Conservative re-use ensuring matches for service selection

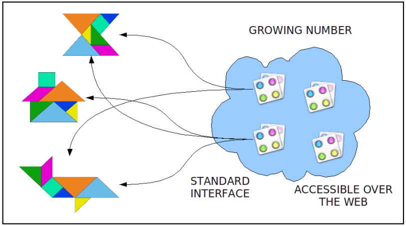
M. Baldoni, C. Baroglio, V. Patti, and C. Schifanella

Dipartimento di Informatica — Università degli Studi di Torino
C.so Svizzera, 185 — I-10149 Torino (Italy)
{baldoni,baroglio,patti,schi}@di.unito.it

November 17, 2008

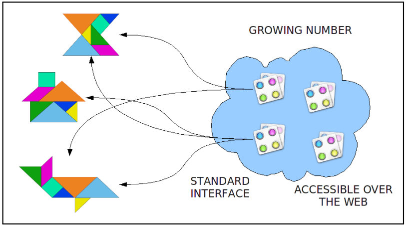
(Web) Services re-use

- Automatic retrival, selection, composition of (Web) Services
- The need for a *semantic layer*, e.g. *OWL-S* [7] and *WSMO* [4]
- A *richer annotation*, aimed at representing the so called *IOPEs* (inputs, outputs, preconditions and effects of the service)
- *Design by contract* [6]: preconditions and postconditions



(Web) Services re-use: semantic matchmaking

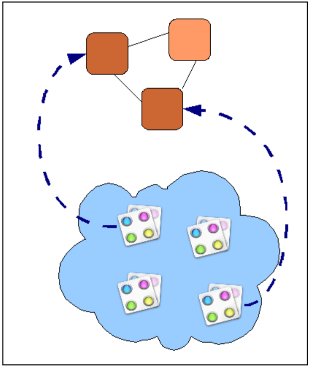
- Semantic annotation allows the discovery of services, whose descriptions *do not exactly match* with the corresponding queries
- E.g., PIM (*Plugin Match*): $Q_{pre} \Rightarrow S_{pre} \wedge S_{post} \Rightarrow Q_{post}$



(Web) Services re-use: semantic matchmaking

- Semantic matchmaking focuses on the discovery of single services, a *single operation*

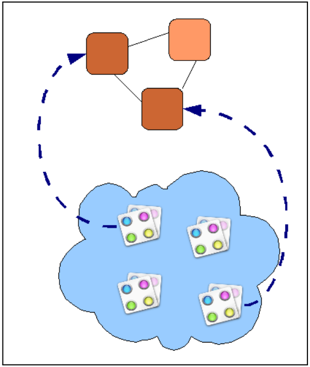
- In general, however, the use of a web service implies the execution of a *complex sequence of operations* in a particular *order*, which might even involve other services
- WS-CDL [10] is aimed at specifying (complex) patterns of interaction



(Web) Services re-use: semantic matchmaking

- Semantic matchmaking focuses on the discovery of single services, a *single operation*

- In general, however, the use of a web service implies the execution of a *complex sequence of operations* in a particular *order*, which might even involve other services
- WS-CDL [10] is aimed at specifying (complex) patterns of interaction

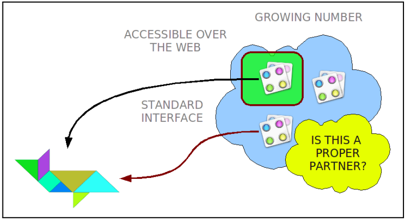


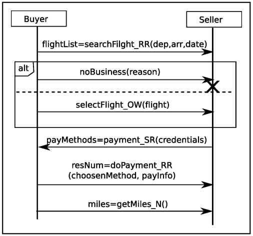
(Web) Services re-use: own goals

Selecting existing services that have to play the roles of a given choreography

This task implies verifying two things:

