Reasoning about interaction protocols for web service composition

Matteo Baldoni, Cristina Baroglio, Laura Giordano, Alberto Martelli, Viviana Patti

Protocolli descritti mediante scambio di messaggi.

Teoria delle azioni per esprimere azioni comunicative atomiche e composte (protocolli).

Composizione: dati n servizi, costruire dei **piani** per interagire con i servizi rispettando un insieme di vincoli.

Due strumenti per ragionare su protocolli:

- DyLOG
- DLTL: logica dinamica + temporale linear-time.

DyLOG: overview [ICTCS 2003, AI*IA, 2003]

- A language for programming agents, based on a *modal approach* for reasoning about actions and change in *a logic programming setting*
- The behavior of an agent ag_i is described by a domain description DD^{agi} which consists of:
  - Π
    1. a set of primitive actions: preconditions and effects
    2. a set of sensing actions: interaction with the word
    3. a set of complex actions defined by means of prolog-like procedures
  - + S_0 a description of the initial situation (a set of initial beliefs)
  - + $CKit^{agi}$ a communication kit
- mentalistic approach

DyLOG Overview

- For each action a^{agi} atomic or complex ⟶ a modal operator $[a^{agi}]$
  $[a^{agi}]\alpha$ means that α holds after every execution of action a by agent agi

K

$[a^{agi}]\alpha$
$< a^{agi} > \alpha$

S4

$\Box\,\alpha$

K

$Done(a^{agi})T$

Always

$\Box\,\alpha \supset [a^{agi}]\alpha$

Awareness

$Done(a^{agi})\alpha \supset B^{agi}Done(a^{agi})\alpha$

- epistemic operator B^{agi}: The mental state includes beliefs and nested beliefs of rank 2 ($B^{agi}B^{agj}\ I$) of type KD45 for representing *what the other agents believe* and *for reasoning on how they can be affected by communicative actions*

KD + 45

$B^{agi}\alpha$

$M^{agi}\alpha \equiv \neg B^{agi}\neg\alpha$

- complex actions operators from the dynamic logic:
  - sequencing: ";"
  - non-det choice: "U"
  - test: "?"

The communication kit: Ckit

- *Integrating a communication theory in the general agent theory*:

$$DD^{agi} = (\Pi, Ckit^{agi}, S_0)$$

Π_C

a set of simple action laws to define the agent *speech acts* (inform, query, request, …)

Π_{CP}

a set of procedure axioms to specify the agent *conversation policies (i.e. internal representation of protocols)*

Π_{Sget}

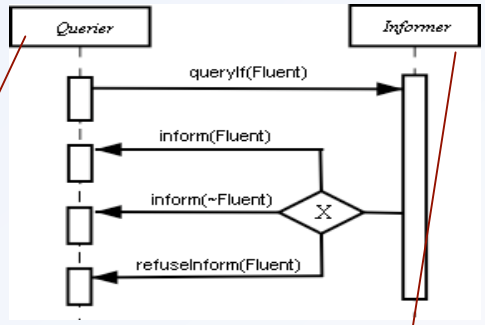
a set of sensing axioms to represent *messages from other agents*

speech acts and conversation policies are, as well, represented as primitive actions, sensing actions and procedure definitions of a DyLOG agent theory

CKit: conversation protocols

- Individual speech acts are used in the context of *predefined conversation protocols*, that specify communication patterns [Pitt & Madami 2000]

- Agents have a *subjective perception* of communication with the others (see also [Endriss et al., IJCAI'03]) → an agent represents a protocol as one of *its* (conversation) policies



Agent
Unified
Modelling
Language
sequence
diagram

$$\langle \text{yes\_no\_query}_Q(Self, Other, Fluent)\rangle\varphi \subset \langle \text{queryIf}(Self, Other, Fluent);\ \text{get\_answer}(Self, Other, Fluent)\rangle\varphi$$

$$[\text{get\_answer}(Self, Other, Fluent)]\varphi \equiv [\text{inform}(Other, Self, Fluent) \cup \text{inform}(Other, Self, \neg Fluent) \cup \text{refuseInform}(Other, Self, Fluent)]\varphi$$

$$\langle \text{yes\_no\_query}_I(Self, Other, Fluent)\rangle\varphi \subset \langle \text{get\_start}(Self, Other, Fluent);\ \mathcal{B}^{Self} Fluent?; \text{inform}(Self, Other, Fluent)\rangle\varphi$$

