A Logic-Programming Semantics of Services

Ionuţ Ţuţu[1,2] and José Luiz Fiadeiro[1]

[1] Dept. Computer Science, Royal Holloway University of London, UK
[2] Institute of Mathematics of the Romanian Academy,
Research group of the project ID-3-0439
ittutu@gmail.com, jose.fiadeiro@rhul.ac.uk

Abstract. We develop formal foundations for notions and mechanisms needed to support service-oriented computing. Our work provides semantics for the service overlay by abstracting concepts from logic programming. It draws a strong analogy between the discovery of a service that can be bound to a client application and the search for a clause that can be used for computing an answer to a query. In addition, it describes the process of binding services and the reconfiguration of applications as service-oriented derivatives of unification and resolution.

1 Introduction

Service-Oriented Computing. SOC is a recent paradigm that addresses computation in 'global computers' – computational infrastructures available globally in which software applications can discover and bind dynamically, at run time, to services offered by providers. Whereas the paradigm has been effectively in use for a more than a decade in the form of Web services [1] or Grid computing [13], research into its formal foundations has lagged somewhat behind, partly because of our lack of understanding of (or agreement on) what is really new about the paradigm.

It is fair to say that significant advances have been made towards formalising new forms of distributed computation that have arisen around the notion of service (e.g. choreography [18]), notably through several variants of the π-calculus. However, SOC raises more profound challenges at the level of the structure of systems thanks to their ability to discover and bind dynamically, in a non-programmed way, to other systems. The structure of the systems we are now creating in the virtual space of computational networks is intrinsically dynamic, a phenomenon hitherto unknown. Formalisms such as the π-calculus do not address these structural properties of systems.

Towards that end, we have investigated algebraic structures that account for modularity (e.g. [10,12]) – the way services are orchestrated as composite structures of components and how binding is performed through interaction protocols – and the mechanisms through which discovery can be formalised in terms of logical specifications of required/provided services and constraint optimisation for service-level agreements (e.g. [9,11]). In the present paper, we take further this research to address the operational aspects behind *dynamic* discovery and binding, i.e. the mechanisms through which applications discover and bind, at run time, to services. Our aim is to develop an abstract, foundational setting – i.e. independent of the specific technologies that are currently deployed, such as SOAP for message-exchange protocols and the UDDI for

description, discovery, and integration – that combines both *declarative* and *dynamic* semantics of services. The challenge here is to develop an integrated algebraic framework that accounts for (*a*) logical specifications of services, (*b*) the way models of those specifications capture orchestrations of components that depend on externally provided services to be discovered, and (*c*) the way the discovery of services and the binding of their orchestrations to client applications can be expressed in logical/algebraic terms.

Logic Programming. The approach that we propose to develop to meet this challenge builds on (Horn-clause) logic programming (LP) – the paradigm that epitomises the integration of declarative and operational aspects of logic. In LP, clauses have a declarative semantics as disjunctions of literals that express relationships over a domain (the Herbrand universe), and an operational semantics that derives from resolution and term unification: definite clauses (from a program) are used to resolve queries (expressed as goal clauses) by generating new queries and, through term unification, computing partial answers as substitutions for the variables of the original query.

In a nutshell, the analogy with SOC that we propose to develop works as follows:

- The Herbrand universe consists of all possible service orchestrations with no dependencies on external services ('ground services').
- Variables correspond to requires-points of orchestrations, i.e. dependencies on external services that need to be discovered.
- Terms correspond to services delivered through provides-points of orchestrations.
- Definite clauses express properties of provides-points (head) and requires-points (body) of service orchestrations – what in [11] we call service modules. Their declarative semantics is that, when bound to applications that deliver services satisfying the properties of the requires-points, the orchestrations will deliver at the provides-points services that satisfy the specified properties.
- Goal clauses express properties of orchestrations of services that an application requires in order to fulfil given business goals – what in [11] we call activity modules.
- Programs correspond to service repositories.
- Resolution and term unification account for service discovery by matching required properties with provided ones and the binding of required with provided services.

Structure of the Paper. In Sect. 2 we propose an algebraic model of service orchestrations as asynchronous relational networks similar to those used in [8], and we define the logical framework over which we can express properties of the interaction points through which such networks can be interconnected. We prove that the resulting logic constitutes an institution [14], which provides the declarative semantics of our approach to SOC. In Sect. 3 we show how clauses, unification and resolution can be defined over that institution, thus providing the corresponding operational semantics of SOC.

2 Asynchronous Relational Networks

Our first contribution is the formulation of an institution of asynchronous relational networks. This accounts for the service-oriented counterpart of the declarative aspects of conventional LP within first-order logic, in particular the key role played by variables – the structures over which the computational aspects of LP operate.

The concepts discussed here depend upon elements of linear temporal logic (LTL). However, the proposed theory is largely independent of the logical framework of choice, and can be easily adapted to any institution such that

- the category of signatures is (finitely) co-complete;
- there exist co-free models along any signature morphism, i.e. the reduct functor of any signature morphism has a right adjoint;
- the category of models of any signature has products;
- any model homomorphism reflects the satisfaction of any sentence.

In order to capture a more operational notion of service orchestration, we work with a variant MA-LTL of LTL whose models are not traces, but Muller automata [17]; by definition, an automaton satisfies a sentence if and only if every trace accepted by the automaton satisfies the considered sentence. A detailed definition of MA-LTL together with proofs of the four aforementioned properties can be found in Appendices B and C.

2.1 Signatures and Signature Morphisms

We start by defining the category $\mathbb{A}RN$ of signatures and signature morphisms of our institution. These are asynchronous relational networks similar to those defined in [8] except that we use Muller automata instead of sets of traces as models of behaviour, and hypergraphs instead of graphs.

Following [8], we regard service components as networks of processes that interact asynchronously by exchanging messages through communication channels. Messages are considered to be atomic units of communication. They can be grouped into structures – ports – through which processes and channels can be interconnected.

Ports are sets of messages with attached polarities. As in [2,3] we distinguish between outgoing or published messages (labelled with a minus sign) and incoming or delivered messages (labelled with a plus sign).

Definition 1 (Port). *A* port M *is a pair* $\langle M^-, M^+ \rangle$ *of disjoint finite sets of messages. The set of all* messages *of* M *is given by the union* $M^- \cup M^+$*, usually denoted by* M*. Every port* M *determines the set of* actions $A_M = A_{M^-} \cup A_{M^+}$*, where* A_{M^-} *is the set* $\{m! \mid m \in M^-\}$ *of* publications*, and* A_{M^+} *is the set* $\{m¡ \mid m \in M^+\}$ *of* deliveries*.*

Processes are defined by sets of interaction points labelled with ports and by Muller automata that describe process behaviour in terms of observable actions.

Definition 2 (Process). *A* process $(\{M_x \mid x \in X\}, \Lambda)$ *consists of a finite set* X *of* interaction points*, each point* $x \in X$ *being labelled with a port* M_x*, and a Muller automaton* Λ *over the alphabet* $\mathcal{P}(A_M)$*, where* M *is the port given by*

$$M^{\mp} = \biguplus_{x \in X} M_x^{\mp} = \{x.m \mid x \in X, m \in M_x^{\mp}\} \ .$$

Example 1. In Fig. 1 we depict a process JourneyPlanner that provides directions from a source to a target location. The process interacts with the environment through two ports: JP_1 and JP_2. The first port is used for communicating with potential client processes – the request for directions (including the source and the target locations) is encoded into the incoming message planJourney, while the response is represented by the

outgoing message directions. The second port defines messages that JourneyPlanner exchanges with other processes in order to complete its task – the outgoing message getRoutes can be seen as a query for all possible routes between the specified source and target locations, and the incoming messages routes and timetables define the result of the query and the timetables of the available transport services for the selected routes.