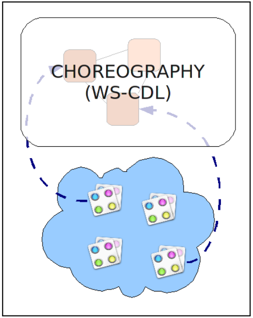
1. the *conformance* of the service to the specification of a role of interest
2. the use of that service allows the achievement of the *goal*, that caused its search

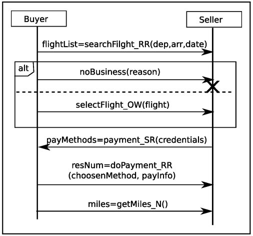




(Web) Services re-use: unbound roles

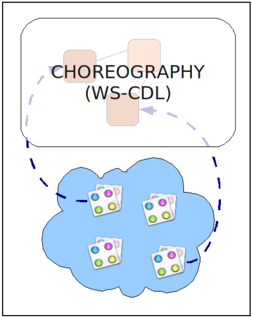
- The achievement of the goal depends on the operation sequence because each operation can influence the executability and the outcomes of the subsequent ones
- Many of the operations, however, are offered by the partners in the interaction which, at the time when the service reasons about the choreography, they are still unknown
- Unbound operations (selection process will link them to actual players)

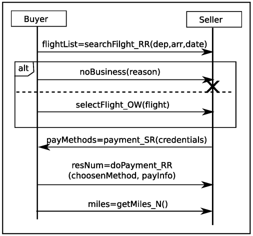




(Web) Services re-use: reasoning on choreographies

- The reasoning process must, therefore, use the specifications of the operations that will be supplied by the partners, specifications which are included in the choreography
- A selection process will link unbound operations with operations offered by existing services, and it does so by applying some kind of (possibly flexible) match

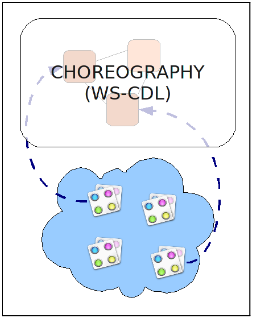


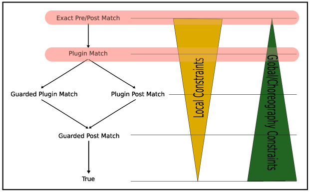


(Web) Services re-use: matchmaking and goals

- In [2], however, we showed that performing a match operation by operation, by applying the definitions in [11], *does not* preserve the global goal

- Therefore, the matchmaking process, that is applied to discover services, should not only focus on local properties of the single operations, e.g. IOPEs, but it should also consider constraints that derive from the global schema of execution, which is given by the choreography





In this paper

- We exploit an *action-based representation* of the specifications of the operations of a service: each operation is described in terms of its preconditions and effects, input and output
- We show how to enrich the class of *re-use ensuring matches* [3] proposed in the literature so to produce substitutions that preserve goals

Definition (Re-use ensuring match [3])

A specification match M is re-use ensuring iff for any S and Q, $M(I, Q) \wedge \{S_{pre}\}S\{S_{post}\} \Rightarrow \{Q_{pre}\}S\{Q_{post}\}$.

In this paper

- We exploit an *action-based representation* of the specifications of the operations of a service: each operation is described in terms of its preconditions and effects, input and output
- We show how to enrich the class of *re-use ensuring matches* [3] proposed in the literature so to produce substitutions that preserve goals

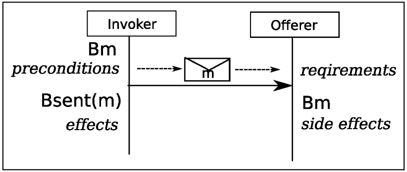
Definition (Re-use ensuring match [3])

A specification match M is re-use ensuring iff for any S and Q, $M(I, Q) \wedge \{S_{pre}\}S\{S_{post}\} \Rightarrow \{Q_{pre}\}S\{Q_{post}\}$.