$$\langle \text{yes\_no\_query}_I(Self, Other, Fluent)\rangle\varphi \subset \langle \text{get\_start}(Self, Other, Fluent);\ \mathcal{B}^{Self} \neg Fluent?; \text{inform}(Self, Other, \neg Fluent)\rangle\varphi$$

$$\langle \text{yes\_no\_query}_I(Self, Other, Fluent)\rangle\varphi \subset \langle \text{get\_start}(Self, Other, Fluent);\ \mathcal{U}^{Self} Fluent?; \text{refuseInform}(Self, Other, Fluent)\rangle\varphi$$

$$[\text{get\_start}(Self, Other, Fluent)]\varphi \equiv [\text{queryIf}(Other, Self, Fluent)]\varphi$$

DyLOG: overview

- Given a domain description, we can reason about it by means of existential queries

$$<p_1><p_2> \ldots <p_n>Fs$$

  - *Temporal projection*, when p_k's are all atomic actions
  - *Procedural planning*, when p_k's are both atomic and complex actions: “Is there an execution trace of p_1, ..., p_n (a plan) leading to a state where *Fs* holds?"

- *Notice that*: as a difference with classical planning, the procedure definitions constraint the search space
- *Linear plan:* it contains assumptions on the sensing outcome
- *Conditional plan*: for each sensing action it contains as many branches as possible action outcomes
- Correct plans

Reasoning about conversations

- Given a domain description, we can reason about it by means of *existential queries*:

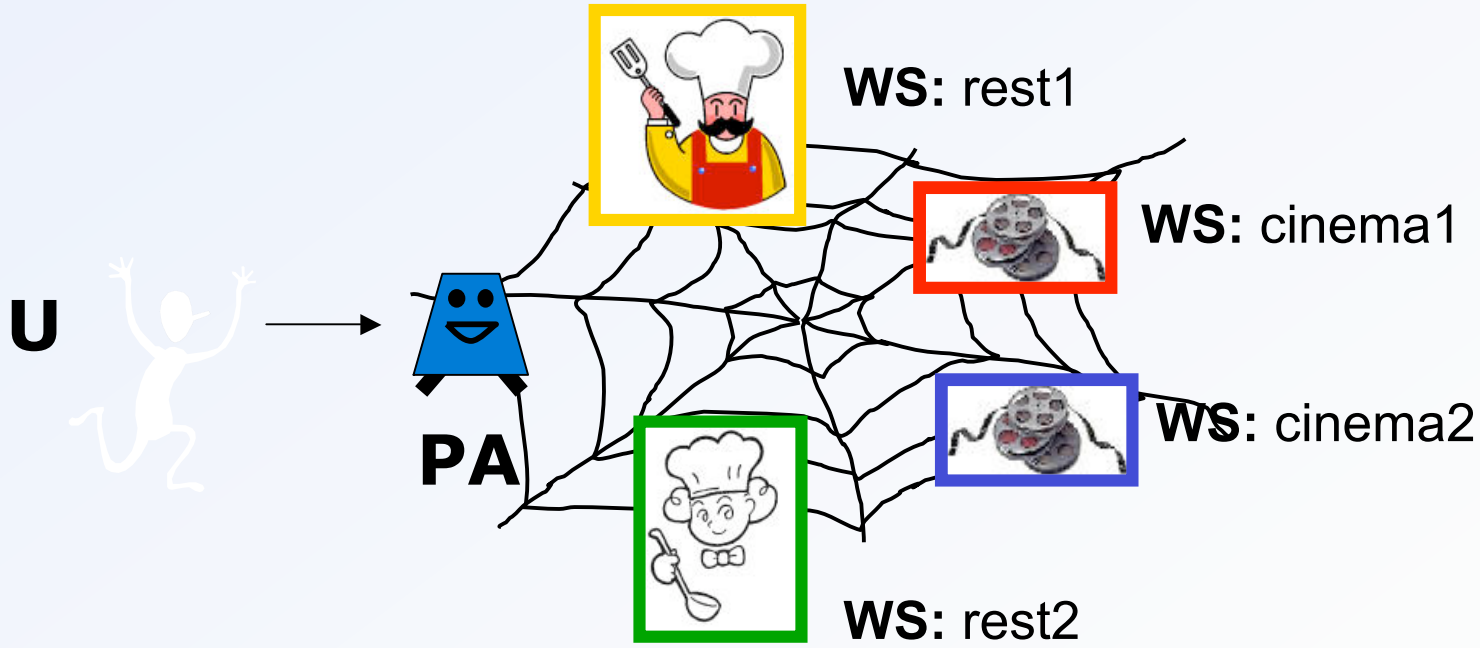
$?\ \langle p_m \rangle Fs$

Is there an execution of pm (a plan) leading to a state where Fs holds?