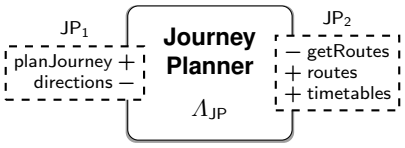


Fig. 1. The JourneyPlanner process

The behaviour of JourneyPlanner is given by the Muller automaton depicted in Fig. 2, whose final state sets contain q_0 whenever they contain q_5. We can describe it informally as follows: whenever JourneyPlanner receives a request planJourney it immediately initiates the search of the available routes by sending the message getRoutes; it then waits for the delivery of the routes and of the corresponding timetables; once it receives both it compiles the directions and replies to the client.

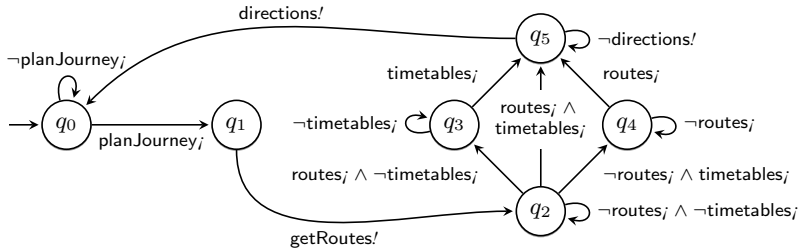


Fig. 2. The JourneyPlanner automaton[3]

Note that every polarity-preserving map θ between ports M and M' defines a function $A_\theta \colon A_M \to A_{M'}$, often denoted simply by θ, that maps every publication action $m!$ into $\theta(m)!$ and every delivery action $m¡$ into $\theta(m)¡$.

Fact 1. For any process $(\{M_x \mid x \in X\}, \Lambda)$, the injections $\{x\_ \colon A_{M_x} \to A_M\}_{x \in X}$ define a co-product in the category of MA-LTL-signatures.

Processes communicate by transmitting messages through channels. As in [3,8], channels are bidirectional: they may transmit both incoming and outgoing messages.

Definition 3 (Channel). *A* channel (M, Λ) *consists of a finite set* M *of* messages *and a Muller automaton* Λ *over the alphabet* $\mathcal{P}(A_M)$*, where* A_M *is the union* $A_M^- \cup A_M^+$ *of* $A_M^- = \{m! \mid m \in M\}$ *and* $A_M^+ = \{m¡ \mid m \in M\}$.

[3] In the graphical representation the transitions are labelled with propositional sentences; this means that there exists a transition for any model (set of actions) of the considered sentence.

In order to enable given processes to exchange messages, channels need to be attached to their ports, thus forming connections.

Definition 4 (Connection). *A* connection $(\{\mu_x \colon M \rightharpoonup M_x \mid x \in X\}, \Lambda)$ *between the ports* $\{M_x \mid x \in X\}$ *consists of a channel* (M, Λ) *and a finite family of partial* attachment injections $\{\mu_x \colon M \rightharpoonup M_x \mid x \in X\}$ *such that* $M = \bigcup_{x \in X} \mathrm{dom}(\mu_x)$ *and* $\mu_x^{-1}(M_x^{\mp}) \subseteq \bigcup_{y \in X \setminus \{x\}} \mu_y^{-1}(M_y^{\pm})$*, for any* $x \in X$.

This notion of connection differs from the one found in [8] in that messages can be transmitted between more than two ports. The additional condition ensures that messages are well paired: every published message of M_x, for $x \in X$, is paired with a delivered message of M_y, for $y \in X \setminus \{x\}$, and vice versa.

Example 2. In order to illustrate how the process JourneyPlanner can send or receive messages, we consider the connection C depicted in Fig. 3 that moderates the flow of messages between the port JP_2 and two other ports, R_1 and R_2.

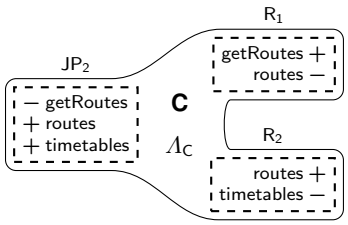


Fig. 3. The JourneyPlanner's connection

The underlying channel of C is given by the set of messages $M = \{g, r, t\}$ together with the automaton Λ_{C} that specifies the delivery of all published messages without any delay. Λ_{C} can be built as the product of the automata Λ_m, for $m \in M$, whose transition map is depicted in Fig. 4, and whose sets of states are all marked as final.

The channel is attached to the ports JP_2, R_1 and R_2 through the partial injections:

- $\mu_{\mathsf{JP}_2} = \{g \mapsto \mathsf{getRoutes}, r \mapsto \mathsf{routes}, t \mapsto \mathsf{timetables}\}$,
- $\mu_{\mathsf{R}_1} = \{g \mapsto \mathsf{getRoutes}, r \mapsto \mathsf{routes}\}$ and
- $\mu_{\mathsf{R}_2} = \{r \mapsto \mathsf{routes}, t \mapsto \mathsf{timetables}\}$.

Note that the actual senders and receivers of messages are specified through the attachment injections. For example, g is delivered only to the port R_1 (because μ_{R_2} is not defined on g), while r is simultaneously delivered to both JP_2 and R_2.

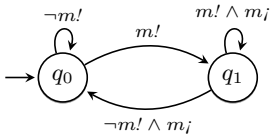


Fig. 4. The Λ_m automaton

As already suggested in Ex. 1 and 2, processes and connections have dual roles and interpret the polarities of the messages accordingly. In this sense, processes are responsible for publishing messages (i.e. delivered messages are inputs and published messages are outputs), while connections are responsible for delivering messages.

We clarify this dual nature of connections $(\{\mu_x \colon M \rightharpoonup M_x \mid x \in X\}, \Lambda)$ by defining partial translations $\{A_{\mu_x} \colon A_M \rightharpoonup A_{M_x} \mid x \in X\}$ given by

- $\mathrm{dom}(A_{\mu_x}) = \{m! \mid m \in \mu_x^{-1}(M_x^-)\} \cup \{m¡ \mid m \in \mu_x^{-1}(M_x^+)\}$,
- $A_{\mu_x}(m!) = \mu_x(m)!$ for all $m \in \mu_x^{-1}(M_x^-)$,
- $A_{\mu_x}(m¡) = \mu_x(m)¡$ for all $m \in \mu_x^{-1}(M_x^+)$.

We often designate the partial maps A_{μ_x} simply by μ_x if there is no risk of confusion.

Fact 2. Every connection $(\{\mu_x \colon M \rightharpoonup M_x \mid x \in X\}, \Lambda)$ defines a family of spans $\left\{A_M \xleftarrow{\supseteq} \mathrm{dom}(\mu_x) \xrightarrow{\mu_x} A_{M_x}\right\}_{x \in X}$ in the category of MA-LTL-signatures.

We can now define asynchronous networks of processes as hypergraphs having vertices labelled with ports and hyperedges labelled with processes and connections.

Definition 5 (Hypergraph). *An* (edge-labelled) hypergraph (X, E, γ) *consists of a set* X *of* vertices *or* nodes*, a set* E *of* hyperedges *disjoint from* X*, and an* incidence map $\gamma \colon E \to \mathcal{P}(X)$ *defining for every hyperedge* $e \in E$ *a non-empty set* $\gamma_e \subseteq X$ *of vertices it is incident with. A hypergraph* (X, E, γ) *is said to be* edge-bipartite *if* E *is partitioned into two subsets* F *and* G *such that no adjacent hyperedges belong to the same partition, i.e. for every two hyperedges* $e_1, e_2 \in E$ *such that* $\gamma_{e_1} \cap \gamma_{e_2} \neq \emptyset$*, either* $e_1 \in F$ *and* $e_2 \in G$*, or* $e_1 \in G$ *and* $e_2 \in F$.

