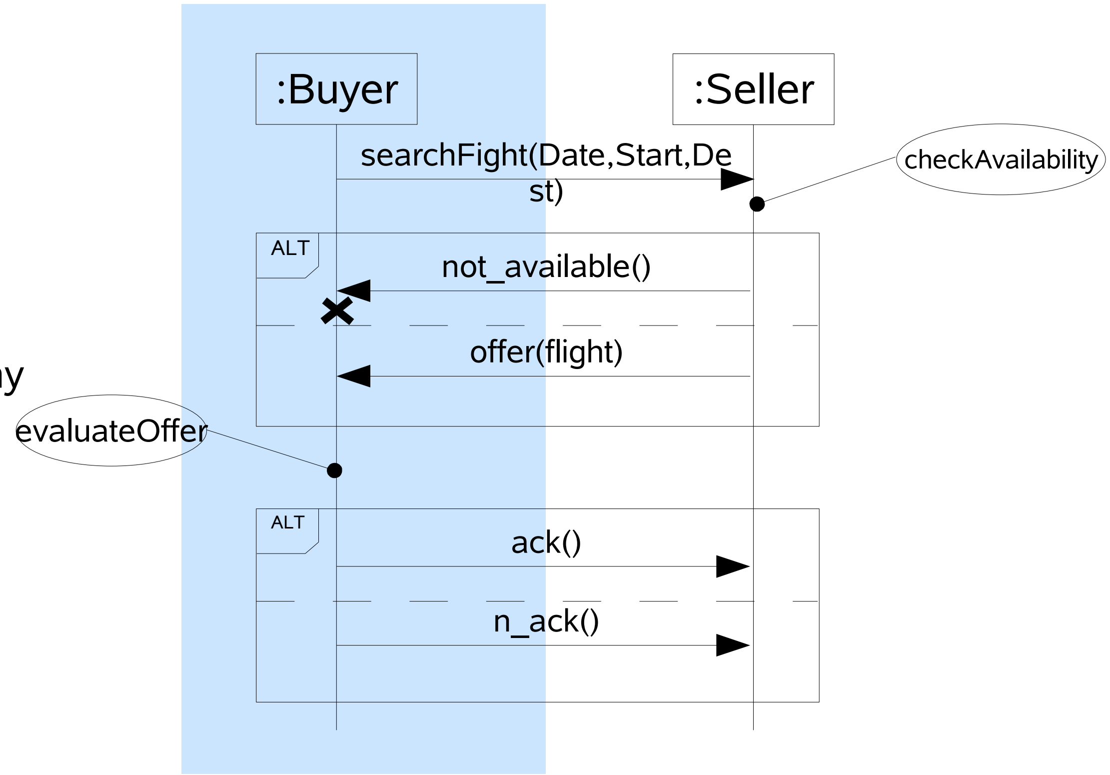
The buyer wants to have after the interaction a reservation on a smoking flight

The flight reservation example

Let us consider the candidate seller s1 and the plugin match as matching rule:

search_flight$^{\gg}$ **causes** {Bwill_get_offer,
B¬smoking_flight}
search_flight$^{\gg}$ **possible if** {Bstart, Bdest, Bdate}

(other operations are exactly as in the choreography role)
We will find the substitution [search_flight$^{\gg}$ / search_flight$^{\gg}_{u}$] but in this case the query:



FAILS!

$$(\langle O\theta^{s1}_{PIM}, G\theta^{s1}_{PIM}, P\theta^{s1}_{PIM}\rangle, S_0) \vdash G$$

b1 is looking for another service, which can play the role of seller

The flight reservation example

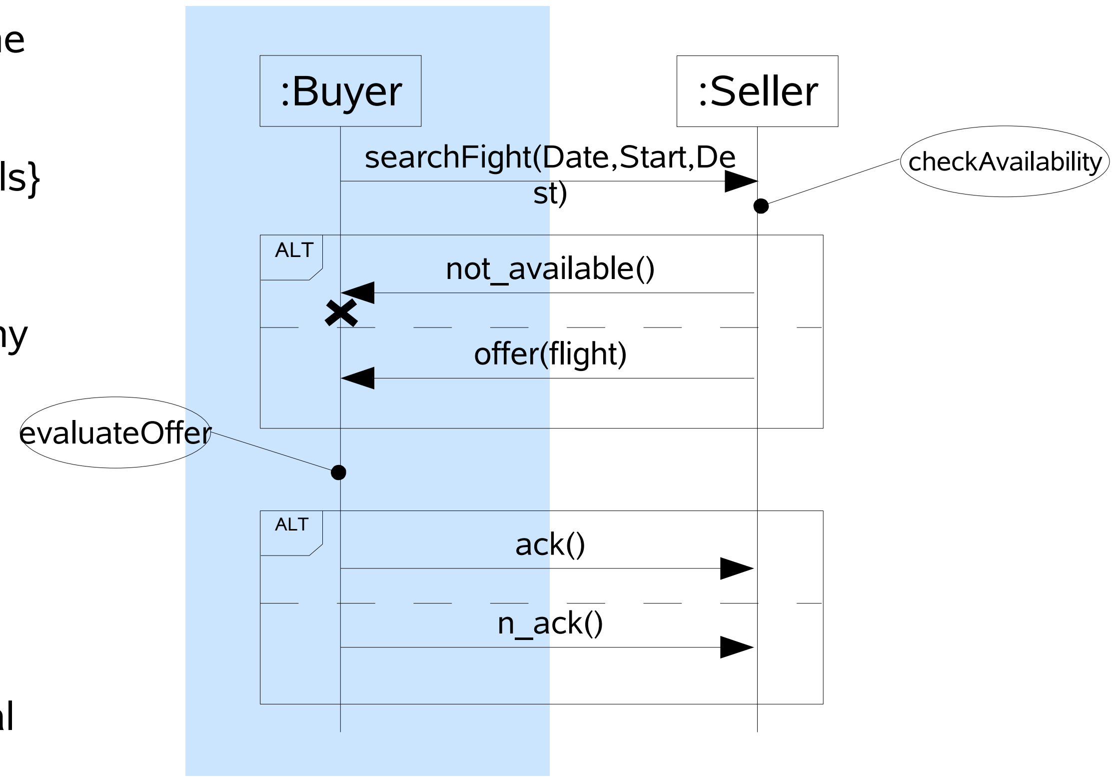
Let us consider **another** candidate seller s2 and the plugin match as matching rule:

search_flight$^{\gg}$ **causes** {Bwill_get_offer, Bveg_meals}

search_flight$^{\gg}$ **possible if** {Bstart, Bdest, Bdate}

(other operations are exactly as in the choreography role)

We will find the substitution [search_flight$^{\gg}$ / search_flight$^{\gg}_{u}$] in this case the query:



SUCCEEDS!

The additional effect Bveg_meals is an uninfluential fluents $(\langle O\theta^{s2}_{PIM}, G\theta^{s2}_{PIM}, P\theta^{s2}_{PIM}\rangle, S_0) \vdash G$

b1 is looking for another service, which can play the role of seller