  - p_m is an interaction protocol
  - We look for a conversation, which is an instance of the protocol described by p_m, after which the condition *Fs* holds
  - We treat *get message* actions as *sensing actions*, whose outcome cannot be known at planning time.
- Goal directed proof procedure, based on negation as failure (dealing with persistency) [ICTCS 2003]

A semantic web scenario

The agent PA is requested to organize day out:



- The user wants **to eat out**
- Watch a certain **movie**
- Benefit of a promotion on the cinema ticket
- And to avoid the use of credit card

Composition of WS: example 1/4

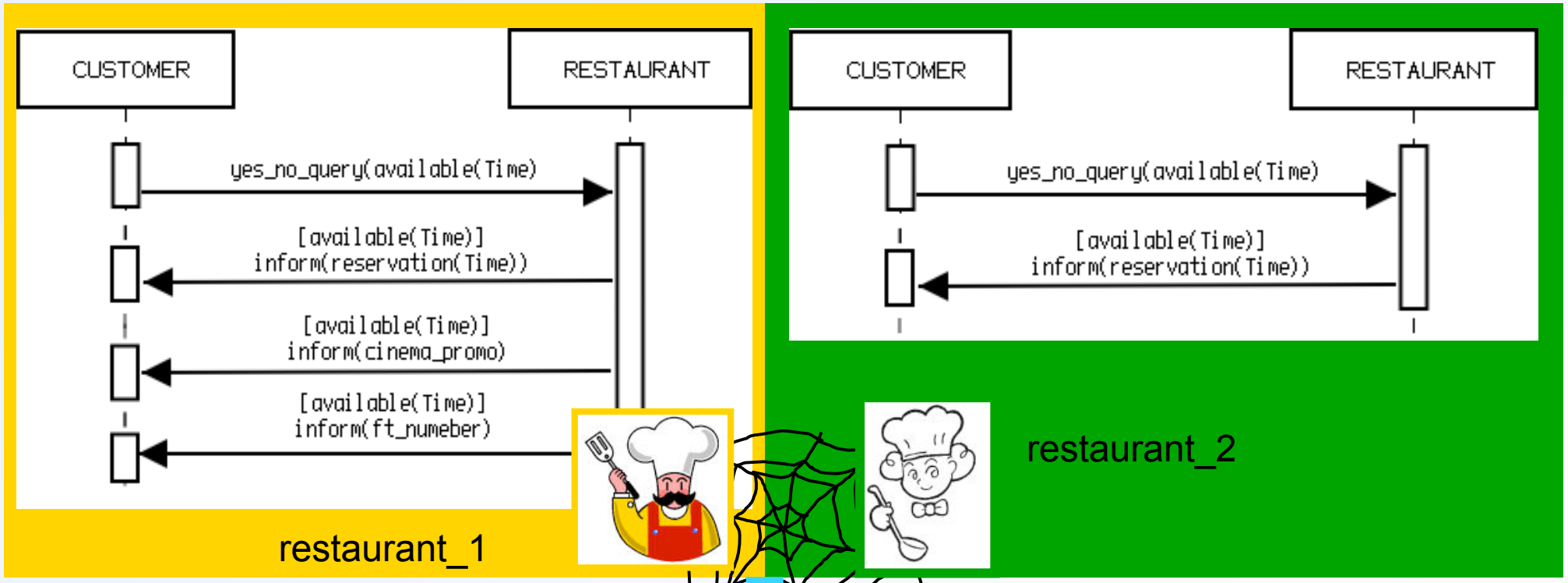
- Among initial beliefs:

$$\mathcal{B}^{pa}\, service(restaurant, restaurant1, reserv\_rest\_1_C)$$
$$\mathcal{B}^{pa}\, service(restaurant, restaurant2, reserv\_rest\_2_C)$$
$$\mathcal{B}^{pa}\, service(cinema, cinema1, reserv\_cinema\_1_C)$$
$$\mathcal{B}^{pa}\, service(cinema, cinema2, reserv\_cinema\_2_C)$$