Hypergraphs have been used extensively in the context of graph-rewriting-based approaches to concurrency, including SOC (e.g., [4,7]). We use them instead of graphs [8] because they offer a more flexible mathematical framework for handling the notions of variable and variable binding that we require in Sect. 3.

Definition 6 (Asynchronous Relational Network – ARN). *An* asynchronous relational network $\alpha = (X, P, C, \gamma, M, \mu, \Lambda)$ *consists of a (finite) edge-bipartite hypergraph* (X, P, C, γ) *of* points $x \in X$*,* computation hyperedges $p \in P$ *and* communication hyperedges $c \in C$*, together with*

- *a port* M_x *for every point* $x \in X$*,*
- *a process* $(\{M_x \mid x \in \gamma_p\}, \Lambda_p)$ *for every hyperedge* $p \in P$*, and*
- *a connection* $(\{\mu_x^c \colon M_c \rightharpoonup M_x \mid x \in \gamma_c\}, \Lambda_c)$ *for every hyperedge* $c \in C$.

Example 3. By putting together the process and the connection presented in Ex. 1 and 2, we obtain the ARN JourneyPlanner depicted in Fig. 5. Its underlying hypergraph consists of the points JP_1, JP_2, R_1 and R_2, the computation hyperedge JP, the communication hyperedge C, and the incidence map γ given by $\gamma_{\mathsf{JP}} = \{\mathsf{JP}_1, \mathsf{JP}_2\}$ and $\gamma_{\mathsf{C}} = \{\mathsf{JP}_2, \mathsf{R}_1, \mathsf{R}_2\}$.

An *interaction-point* of an ARN α is a point of α that is not bound to both computation and communication hyperedges. We distinguish between requires-points and provides-points, as follows.

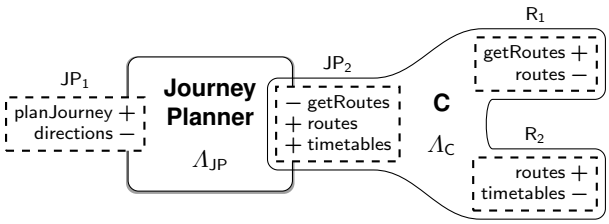


Fig. 5. The JourneyPlanner ARN

Definition 7 (Requires and Provides-point). *A* requires-point *of an ARN α is a point of α that is incident only with a communication hyperedge. Similarly, a* provides-point *of α is a point incident only with a computation hyperedge.*

Morphisms of ARNs can be defined as injective homomorphisms between their underlying hypergraphs that preserve all labels, except those associated with requires-points.

Definition 8 (Homomorphism of Hypergraphs). *A* homomorphism h *between hypergraphs (X, E, γ) and (X', E', γ') consists of functions $h^v \colon X \to X'$ and $h^e \colon E \to E'$ such that for any $x \in X$ and $e \in E$, $x \in \gamma_e$ if and only if $h^v(x) \in \gamma'_{h^e(e)}$.*

Definition 9 (Morphism of ARNs). *Given two ARNs $\alpha = (X, P, C, \gamma, M, \mu, \Lambda)$ and $\alpha' = (X', P', C', \gamma', M', \mu', \Lambda')$, a* morphism $\delta \colon \alpha \to \alpha'$ *consists of*

- *an injective homomorphism $\delta \colon (X, P, C, \gamma) \to (X', P', C', \gamma')$ between the underlying hypergraphs of α and α' such that $\delta^e(P) \subseteq P'$ and $\delta^e(C) \subseteq C'$, and*
- *a family of polarity-preserving injections $\delta^{pt} = \left\{\delta^{pt}_x \colon M_x \to M'_{\delta^v(x)}\right\}_{x \in X}$,*

such that

- *for every non-requires-point $x \in X$, $\delta^{pt}_x = 1_{M_x}$,*
- *for every computation hyperedge $p \in P$, $\Lambda_p = \Lambda'_{\delta^e(p)}$, and*
- *for every communication hyperedge $c \in C$, $M_c = M'_{\delta^e(c)}$, $\Lambda_c = \Lambda'_{\delta^e(c)}$ and the following diagram commutes, for any point $x \in \gamma_c$.*

$$\begin{array}{ccc} M_c = M'_{\delta^e(c)} & \xrightarrow{\mu^c_x} & M_x \\ & \searrow^{(\mu')^{\delta^e(c)}_{\delta^v(x)}} & \downarrow{\scriptstyle \delta^{pt}_x} \\ & & M'_{\delta^v(x)} \end{array}$$

Proposition 1. *The morphisms of ARNs form a category, denoted $\mathbb{A}RN$, in which the composition is defined component-wise, with left and right identities given by morphisms whose components are set-theoretic identities.*

2.2 Sentences and Sentence Translations

We now define the sentence functor of our institution.

Definition 10 (Sentence). *For any ARN α, i.e. for any signature α, the set* $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\alpha)$ *of* (atomic) α-sentences *is defined as the set of pairs (x, ρ), usually denoted $@_x\,\rho$, where x is a point of α and ρ is an* MA-$\underline{\mathrm{LTL}}$*-sentence over A_{M_x}.*

The *translation* of sentences is straightforward: for every morphism $\delta\colon \alpha \to \alpha'$ of ARNs, the map $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\delta)\colon \mathsf{Sen}^{\underline{\mathrm{SOC}}}(\alpha) \to \mathsf{Sen}^{\underline{\mathrm{SOC}}}(\alpha')$ is given by

$$\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\delta)(@_x\,\rho) = @_{\delta^v(x)}\,\delta^{pt}_x(\rho)$$

for any point x of α and any MA-$\underline{\mathrm{LTL}}$-sentence ρ over the actions of x.

Proposition 2. $\mathsf{Sen}^{\underline{\mathrm{SOC}}}$ *is a functor $\mathbb{A}RN \to \mathbb{S}et$.*

2.3 Models and Model Reductions

The model functor of our institution assigns ground ARNs to the requires-points of the considered networks.

Definition 11 (Ground ARN). *An ARN is said to be* ground *if it has no requires-points. We denote by $\mathbb{GA}RN$ the full subcategory of $\mathbb{A}RN$ determined by ground ARNs.*

Definition 12 (Model). *For any ARN α, the category* $\mathsf{Mod}^{\underline{\mathrm{SOC}}}(\alpha)$ *of* α-models *or* α-interpretations *is the comma category $\alpha/\mathbb{GA}RN$.*

It follows that α-interpretations are morphisms of ARNs $\iota\colon \alpha \to \beta$ such that β is a ground network, which can also be seen as collections of ground ARNs that are designated to the requires-points of α. In order to explain this in more detail let us introduce the following notions of dependency and ARN defined by a point.

Definition 13 (Dependency). *Let x and y be points of an ARN α. x is said to be* dependent *on y if there exists a path from x to y that begins with a computation hyperedge, i.e. if there exists an alternating sequence $x\, e_1\, x_1 \cdots e_n\, y$ of (distinct) points and hyperedges such that $x \in \gamma_{e_1}$, $y \in \gamma_{e_n}$, $x_i \in \gamma_{e_i} \cap \gamma_{e_{i+1}}$ for any $1 \leq i < n$, and $e_1 \in P$.*

Definition 14 (ARN Defined by a Point). *The* ARN defined by a point *x of an ARN α is the full sub-ARN α_x of α determined by x and the points on which x is dependent.*

One can now see that any interpretation $\iota\colon \alpha \to \beta$ of an ARN α assigns to each requires-point x of α the ground sub-ARN $\beta_{\iota^v(x)}$ of β defined by $\iota^v(x)$.