Reasoning about services

- The notation is based on a logical theory for reasoning about actions and change in a modal logic programming setting
- The problem of reasoning amounts either to build or to traverse a sequence of transitions between *states*
- A state is a set of *fluents*, i.e., properties whose truth value can change over time, due to the application of actions
- There are four basic kinds of operations [1] (or atomic processes, when using OWL-S terminology [7]):
  - *one-way*
  - *notify*
  - *request-response*
  - *solicit-response*

One-Way Operation



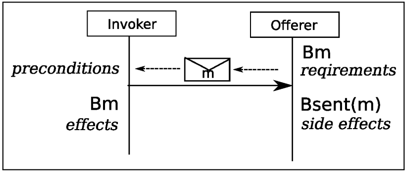
One-Way, invoker point of view:

(a) $operation^{\gg}_{ow}(m_{in})$ **possible if** $\mathbf{B}^{Invoker} m_{in} \wedge P_s$
(b) $operation^{\gg}_{ow}(m_{in})$ **causes** $\mathbf{B}^{Invoker} sent(m_{in})$
(c) $operation^{\gg}_{ow}(m_{in})$ **causes** E_s

One-Way, supplier point of view:

(a) $operation^{\ll}_{ow}(m_{in})$ **possible if** R_s
(b) $operation^{\ll}_{ow}(m_{in})$ **causes** $\mathbf{B}^{Offerer} m_{in}$
(c) $operation^{\ll}_{ow}(m_{in})$ **causes** S_s

Notify Operation



Notify, invoker point of view:

(a) $operation_n^{\gg}(m_{out})$ **possible if** P_s

(b) $operation_n^{\gg}(m_{out})$ **causes** $\mathbf{B}^{Invoker} m_{out}$

(c) $operation_n^{\gg}(m_{out})$ **causes** E_s

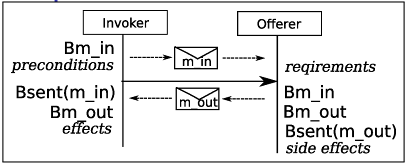
Notify, supplier point of view:

(a) $operation_n^{\ll}(m_{out})$ **possible if** $\mathbf{B}^{Offerer} m_{out} \wedge R_s$

(b) $operation_n^{\ll}(m_{out})$ **causes** $\mathbf{B}^{Offerer} sent(m_{out})$

(c) $operation_n^{\ll}(m_{out})$ **causes** S_s

Request-response Operation



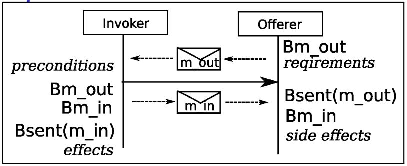
Request-response, invoker point of view:

(a) $\text{operation}^{\gg}_{rr}(m_{in}, m_{out})$ **possible if** $\mathbf{B}^{Invoker} m_{in} \wedge P_s$
(b) $\text{operation}^{\gg}_{rr}(m_{in}, m_{out})$ **causes** $\mathbf{B}^{Invoker} sent(m_{in})$
(c) $\text{operation}^{\gg}_{rr}(m_{in}, m_{out})$ **causes** $\mathbf{B}^{Invoker} m_{out}$
(d) $\text{operation}^{\gg}_{rr}(m_{in}, m_{out})$ **causes** E_s

Request-response, supplier point of view:

(a) $\text{operation}^{\ll}_{rr}(m_{in}, m_{out})$ **possible if** R_s
(b) $\text{operation}^{\ll}_{rr}(m_{in}, m_{out})$ **causes** $\mathbf{B}^{Offerer} m_{in}$
(c) $\text{operation}^{\ll}_{rr}(m_{in}, m_{out})$ **causes** $\mathbf{B}^{Offerer} m_{out}$
(d) $\text{operation}^{\ll}_{rr}(m_{in}, m_{out})$ **causes** $\mathbf{B}^{Offerer} sent(m_{out})$
(e) $\text{operation}^{\ll}_{rr}(m_{in}, m_{out})$ **causes** S_s

Solicit-response Operation



Solicit-response, invoker point of view:

(a) $\text{operation}^{\gg}_{sr}(m_{in}, m_{out})$ **possible if** P_s
(b) $\text{operation}^{\gg}_{sr}(m_{in}, m_{out})$ **causes** $\mathbf{B}^{Invoker} m_{out}$
(c) $\text{operation}^{\gg}_{sr}(m_{in}, m_{out})$ **causes** $\mathbf{B}^{Invoker} m_{in}$
(d) $\text{operation}^{\gg}_{sr}(m_{in}, m_{out})$ **causes** $\mathbf{B}^{Invoker} sent(m_{in})$
(e) $\text{operation}^{\gg}_{sr}(m_{in}, m_{out})$ **causes** E_s

Solicit-response, supplier point of view:

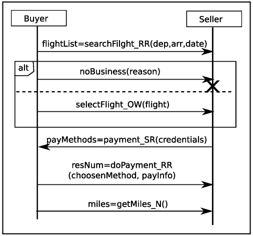
(a) $\text{operation}^{\ll}_{sr}(m_{in}, m_{out})$ **possible if** $\mathbf{B}^{Offerer} m_{out} \wedge R_s$
(b) $\text{operation}^{\ll}_{sr}(m_{in}, m_{out})$ **causes** $\mathbf{B}^{Offerer} sent(m_{out})$
(c) $\text{operation}^{\ll}_{sr}(m_{in}, m_{out})$ **causes** $\mathbf{B}^{Offerer} m_{in}$
(d) $\text{operation}^{\ll}_{sr}(m_{in}, m_{out})$ **causes** S_s

Choreographies and roles

- $\mathcal{P}$ encodes the complex behavior of the service; it is a collection of clauses of the kind:

$$p_0 \textbf{ is } p_1, \ldots, p_n$$

- p_i, $i = 1, \ldots, n$, is either an atomic action (operation), a test action (denoted by the symbol ?), or a procedure call
- Procedures can be recursive and are executed in a goal-directed way, similarly to standard logic programs, and their definitions can be non-deterministic as in Prolog.
- A *choreography* is made of a set of interacting *roles*, a role being a subjective view of the interaction that is encoded.

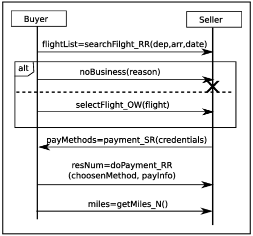


Reasoning on goals

- We need a mechanism that verifies if a goal condition holds after the interaction with the service has taken place
- *Fs* is the set of facts that we would like to hold “after” p
- This form of reasoning is known as *temporal projection*
- Queries of the form:

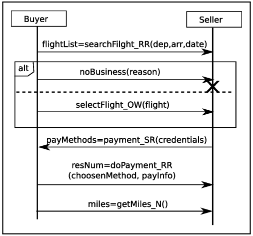
$$Fs \ \textbf{after} \ p$$

- The execution of the above query returns as a side-effect an *execution trace* σ of p



Choreographies and roles

- When a service plays a role in a choreography, its policy will contain some operations which are not of the service itself but belong to some other role of the choreography, with which it interacts
- Basic operations: a set of *bound operations* and a set of *unbound operations*, that must be supplied by some counterpart(s)
- Until the counterpart service(s) is (are) not defined, such operations will be those specified in the choreography
- Such operations will be offered by the interlocutors as ≫ operations



Conservative matching

Substitution

Let $S_d = \langle \mathcal{S}, \mathcal{P} \rangle$ be a service description, and let $\mathcal{S}_u$ be a subset of $\mathcal{S}$, containing unbound operations that are to be supplied by a same counterpart S_i. Let $\mathcal{S}_{S_i}$ be the set of operations in S_i that we want S_d to use, binding them to $\mathcal{S}_u$. We represent the binding by the substitution $\theta = [\mathcal{S}_{S_i} / \mathcal{S}_u]$ applied to S_d, i.e.: $S_d\theta = \langle \mathcal{S}\theta, \mathcal{P}\theta \rangle$, where every element of $\mathcal{S}_u$ is substituted by/bound to an element of $\mathcal{S}_{S_i}$.