KB

  - + ...U's credit card number
  - U's desire not to use it in the current transaction
  - No ticket for nausicaa has been booked yet
  - Hypothesis on the WS mental state (e.g. cinema1 does not know U's credit card number)

Interaction Protocols of WS: example 2/4

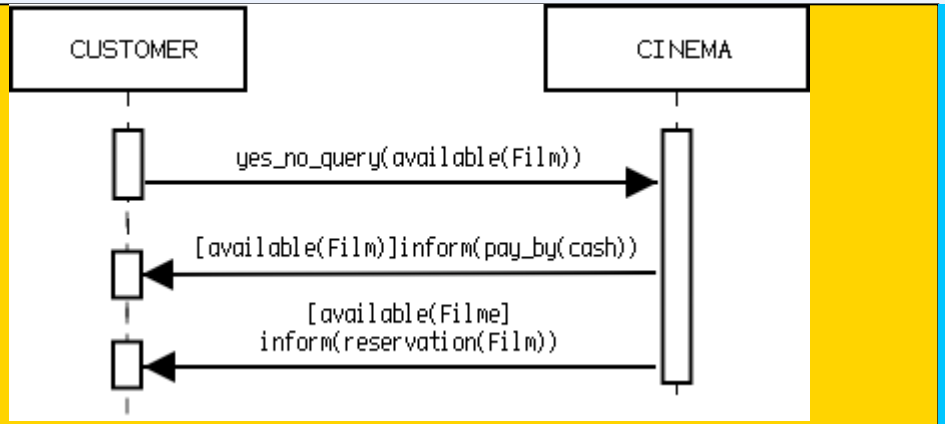


$$\langle \text{reserv\_rest\_1}_C(Self, WebS, Time)\rangle\varphi \subset \langle \text{yes\_no\_query}_Q(Self, WebS, available(Time))\ ;\ \mathcal{B}^{Self}\, available(Time)?\ ;\ \text{get\_info}(Self, WebS, reservation(Time))\ ;\ \text{get\_info}(Self, WebS, cinema\_promo)\ ;\ \text{get\_info}(Self, WebS, ft\_number)\rangle\varphi$$

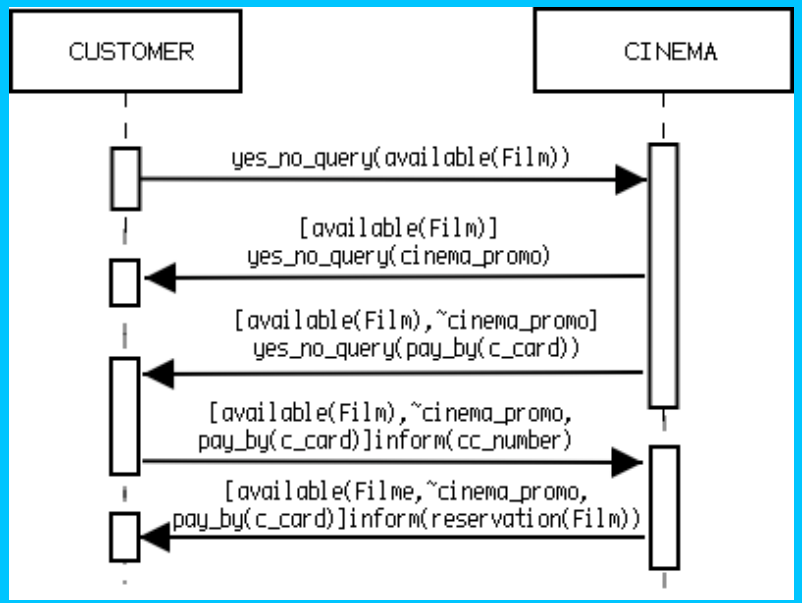
$$\langle \text{reserv\_rest\_2}_C(Self, WebS, Time)\rangle\varphi \subset \langle \text{yes\_no\_query}_Q(Self, WebS, available(Time))\ ;\ \mathcal{B}^{Self}\, available(Time)?\ ;\ \text{get\_info}(Self, WebS, reservation(Time))\rangle\varphi$$