Example 4. Based on the ground ARN depicted in Fig. 6 we can define an interpretation $\iota\colon \mathrm{JourneyPlanner} \to \mathrm{JourneyPlannerNet}$ that preserves all the labels, points and hyperedges of $\mathrm{JourneyPlanner}$, with the exception of the points R_1 and R_2, which are mapped to MS_1 and TS_1, respectively. In this case, the point MS_1 only depends on itself, hence the sub-ARN of $\mathrm{JourneyPlannerNet}$ defined by MS_1, i.e. the ground ARN assigned to the requires-point R_1 of $\mathrm{JourneyPlanner}$, is given by the process MS and its port MS_1. In contrast, the point JP_1 depends on all other points of $\mathrm{JourneyPlannerNet}$, and thus it defines the entire ARN $\mathrm{JourneyPlannerNet}$.

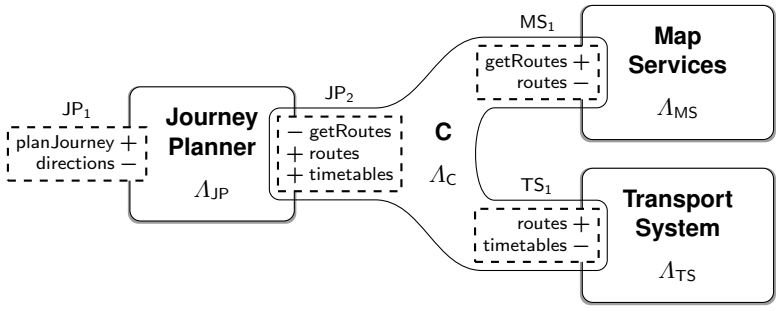


Fig. 6. The JourneyPlannerNet ARN

The reduction of interpretations is defined as the left composition with the considered ARN morphism. For every morphism of ARNs $\delta\colon \alpha \to \alpha'$, the reduct functor $\mathsf{Mod}^{\underline{\mathrm{SOC}}}(\delta)$ is just the composition functor $\delta/\mathbb{GA}RN\colon \alpha'/\mathbb{GA}RN \to \alpha/\mathbb{GA}RN$ given by $(\delta/\mathbb{GA}RN)(\iota') = \delta;\iota'$ and $(\delta/\mathbb{GA}RN)(\zeta') = \zeta'$ for every α'-interpretation ι' and every α'-interpretation homomorphism ζ'.

Proposition 3. $\mathsf{Mod}^{\underline{\mathrm{SOC}}}$ *is a contravariant functor* $\mathbb{A}RN^{\,op} \to \mathbb{C}at$.

2.4 The Satisfaction Relation

The evaluation of ARN sentences with respect to ARN interpretations relies on the concepts of diagram of a network and of automaton defined by a point, whose purpose is to describe the observable behaviour of a ground ARN through one of its points. We start by extending Facts 1 and 2 to ARNs.

Fact 3 (Diagram of an ARN). Every ARN $\alpha = (X, P, C, \gamma, M, \mu, \Lambda)$ defines a diagram $D_\alpha\colon \mathbb{J}_\alpha \to \mathbb{S}ig^{\text{MA-}\underline{\mathrm{LTL}}}$ as follows:

- $\mathbb{J}_\alpha$ is the free preordered category given by the set of objects

$$X \cup P \cup C \cup \{\langle c, x, \alpha\rangle \mid c \in C, x \in \gamma_c\}$$

  and the arrows
  - $\{x \to p \mid p \in P, x \in \gamma_p\}$ for computation hyperedges, and
  - $\{c \leftarrow \langle c, x, \alpha\rangle \to x \mid c \in C, x \in \gamma_c\}$ for communication hyperedges;
- D_α is the functor that provides the sets of actions of ports, processes and channels, together with the appropriate mappings between them. For example, given a communication hyperedge $c \in C$ and a point $x \in \gamma_c$,
  - $D_\alpha(c) = A_{M_c}$, $D_\alpha(\langle c, x, \alpha\rangle) = \mathrm{dom}(\mu^c_x)$, $D_\alpha(x) = A_{M_x}$,
  - $D_\alpha(\langle c, x, \alpha\rangle \to c) = \big(\mathrm{dom}(\mu^c_x) \subseteq A_{M_c}\big)$, and
  - $D_\alpha(\langle c, x, \alpha\rangle \to x) = \mu^c_x$.

Because the category $\mathbb{S}ig^{\text{MA-}\underline{\mathrm{LTL}}}$ is finitely co-complete, we can define the signature of an ARN based on its diagram.

Definition 15 (Signature of an ARN). *The signature of an ARN α is the co-limiting co-cone $\xi\colon D_\alpha \Rightarrow A_\alpha$ of the diagram D_α.*

The most important construction that allows us to define the satisfaction relation is the one that defines the observed behaviour of a (ground) network at one of its points.

Definition 16 (Automaton Defined by a Point). *Let x be a point of a ground ARN β. The* observed automaton *Λ_x at x is given by the reduct $\Lambda_{\beta_x}\restriction_{\xi_x}$, where*

- *$\beta_x = (X, P, C, \gamma, M, \mu, \Lambda)$ is the sub-ARN of β defined by x,*
- *$\xi\colon D_{\beta_x} \Rightarrow A_{\beta_x}$ is the signature of β_x,*
- *Λ_{β_x} is the product automaton $\prod_{e\,\in\, P\cup C} \Lambda_e^{\beta_x}$, and*
- *$\Lambda_e^{\beta_x}$ is the co-free expansion of Λ_e along ξ_e, for any hyperedge $e \in P \cup C$.*

Example 5. Let us consider the ground ARN outlined in Fig. 6. The automaton defined by the point MS_1 is just $\Lambda_{\mathsf{MS}}\restriction_{A_{\mathsf{MS}_1}}$ – this follows from the observation that the ARN defined by MS_1 consists solely of the process MS and the port MS_1. On the other hand, the calculation of the automaton defined by provides-point JP_1 involves the product of the co-free expansions of all four automata Λ_{JP}, Λ_{MS}, Λ_{TS} and Λ_{C}.

We now have all the necessary concepts for defining the satisfaction of ARN sentences by ARN interpretations. Let us thus consider an ARN α, an α-interpretation $\iota\colon \alpha \to \beta$ and an α-sentence $@_x\,\rho$. Then

$$\iota \models^{\underline{\mathrm{SOC}}}_{\alpha} @_x\,\rho \quad \text{if and only if} \quad \Lambda_{\iota^v(x)}\restriction_{\iota^{pt}_x} \models^{\text{MA-}\underline{\mathrm{LTL}}} \rho \ ,$$

where $\Lambda_{\iota^v(x)}$ is the observed automaton at $\iota^v(x)$ in β.

The construction of the institution of ARNs is completed by the following result, which states that satisfaction is invariant with respect to changes of ARNs.