Definition (Conservative substitution)

Let us consider a service $S_i = \langle \mathcal{S}, \mathcal{P} \rangle$ playing a role R_i in a given choreography, and a query G such that, given an initial state S_0, $(\langle \mathcal{S}, \mathcal{P} \rangle, S_0) \vdash G$ w.a. σ Consider a substitution $\theta = [\mathcal{S}_{S_j} / \mathcal{S}^{\sigma}_{u(R_j)}]$, where $\mathcal{S}^{\sigma}_{u(R_j)} = \{o_u \in \mathcal{S} \mid o \text{ occurs in } \sigma\}$ is the set of all unbound operations that refer to another role R_j, $j \neq i$, of the same choreography, that are used in the execution trace σ. θ is conservative when the following holds: $(\langle \mathcal{S}\theta, \mathcal{P}\theta \rangle, S_0) \vdash G$ w.a. $\sigma\theta$

Conservative matching

Substitution

Let $S_d = \langle \mathcal{S}, \mathcal{P} \rangle$ be a service description, and let $\mathcal{S}_u$ be a subset of $\mathcal{S}$, containing unbound operations that are to be supplied by a same counterpart S_i. Let $\mathcal{S}_{S_i}$ be the set of operations in S_i that we want S_d to use, binding them to $\mathcal{S}_u$. We represent the binding by the substitution $\theta = [\mathcal{S}_{S_i}/\mathcal{S}_u]$ applied to S_d, i.e.: $S_d\theta = \langle \mathcal{S}\theta, \mathcal{P}\theta \rangle$, where every element of $\mathcal{S}_u$ is substituted by/bound to an element of $\mathcal{S}_{S_i}$.

Definition (Conservative substitution)

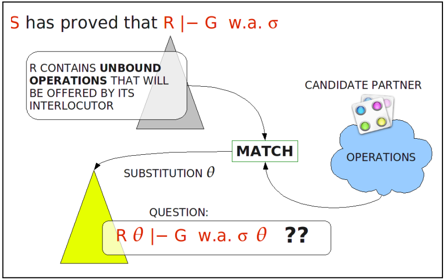
Let us consider a service $S_i = \langle \mathcal{S}, \mathcal{P} \rangle$ playing a role R_i in a given choreography, and a query G such that, given an initial state S_0, $(\langle \mathcal{S}, \mathcal{P} \rangle, S_0) \vdash G$ w.a. σ Consider a substitution $\theta = [\mathcal{S}_{S_j}/\mathcal{S}^{\sigma}_{u(R_j)}]$, where $\mathcal{S}^{\sigma}_{u(R_j)} = \{o_u \in \mathcal{S} \mid o \text{ occurs in } \sigma\}$ is the set of all unbound operations that refer to another role R_j, $j \neq i$, of the same choreography, that are used in the execution trace σ. θ is conservative when the following holds: $(\langle \mathcal{S}\theta, \mathcal{P}\theta \rangle, S_0) \vdash G$ w.a. $\sigma\theta$

Conservative matching

Definition (Conservative substitution)

Let us consider a service $S_i = \langle \mathcal{S}, \mathcal{P} \rangle$ playing a role R_i in a given choreography, and a query G such that, given an initial state S_0, $(\langle \mathcal{S}, \mathcal{P} \rangle, S_0) \vdash G$ w.a. σ Consider a substitution $\theta = [\mathcal{S}_{S_j} / \mathcal{S}^{\sigma}_{u(R_j)}]$, where $\mathcal{S}^{\sigma}_{u(R_j)} = \{o_u \in \mathcal{S} \mid o \text{ occurs in } \sigma\}$ is the set of all unbound operations that refer to another role R_j, $j \neq i$, of the same choreography, that are used in the execution trace σ. θ is conservative when the following holds: $(\langle \mathcal{S}\theta, \mathcal{P}\theta \rangle, S_0) \vdash G$ w.a. $\sigma\theta$