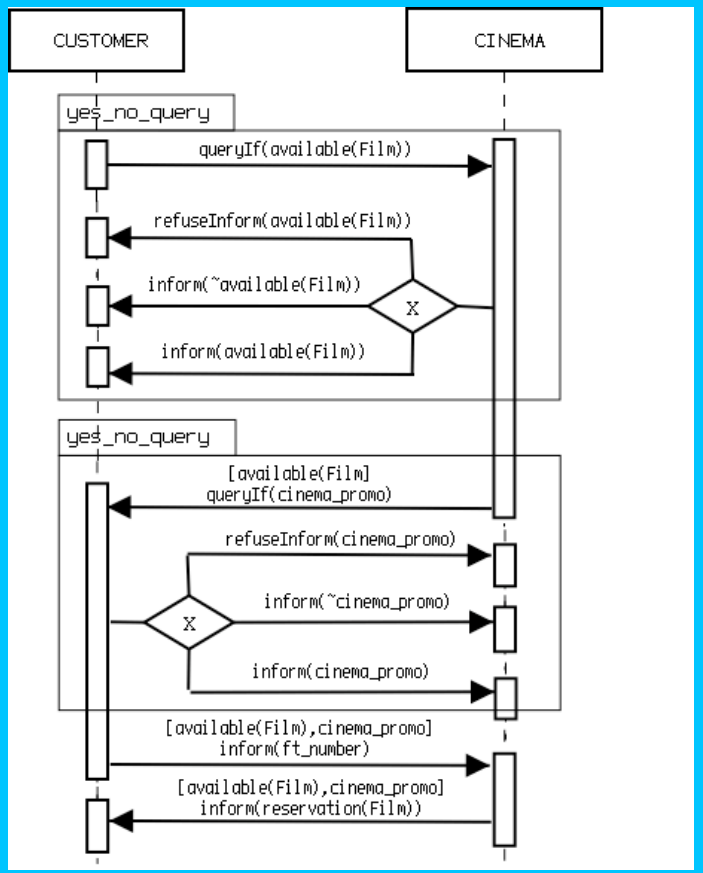
KB

Interaction Protocols of WS: example 3/4



cinema_2





cinema_1

Composition of WS: example 4/4

- Combine services...

KB

$$\langle comp\_services([\,])\rangle\varphi \supset true$$
$$\langle comp\_services([[TypeService, Name, Data]|Services])\rangle\varphi \supset$$
$$\langle \mathcal{B}^{pa} service(TypeService, Name, Protocol)\ ;$$
$$Protocol(pa, Name, Data)\ ;$$
$$comp\_services(Services)\rangle\varphi$$

compose-by-sequencing

- Query

$$\langle comp\_services([[restaurant, R, dinner], [cinema, C, nausicaa]])\rangle$$
$$(\mathcal{B}^{pa} cinema\_promo \wedge \mathcal{B}^{pa} reservation(dinner) \wedge$$
$$\mathcal{B}^{pa} reservation(nausicaa) \wedge \mathcal{B}^{pa} \neg \mathcal{B}^{C} cc\_number \wedge \mathcal{B}^{pa} \mathcal{B}^{C} ft\_number)$$

- The answer is a linear plan...

there is no other execution trace of comp_services that satisfies the goal

$$\text{queryIf}(pa, restaurant1, available(dinner))\ ;$$
$$\text{inform}(restaurant1, pa, available(dinner))\ ;$$
$$\text{inform}(restaurant1, pa, reservation(dinner))\ ;$$
$$\text{inform}(restaurant1, pa, cinema\_promo)\ ;$$
$$\text{inform}(restaurant1, pa, ft\_number)\ ;$$
$$\text{queryIf}(pa, cinema1, available(nausicaa))\ ;$$
$$\text{inform}(cinema1, pa, available(nausicaa))\ ;$$
$$\text{queryIf}(cinema1, pa, cinema\_promo)\ ;$$
$$\text{inform}(pa, cinema1, cinema\_promo)\ ;$$
$$\text{inform}(pa, cinema1, ft\_number)\ ;$$
$$\text{inform}(cinema1, pa, reservation(nausicaa))$$