Proposition 4. *For every ARN morphism $\delta\colon \alpha \to \alpha'$, any α'-interpretation ι' and any α-sentence $@_x\,\rho$,*

$$\iota' \models^{\underline{\mathrm{SOC}}}_{\alpha'} \mathsf{Sen}^{\underline{\mathrm{SOC}}}(\delta)(@_x\,\rho) \quad \text{if and only if} \quad \mathsf{Mod}^{\underline{\mathrm{SOC}}}(\delta)(\iota') \models^{\underline{\mathrm{SOC}}}_{\alpha} @_x\,\rho \ .$$

Corollary 1. $\underline{\mathrm{SOC}} = \big(\mathbb{A}RN, \mathsf{Sen}^{\underline{\mathrm{SOC}}}, \mathsf{Mod}^{\underline{\mathrm{SOC}}}, \models^{\underline{\mathrm{SOC}}}\big)$ *is an institution.*

3 A Logical View on Service Discovery and Binding

Building on the results of Sect. 2, we now investigate how the semantics of the service overlay can be characterised using fundamental computational aspects of the LP paradigm such as unification and resolution.

Our approach is built upon a simple and intuitive analogy between the SOC concepts of service module and client application [11] and the LP concepts of clause and query [16]. In order to clarify this analogy we rely on the institution $\underline{\mathrm{FOL}}$ of first-order logic [6] and also on the studies on internal logic developed in [19] and [5].

We begin by briefly describing the structure that provides the basic elements involved in defining the denotational and operational semantics of relational LP – the institution of (sets of) variables and substitutions over a first-order signature (S, F, P).

The signatures of this institution are finite sets (or blocks) of variables, i.e. finite sets of pairs (x, s), where x is the name of the variable (distinct from the names of

other variables) and $s \in S$ is its sort. The models, sentences, and the satisfaction relation are inherited from FOL. In this sense, for every set of variables X, we consider the corresponding category of models, set of sentences and satisfaction relation of the extended first-order signature $(S, F \cup X, P)$.

The morphisms of signatures $X \to Y$ are substitutions, i.e. mappings of the variables of X into terms over Y. Based on the evaluation of terms in models and on the canonical extension of substitutions from variables to terms, the substitutions define appropriate reductions of models and translations of sentences, about which it has been shown in [5] that the satisfaction condition holds.

3.1 The Clausal Structure of Services

Given the above constructions, we can describe definite first-order clauses as structures

$$C \xleftarrow[X]{} H$$

such that X is a block of variables, C is a relational atom over X, i.e. a relational atom of the extended signature $(S, F \cup X, P)$, and H is a finite set of relational atoms over X. Their semantics is given by the class of (S, F, P)-algebras whose expansions to $(S, F \cup X, P)$ satisfy C whenever they satisfy every sentence in H. Note that in traditional LP the symbols of variables are often distinguished from other symbols through notational conventions. For this reason, the block X of variables is at times omitted.

Service clauses can be defined in a similar manner, essentially by replacing the institution of first-order substitutions with the institution of ARNs. Intuitively, this means that we replace blocks of variables with ARNs, variables with requires-points, and terms (over variables) with provides-points.

Definition 17 (Clause). *A* SOC-clause *is a structure* (P, α, R)*, also written*

$$P \xleftarrow[\alpha]{} R$$

such that α *is an ARN,* P *is an* α*-sentence referring to a provides-point of* α *and* R *is a finite set of* α*-sentences referring to distinct requires-points of* α*.*

The semantics of service clauses is defined just as the semantics of first-order clauses, except that they are evaluated over the class $\mathcal{G}$ of ground ARNs instead of (S, F, P) ground terms. In this sense, $\mathcal{G}$ (which may be intuitively regarded as the Herbrand universe) satisfies a clause (P, α, R) if and only if any interpretation of α that satisfies all sentences in R satisfies P as well.

Example 6. The ARN JourneyPlanner introduced in Ex. 3 can orchestrate a service module that consistently delivers the requested directions, provided that the routes and the timetables can always be obtained. We specify this through the service clause

$$@_{\mathsf{JP}_1}\, \rho^{\mathrm{JP}} \xleftarrow[\mathrm{JourneyPlanner}]{} \left\{ @_{\mathsf{R}_1}\, \rho_1^{\mathrm{JP}}, @_{\mathsf{R}_2}\, \rho_2^{\mathrm{JP}} \right\}$$

where ρ^{JP}, ρ_1^{JP}, ρ_2^{JP} are the MA-LTL-sentences $\Box\,(\mathsf{planJourney}_{\mathsf{i}} \supset \Diamond\, \mathsf{directions}!)$, $\Box\,(\mathsf{getRoutes}_{\mathsf{i}} \supset \Diamond\, \mathsf{routes}!)$ and $\Box\,(\mathsf{routes}_{\mathsf{i}} \supset \Diamond\, \mathsf{timetables}!)$, respectively.

Client applications are captured in the present setting by service queries. They are defined in a similar manner to service clauses and their semantics are, as expected, existential rather than universal.

Definition 18 (Query). *A* SOC-query *is a structure* (α, Q)*, also denoted*

$$\underset{\alpha}{\vdash\!\!\!-} Q$$

such that α *is an ARN and* Q *is a finite set of* α*-sentences referring to distinct requires-points of* α*. With respect to semantics,* $\mathcal{G}$ *satisfies* (α, Q) *if and only if there exists an interpretation of* α *satisfying all sentences in* Q*.*

Example 7. Figure 7 outlines the ARN of a possible client for the Journey Planner service. We specify the actual client through the service query

$$\underset{\text{Client}}{\vdash\!\!\!-} \left\{ @_{\mathsf{R}_1}\, \rho_1^{\mathrm{C}} \right\}$$

given by the MA-LTL-sentence $\Box\,(\mathsf{getRoute}_{\mathit{i}} \supset \Diamond\, \mathsf{route}_{\mathit{i}}')$.

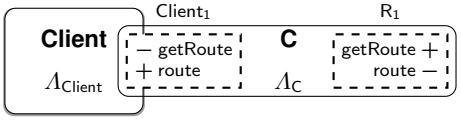


Fig. 7. The Client ARN

3.2 Resolution as Service Discovery and Binding

Service discovery represents, as in conventional LP, the search for a clause that could take the current goal one step closer to a possible solution. The solutions to service queries are defined in the same way as the solutions to first-order queries, but with ARN morphisms in the role of term substitutions.

Definition 19 (Solution). *A* solution *to* (α, Q) *consists of a morphism* $\theta\colon \alpha \to \alpha'$ *such that any interpretation of* α' *satisfies the* θ*-translation of any sentence in* Q*.*

The following result relates satisfiable queries and queries that have solutions, and may be regarded as a service-oriented correspondent of Herbrand's theorem.

Proposition 5. *A service query* (α, Q) *is satisfiable if and only if it admits a solution* $\theta\colon \alpha \to \alpha'$ *such that the category* $\mathsf{Mod}^{\mathrm{SOC}}(\alpha')$ *is not empty.*

The procedure that ultimately decides whether or not a service can be bound to an application is *unification*.

Definition 20 (Unifier). *Let* $@_{x_1}\, \rho_1$ *and* $@_{x_2}\, \rho_2$ *be two sentences of ARNs* α_1 *and* α_2*, respectively. A* unifier *of* $@_{x_1}\, \rho_1$ *and* $@_{x_2}\, \rho_2$ *consists of a pair* $\langle \theta_1, \theta_2 \rangle$ *of morphisms* $\theta_1\colon \alpha_1 \to \alpha$ *and* $\theta_2\colon \alpha_2 \to \alpha$ *such that* $\theta_1^v(x_1) = \theta_2^v(x_2)$ *and*

$$\theta^{pt}_{2,x_2}(\rho_2) \models^{\text{MA-LTL}} \theta^{pt}_{1,x_1}(\rho_1) \ .$$

In conventional LP the resolution process simplifies the current goal and at the same time, through unification, yields computed substitutions that could eventually deliver a solution to the initial query. This process is accurately reflected in the case of SOC by *service binding*.