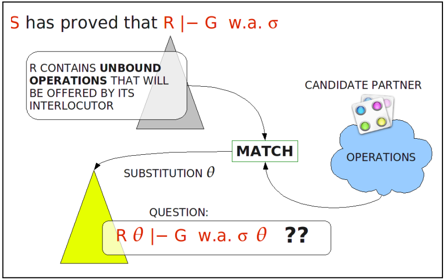
In [2] we have proved that, in general, flexible matches **are not conservative substitution**



Conservative matching

Definition (Conservative substitution)

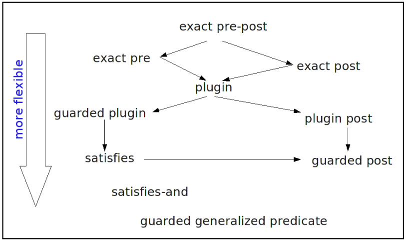
Let us consider a service $S_i = \langle \mathcal{S}, \mathcal{P} \rangle$ playing a role R_i in a given choreography, and a query G such that, given an initial state S_0, $(\langle \mathcal{S}, \mathcal{P} \rangle, S_0) \vdash G$ w.a. σ Consider a substitution $\theta = [\mathcal{S}_{S_j} / \mathcal{S}^{\sigma}_{u(R_j)}]$, where $\mathcal{S}^{\sigma}_{u(R_j)} = \{o_u \in \mathcal{S} \mid o \text{ occurs in } \sigma\}$ is the set of all unbound operations that refer to another role R_j, $j \neq i$, of the same choreography, that are used in the execution trace σ. θ is conservative when the following holds: $(\langle \mathcal{S}\theta, \mathcal{P}\theta \rangle, S_0) \vdash G$ w.a. $\sigma\theta$



In [2] we have proved that, in general, flexible matches **are not conservative substitution**

A lattice of matches

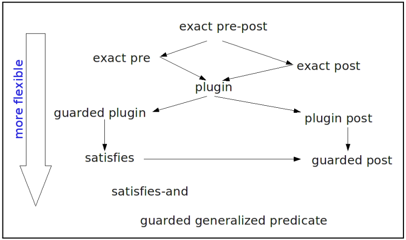
- In the literature it is possible to find many match algorithms, mostly based on the seminal work by Zaremski and Wing [11] on software components, and surveyed in
- "On theory and practice of Assertion Based Software Development Herbert Toth, J. of Object Technology 2005" [9]



A lattice of matches

Given a software component I, with precondition S_{pre} and postcondition S_{post}, and a specification (or query, in the match-making community) Q, with precondition Q_{pre} and postcondition Q_{post}, the most important kinds of relaxed match between Q and S are:

- EM (*Exact Pre/Post Match*): $Q_{pre} \Leftrightarrow S_{pre} \wedge Q_{post} \Leftrightarrow S_{post}$
- EPREM (*Exact Pre Match*): $Q_{pre} \Leftrightarrow S_{pre} \wedge S_{post} \Rightarrow Q_{post}$
- EPOM (*Exact Post Match*): $Q_{pre} \Rightarrow S_{pre} \wedge Q_{post} \Leftrightarrow S_{post}$
- PIM (*Plugin Match*): $Q_{pre} \Rightarrow S_{pre} \wedge S_{post} \Rightarrow Q_{post}$
- POM (*Plugin Post Match*): $S_{post} \Rightarrow Q_{post}$
- GPIM (*Guarded Plugin Match*, a.k.a. Weak-Plugin [8]): $Q_{pre} \Rightarrow S_{pre} \wedge ((S_{pre} \wedge S_{post}) \Rightarrow Q_{post})$
- SM (*Satisfies Match*, a.k.a. relaxed plug-in in [3], plug-in compatibility [5]): $Q_{pre} \Rightarrow S_{pre} \wedge (Q_{pre} \wedge S_{post} \Rightarrow Q_{post})$
- GPOM (*Guarded Post Match*, a.k.a. Weak-Post [8]): $((S_{pre} \wedge S_{post}) \Rightarrow Q_{post})$
- GGP (*Guarded-Generalized Predicate*): $(Q_{pre} \Rightarrow S_{pre}) \wedge ((S_{pre} \Rightarrow S_{post}) \Rightarrow (Q_{pre} \Rightarrow Q_{post}))$



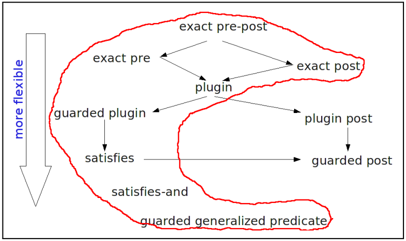
A lattice of matches

Given a software component S, with precondition S_{pre} and postcondition S_{post}, and a specification (or query, in the match-making community) Q, with precondition Q_{pre} and postcondition Q_{post}, the most important kinds of relaxed match between Q and S are:

- EM (*Exact Pre/Post Match*): $Q_{pre} \Leftrightarrow S_{pre} \wedge Q_{post} \Leftrightarrow S_{post}$
- EPREM (*Exact Pre Match*): $Q_{pre} \Leftrightarrow S_{pre} \wedge S_{post} \Rightarrow Q_{post}$
- EPOM (*Exact Post Match*): $Q_{pre} \Rightarrow S_{pre} \wedge Q_{post} \Leftrightarrow S_{post}$
- PIM (*Plugin Match*): $Q_{pre} \Rightarrow S_{pre} \wedge S_{post} \Rightarrow Q_{post}$
- POM (*Plugin Post Match*): $S_{post} \Rightarrow Q_{post}$
- GPIM (*Guarded Plugin Match*, a.k.a. Weak-Plugin [8]): $Q_{pre} \Rightarrow S_{pre} \wedge ((S_{pre} \wedge S_{post}) \Rightarrow Q_{post})$
- SM (*Satisfies Match*, a.k.a. relaxed plug-in in [3], plug-in compatibility [5]): $Q_{pre} \Rightarrow S_{pre} \wedge (Q_{pre} \wedge S_{post} \Rightarrow Q_{post})$
- GPOM (*Guarded Post Match*, a.k.a. Weak-Post [8]): $((S_{pre} \wedge S_{post}) \Rightarrow Q_{post})$
- GGP (*Guarded-Generalized Predicate*): $(Q_{pre} \Rightarrow S_{pre}) \wedge ((S_{pre} \Rightarrow S_{post}) \Rightarrow (Q_{pre} \Rightarrow Q_{post}))$

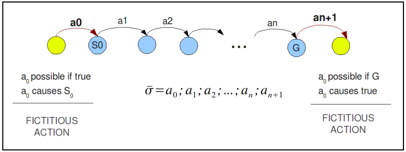
Definition (Re-use ensuring match [3])

A specification match M is re-use ensuring iff for any S and Q, $M(S,Q) \wedge \{S_{pre}\}S\{S_{post}\} \Rightarrow \{Q_{pre}\}S\{Q_{post}\}$.



Verrifying conservative matches

- Intuitively, we take into account the *dependencies* between operations, which produce as effects fluents, that are used as preconditions by subsequent operations



- The idea is to verify that the "causal chains" which allow the execution of the sequence of operations, are not broken after the substitution
- The obvious hypothesis is that we have a choreography and that we know that it allows to achieve the goal of interest, i.e. that there is an execution σ, which allows the achievement of the goal
- We will use this trace for defining a set of constraints that, whenever satisfied by a substitution obtained by a re-use ensuring match, guarantee that the substitution is also conservative
- This is a "sufficient" condition because there might exist conservative substitutions that do not satisfy this set of constraints

Verrifying conservative matches

Definition

A substitution θ is called *uninfluential* iff for any substitution $[s/o_u]$ in θ, all beliefs in $E_s(s) - E_s(o_u)$ are uninfluential fluents w.r.t. σ.

Proposition

Let M be a re-use ensuring match, any substitution θ_M that is uninfluential is also conservative.