Definition 21 (Resolution). *A service query* (α, Q) *is said to be derived by* resolution *from* (α_1, Q_1) *and* (P_2, α_2, R_2) *using the* computed morphism $\theta_1 \colon \alpha_1 \to \alpha$ *when*

$$\frac{\underset{\alpha_1}{\longmapsto} Q_1 \qquad P_2 \underset{\alpha_2}{\longleftarrow} R_2}{\underset{\alpha}{\longmapsto} \mathsf{Sen}^{\mathrm{SOC}}(\theta_1)(Q_1 \setminus \{R_1\}) \cup \mathsf{Sen}^{\mathrm{SOC}}(\theta_2)(R_2)} \ (\theta_1)$$

- *there exists a unifier* $\langle \theta_1, \theta_2 \rangle$ *of a sentence* $R_1 \in Q_1$ *and* P_2*, and*
- Q *is the set of sentences given by the translation along* θ_1 *and* θ_2 *of the sentences in* $Q_1 \setminus \{R_1\}$ *and* R_2*.*

Example 8. Let us consider the query and the clause detailed in the Ex. 7 and 6. One can easily see that the Client-sentence $@_{\mathsf{R}_1}\, \rho_1^{\mathrm{C}}$ and the JourneyPlanner-sentence $@_{\mathsf{JP}_1}\, \rho^{\mathrm{JP}}$ are unifiable. They admit the unifier $\langle \theta_1, \theta_2 \rangle$ given by

$$\mathrm{Client} \xrightarrow{\theta_1} \mathrm{Client} \| \mathrm{JourneyPlanner} \xleftarrow{\theta_2} \mathrm{JourneyPlanner}$$

- the ARN $\mathrm{Client} \| \mathrm{JourneyPlanner}$ depicted in Fig.8,
- the ARN morphism θ_1 that maps the point R_1 into JP_1, the communication hyperedge C into CJP and the messages $\mathsf{getRoute}$ and route of M_{R_1} into $\mathsf{planJourney}$ and $\mathsf{directions}$, respectively (while preserving all the remaining elements of Client),
- the ARN inclusion morphism θ_2.

It follows that we can derive by resolution a new service query, defined by the network $\mathrm{Client} \| \mathrm{JourneyPlanner}$ and the set of sentences $\left\{ @_{\mathsf{R}_1}\, \rho_1^{\mathrm{JP}}, @_{\mathsf{R}_2}\, \rho_2^{\mathrm{JP}} \right\}$.

$$\frac{\underset{\mathrm{Client}}{\longmapsto} \left\{ @_{\mathsf{R}_1}\, \rho_1^{\mathrm{C}} \right\} \qquad @_{\mathsf{JP}_1}\, \rho^{\mathrm{JP}} \underset{\mathrm{JourneyPlanner}}{\longleftarrow} \left\{ @_{\mathsf{R}_1}\, \rho_1^{\mathrm{JP}}, @_{\mathsf{R}_2}\, \rho_2^{\mathrm{JP}} \right\}}{\underset{\mathrm{Client} \| \mathrm{JourneyPlanner}}{\longmapsto} \left\{ @_{\mathsf{R}_1}\, \rho_1^{\mathrm{JP}}, @_{\mathsf{R}_2}\, \rho_2^{\mathrm{JP}} \right\}} \ (\theta_1)$$

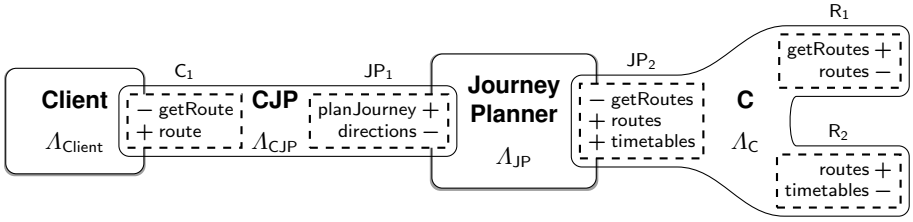


Fig. 8. The $\mathrm{Client} \| \mathrm{JourneyPlanner}$ ARN

Computed Solutions. The process of service discovery and binding allows us to search for solutions to arbitrary queries. The search is triggered by a query (α, Q) and consists in the iterated application of resolution until the derived service query is empty, i.e. a query of the form $(\alpha', \emptyset)$. Whenever the search procedure successfully terminates we obtain a *computed solution* of the original query by sequentially composing the resulting computed morphisms.

The correctness of the search procedure relies on the correctness of resolution.

Proposition 6. *Let (α, Q) be a service query derived by resolution from (α_1, Q_1) and (P_2, α_2, R_2) using the computed morphism θ_1. If (P_2, α_2, R_2) is satisfiable then for any solution θ of (α, Q), $\theta_1; \theta$ is a solution of (α_1, Q_1).*

It is easy to see that any empty query admits a trivial solution, namely the solution given by the identity morphism of its underlying ARN. By applying Prop. 6 backwards, for each resolution step, we deduce that the composition of any terminating sequence of computed morphisms, and thus any computed solution, is a solution.

4 Conclusions

In this paper, we showed how the integration of declarative and operational semantics as provided by Logic Programming can be generalised to Service-Oriented Computing to offer an integrated semantics for the static and dynamic aspects of this paradigm, i.e. to provide, for the first time, an algebraic framework that accounts for the mechanisms through which service interfaces can be orchestrated and for those that allow applications to discover and bind to services. The analogy that we established is based on the identification of the binding of terms to variables in LP with the binding of orchestrations of services to requires-points of software applications in SOC. The answer to a service query – the request for external services – is obtained through resolution using the service clauses (orchestrated service interfaces) available from a repository. This departs from other works on the logic-programming semantics of services such as [15] that considered implementations of the service discovery and binding mechanisms using constraint logic programming.

The analogy is grounded on a declarative semantics of service clauses defined over a novel institution whose models are asynchronous networks of Muller automata (service orchestrations) and whose sentences are linear temporal logic sentences expressing properties that can be observed at given interaction points of a network. Other logics could have been used instead of linear temporal logic, more specifically any institution such that (*a*) the category of signatures is (finitely) co-complete; (*b*) there exist co-free models along any signature morphism; (*c*) the category of models of any signature has products; (*d*) any model homomorphism reflects the satisfaction of any sentence.

These results encourage us to further develop a unifying framework for the foundations of LP that incorporates ideas from existing concrete variants of the phenomena, not necessarily restricted to relational or service-oriented programming. One possible course of action would be to isolate the principles of the LP paradigm in an institutional setting. This assumes institution-independent versions of LP concepts such as Herbrand model, clause, substitution, unifier and resolution, some of which have already been considered in the literature (e.g. [5]).

Acknowledgements. The work of the first author has been supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-II-ID-PCE-2011-3-0439. The authors also wish to thank Fernando Orejas for suggesting the use of hypergraphs and Antónia Lopes for many useful discussions that led to the present form of this paper.