Theorem

Let M be a re-use ensuring match, $S_i = \langle \mathcal{S}, \mathcal{P} \rangle$ be a service which plays a role R_i in a given choreography, and G a query such that, $(\langle \mathcal{S}, \mathcal{P} \rangle, S_0) \vdash G$ w.a. σ, where S_O is the initial state. Let θ_M be a substitution for all unbound operations of S_i that refer to another role R_j played by the service S_j, $j \neq i$. The problem of determining whether θ_M is conservative w.r.t. G is decidable.

Verrifying conservative matches

Definition

A substitution θ is called *uninfluential* iff for any substitution $[s/o_u]$ in θ, all beliefs in $E_s(s) - E_s(o_u)$ are uninfluential fluents w.r.t. σ.

Proposition

Let M be a re-use ensuring match, any substitution θ_M that is uninfluential is also conservative.

Theorem

Let M be a re-use ensuring match, $S_i = \langle \mathcal{S}, \mathcal{P} \rangle$ be a service which plays a role R_i in a given choreography, and G a query such that, $(\langle \mathcal{S}, \mathcal{P} \rangle, S_0) \vdash G$ w.a. σ, where S_O is the initial state. Let θ_M be a substitution for all unbound operations of S_i that refer to another role R_j played by the service S_j, $j \neq i$. The problem of determining whether θ_M is conservative w.r.t. G is decidable.

Verrifying conservative matches

Definition

A substitution θ is called *uninfluential* iff for any substitution $[s/o_u]$ in θ, all beliefs in $E_s(s) - E_s(o_u)$ are uninfluential fluents w.r.t. σ.

Proposition

Let M be a re-use ensuring match, any substitution θ_M that is uninfluential is also conservative.

Theorem

Let M be a re-use ensuring match, $S_i = \langle \mathcal{S}, \mathcal{P} \rangle$ be a service which plays a role R_i in a given choreography, and G a query such that, $(\langle \mathcal{S}, \mathcal{P} \rangle, S_0) \vdash G$ w.a. σ, where S_O is the initial state. Let θ_M be a substitution for all unbound operations of S_i that refer to another role R_j played by the service S_j, $j \neq i$. The problem of determining whether θ_M is conservative w.r.t. G is decidable.

Conclusions

- Locality is a problem when a service has to supply a set of operations to play a role
- Knowledge about the choreography is knowledge about the context, can be exploited to define a WS matching process that preserves goals after substitution
- A WS can reason about:
  - whether playing a role
  - which partner is better for it
- Ontology matching
- More sophisticated reasoning mechanism (reasoning on coalitions?)

G. Alonso, F. Casati, H. Kuno, and V. Machiraju.
Web Services.
Springer, 2004.

M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella.
Service selection by choreography-driven matching.
In T. Gschwind and C. Pautasso, editors, *Emerging Web Services Technology*, volume II of *Whitestein Series in Software Agent Technologies and Autonomic Computing*, chapter 1, pages 5–22.
Birkhäuser, September 2008.

Y. Chen and B. H. C. Cheng.
Foundations of Component-Based Systems, chapter A Semantic Foundation for Specification Matching, pages 91–109.
Cambridge Univ. Press, 2000.

Dieter Fensel, Holger Lausen, Jos de Bruijn, Michael Stollberg, Dumitru Roman, and Axel Polleres.
Enabling Semantic Web Services : The Web Service Modeling Ontology.

Springer.

B. Fischer and G. Snelting.
Reuse by contract.
In *ESEC/FSE-Workshop on Foundations of Component-Based Systems*, 1997.

Bertrand Meyer.
Applying "design by contract".
Computer, 25(10):40–51, 1992.

OWL-S Coalition.
http://www.daml.org/services/owl-s/.

John Penix and Perry Alexander.
Efficient specification-based component retrieval.
Automated Software Engg., 6(2):139–170, 1999.

Herbert Toth.
On theory and practice of assertion based software development.
Journal of Object Technology, 4(2):109–129, 2005.

WS-CDL.
http://www.w3.org/tr/ws-cdl-10/.

A. Moormann Zaremski and J. M. Wing.
Specification matching of software components.
ACM Transactions on SEM, 6(4):333–369, 1997.