References

1. G. Alonso, F. Casati, H. A. Kuno, and V. Machiraju. *Web Services – Concepts, Architectures and Applications*. Springer, 2004.
2. B. Benatallah, F. Casati, and F. Toumani. Representing, analysing and managing web service protocols. *Data Knowl. Eng.*, 58(3):327–357, 2006.
3. D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
4. R. Bruni, F. Gadducci, and A. Lluch-Lafuente. A graph syntax for processes and services. In C. Laneve and J. Su, editors, *WS-FM*, volume 6194 of *LNCS*, pages 46–60. Springer, 2009.
5. R. Diaconescu. Herbrand theorems in arbitrary institutions. *Inf. Process. Lett.*, 90(1):29–37, 2004.
6. R. Diaconescu. *Institution-Independent Model Theory*. Studies in Universal Logic. Birkhäuser, 2008.
7. G. L. Ferrari, D. Hirsch, I. Lanese, U. Montanari, and E. Tuosto. Synchronised hyperedge replacement as a model for service oriented computing. In F. de Boer, M. Bonsangue, S. Graf, and W. de Roever, editors, *FMCO*, volume 4111 of *LNCS*, pages 22–43. Springer, 2005.
8. J. L. Fiadeiro and A. Lopes. An interface theory for service-oriented design. In D. Giannakopoulou and F. Orejas, editors, *FASE*, volume 6603 of *LNCS*, pages 18–33. Springer, 2011.
9. J. L. Fiadeiro and A. Lopes. A model for dynamic reconfiguration in service-oriented architectures. *Software and Systems Modeling*, pages 1–19, 2012.
10. J. L. Fiadeiro, A. Lopes, and L. Bocchi. Algebraic semantics of service component modules. In J. L. Fiadeiro and P.-Y. Schobbens, editors, *WADT*, volume 4409 of *LNCS*, pages 37–55. Springer, 2006.
11. J. L. Fiadeiro, A. Lopes, and L. Bocchi. An abstract model of service discovery and binding. *Formal Asp. Comput.*, 23(4):433–463, 2011.
12. J. L. Fiadeiro and V. Schmitt. Structured co-spans: An algebra of interaction protocols. In T. Mossakowski, U. Montanari, and M. Haveraaen, editors, *CALCO*, volume 4624 of *LNCS*, pages 194–208. Springer, 2007.
13. I. T. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.
14. J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *J. ACM*, 39(1):95–146, 1992.
15. S. Kona, A. Bansal, and G. Gupta. Automatic composition of semantic web services. In *ICWS*, pages 150–158. IEEE Computer Society, 2007.
16. J. W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer, 1987.
17. D. Perrin and J. Éric Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Pure and Applied Mathematics. Elsevier Science, 2004.
18. J. Su, T. Bultan, X. Fu, and X. Zhao. Towards a theory of web service choreographies. In M. Dumas and R. Heckel, editors, *WS-FM*, volume 4937 of *LNCS*, pages 1–16. Springer, 2007.
19. A. Tarlecki. Quasi-varieties in abstract algebraic institutions. *J. Comput. Syst. Sci.*, 33(3):333–360, 1986.

A First-Order Logic

(Many-Sorted) First-Order Logic (FOL) was first presented as an institution in [14]. Here we only briefly recall a number of definitions, mainly for fixing the context in which traditional LP is discussed. More detailed presentations can be found in the recent works on institution theory such as [6].

Signatures. A *many-sorted first-order signature* (S, F, P) consists of

- a set S of *sorts*,
- a family $F = \{F_{w \to s} \mid w \in S^*, s \in S\}$ of sets of *operation symbols*, indexed by *arities* and *sorts*, and
- a family $P = \{P_w \mid w \in S^*\}$ of sets of *relation symbols*, indexed by *arities*.

A *signature morphism* $\varphi\colon (S, F, P) \to (S', F', P')$ is defined by

- a map $\varphi^{st}\colon S \to S'$ between the sets of sorts,
- a family of maps $\varphi^{op} = \big\{\varphi^{op}_{w \to s}\colon F_{w \to s} \to F'_{\varphi^{st}(w) \to \varphi^{st}(s)} \mid w \in S^*, s \in S\big\}$ between the sets of operation symbols, and
- a family of maps $\varphi^{rel} = \big\{\varphi^{rel}_w\colon P_w \to P'_{\varphi^{st}(w)} \mid w \in S^*\big\}$ between the sets of relation symbols.

Models. Given a first-order signature (S, F, P), a *model* M interprets

- each sort $s \in S$ as a set M_s,
- each operation symbol $\sigma \in F_{w \to s}$ as a function $M_\sigma\colon M_w \to M_s$, where M_w denotes the Cartesian product $M_{s_1} \times \cdots \times M_{s_n}$, for $w = s_1 \cdots s_n$, and
- each relation symbol $\pi \in P_w$ as a subset $M_\pi \subseteq M_w$.

For two (S, F, P)-models M and N, a *model homomorphism* $h\colon M \to N$ consists of a family of functions $\{h_s\colon M_s \to N_s \mid s \in S\}$ such that

- $h_s\big(M_\sigma(m)\big) = N_\sigma\big(h_w(m)\big)$ for each operation symbol $\sigma \in F_{w \to s}$ and each element $m \in M_w$, where $h_w\colon M_w \to N_w$ is the component-wise extension of h,
- $h_w(M_\pi) \subseteq N_\pi$ for each relation symbol $\pi \in P_w$.

With respect to *model reducts*, if $\varphi\colon (S, F, P) \to (S', F', P')$ is a signature morphism and M' is a (S', F', P')-model, then $M'{\restriction_\varphi}$ is defined as the (S, F, P)-model M given by $M_x = M'_{\varphi(x)}$, for every sort, operation symbol or relation symbol x of (S, F, P).

Sentences. The set of sentences of a signature (S, F, P) is the least set that contains

- the *equational atoms*, i.e. equalities $t = t'$ of *F-terms* $t = \sigma(t_1, \ldots, t_n)$ and $t' = \sigma'(t'_1, \ldots, t'_{n'})$, where σ and σ' are operation symbols of the same sort and $t_1, \ldots, t_n$ and $t'_1, \ldots, t'_{n'}$ are sub-terms,
- the *relational atoms*, i.e. expressions $\pi(t_1, \ldots, t_n)$, where π is a relation symbol and $t_1, \ldots, t_n$ are F-terms whose sorts correspond to the arity of π,

and is closed under Boolean connectives and quantification over first-order variables. For example, if X is a block of (S, F, P)-variables and ρ is an $(S, F \cup X, P)$-sentence then $(\forall X)\rho$ is a (S, F, P)-sentence.

The *translation of sentences* along a signature morphism $\varphi\colon (S,F,P) \to (S',F',P')$ is defined inductively on the structure of the (S,F,P)-sentences and naturally renames the sorts, operation symbols and relations symbols of (S,F,P) according to φ.

The Satisfaction Relation. The *satisfaction* between models and sentences is the usual Tarskian satisfaction defined inductively on the structure of the sentences. For instance, given a model M and a universal sentence $(\forall X)\rho$ of a signature (S,F,P), $M \models_{(S,F,P)} (\forall X)\rho$ if and only if $M' \models_{(S,F\cup X,P)} \rho$ for all $(S,F\cup X,P)$-expansions M' of M.

B Temporal Logic

The logical system that is used in the paper in defining service clauses, denoted MA-LTL, is a variation of Linear Temporal Logic whose semantics is based on Muller automata instead of conventional temporal models. We begin by recalling the institution of Linear Temporal Logic, which will later be used in defining the institution MA-LTL.

Linear Temporal Logic

Signatures. The *signatures* are arbitrary sets of *actions*. Signature morphisms are just functions, hence the category of signatures is $\mathbb{S}et$.

Models. Let A be a set of actions. An *A-model* is a *trace* of actions of A, i.e. an infinite sequence $\lambda \in \mathcal{P}(A)^\omega$. For every $i \in \omega$, we denote by $\lambda(i)$ the i-th element of λ and by λ^i the suffix of λ that starts at i. Given two traces λ and λ', a *trace homomorphism* $h\colon \lambda \to \lambda'$ consists of an infinite sequence of inclusions $\lambda(i) \subseteq \lambda'(i)$, for $i \in \omega$.
For any signature morphism $\sigma\colon A \to A'$ and any A'-trace λ', the *reduct trace* $\lambda = \lambda'{\restriction_\sigma}$ is defined in terms of its elements by $\lambda(i) = \sigma^{-1}\big(\lambda'(i)\big)$, for any $i \in \omega$.

Sentences. For any set of actions A, the set of A-sentences is defined as the least set that contains the constants $true$ and $false$, the actions in A, and is closed under negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$), implication ($\supset$), next ($\bigcirc$) and until ($\mathcal{U}$). The following well-known notations have been used extensively in the examples discussed in the paper: $\Diamond\,\rho$ stands for $true\ \mathcal{U}\ \rho$, and $\Box\,\rho$ stands for $\neg(true\ \mathcal{U}\ \neg\rho)$.
Given two sets of actions A and A', the translation of A-sentences along a signature morphism $\sigma\colon A \to A'$ is defined by induction on the structure of the sentences. For example, for any action $a \in A$, $\sigma(\neg a) = \neg\sigma(a)$, and for any sentences ρ_1 and ρ_2, $\sigma(\rho_1\ \mathcal{U}\ \rho_2) = \sigma(\rho_1)\ \mathcal{U}\ \sigma(\rho_2)$.

The Satisfaction Relation. For any set of actions A, the interpretation of A-sentences over traces $\lambda \in \mathcal{P}(A)^\omega$ is defined by

- $\lambda \models_A true$,
- $\lambda \not\models_A false$,
- $\lambda \models_A a$ if and only if $a \in \lambda(0)$,

- $\lambda \models_A \neg\rho$ if and only if $\lambda \not\models_A \rho$,
- $\lambda \models_A \rho_1 \vee \rho_2$ if and only if $\lambda \models_A \rho_1$ or $\lambda \models_A \rho_2$,
- $\lambda \models_A \rho_1 \wedge \rho_2$ if and only if $\lambda \models_A \rho_1$ and $\lambda \models_A \rho_2$,
- $\lambda \models_A \bigcirc \rho$ if and only if $\lambda^1 \models_A \rho$,
- $\lambda \models_A \rho_1 \mathcal{U} \rho_2$ if and only if there exists $i \in \omega$ such that $\lambda^i \models_A \rho_2$ and $\lambda^j \models_A \rho_1$ for all $j < i$,

where a is an action in A and ρ, ρ_1 and ρ_2 are A-sentences.

MA-LTL

The institution MA-LTL has the same category of signatures and the same sentence functor as LTL; therefore, we will only focus here on its model functor.

Models. For any set of actions A, the *models* are (non-deterministic) Muller automata over the alphabet $\mathcal{P}(A)$, i.e. tuples $\Lambda = \big(Q, \mathcal{P}(A), T, I, \mathcal{F}\big)$, where

- Q is a finite non-empty set of *states*,
- $T \subseteq Q \times \mathcal{P}(A) \times Q$ is the *transition relation*,
- $I \subseteq Q$ is the subset of *initial states*, and
- $\mathcal{F} \subseteq \mathcal{P}(Q)$ is the set of *final state sets*.

Given two automata $\Lambda = \big(Q, \mathcal{P}(A), T, I, \mathcal{F}\big)$ and $\Lambda' = \big(Q', \mathcal{P}(A), T', I', \mathcal{F}'\big)$ over $\mathcal{P}(A)$, a *homomorphism* $h\colon \Lambda \to \Lambda'$ consists of a function $h\colon Q \to Q'$ such that $h(I) \subseteq I'$, $\big(h(q_1), X, h(q_2)\big) \in T'$ whenever $(q_1, X, q_2) \in T$, and $h(\mathcal{F}) \subseteq \mathcal{F}'$.
Concerning the model reducts, for any map $\sigma\colon A \to A'$ and any Muller automaton $\Lambda' = \big(Q', \mathcal{P}(A'), T', I', \mathcal{F}'\big)$, the *reduct* $\Lambda' \restriction_\sigma = \big(Q', \mathcal{P}(A), T' \restriction_\sigma, I', \mathcal{F}'\big)$ is the automaton with the same states, initial states and final states as Λ', and with the transition relation given by

$$T' \restriction_\sigma = \big\{ \big(q_1, \sigma^{-1}(X'), q_2\big) \mid (q_1, X', q_2) \in T' \big\} \ .$$

The Satisfaction Relation. A *run* of an automaton $\Lambda = \big(Q, \mathcal{P}(A), T, I, \mathcal{F}\big)$ on a trace λ over A is an infinite sequence $r \in Q^\omega$ such that $\big(r(i), \lambda(i), r(i+1)\big) \in T$ for all $i \in \omega$. A run r is said to be *accepting* if $r(0) \in I$ and $\mathrm{Inf}(r) \in \mathcal{F}$, where $\mathrm{Inf}(r) \subseteq Q$ is the set of states that appear infinitely often in r.

A Muller automaton $\Lambda = \big(Q, \mathcal{P}(A), T, I, \mathcal{F}\big)$ accepts a trace λ if and only if it admits an accepting run on λ. Furthermore, Λ satisfies an LTL sentence ρ over A if and only if any trace λ accepted by Λ satisfies ρ.

$$\Lambda \models_A^{\text{MA-LTL}} \rho \quad \text{if and only if} \quad \lambda \models_A^{\text{LTL}} \rho \text{ for all } \lambda \text{ accepted by } \Lambda$$

C Proofs

Let us recall that the construction of the institution of ARNs presented in Sect. 2 relies on a base institution that satisfies the following properties:

- the category of signatures is finitely co-complete,

– there exist co-free models along any signature morphism,
– the category of models of any signature has products, and
– any model homomorphism reflects any sentence.

All these hypotheses hold for the institution MA-$\underline{\text{LTL}}$ as follows: the first is a well-known result about the existence of small co-limits in $\mathbb{S}et$, the second is provided by Prop. 7 below, the third is ensured by the closure of Muller automata under intersection, which is detailed categorically in Prop. 8, and the fourth is given by Prop. 9.

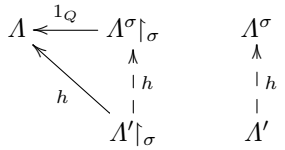
Proposition 7. *For any morphism of* MA-$\underline{\text{LTL}}$*-signatures* $\sigma\colon A \to A'$*, the model functor* $\_{\restriction_\sigma}$ *has a right adjoint, denoted* $(\_)^\sigma$.

Proof. According to a general result about adjoints, it suffices to show that for any automaton Λ over $\mathcal{P}(A)$ there exists an universal arrow from $\_{\restriction_\sigma}$ to Λ.

Let us consider an automaton $\Lambda = \big(Q, \mathcal{P}(A), T, I, \mathcal{F}\big)$ over $\mathcal{P}(A)$. We define the automaton $\Lambda^\sigma = \big(Q, \mathcal{P}(A'), T^\sigma, I, \mathcal{F}\big)$ over $\mathcal{P}(A')$ by

$$T^\sigma = \big\{(q_1, X', q_2) \mid \big(q_1, \sigma^{-1}(X'), q_2\big) \in T\big\} \ .$$

It is easy to see that the identity map 1_Q forms a homomorphism of automata $\Lambda^\sigma{\restriction_\sigma} \to \Lambda$: for any transition $(q_1, X, q_2) \in T^\sigma{\restriction_\sigma}$, by the definition of the reduct functor, we know there exists a set $X' \subseteq A'$ such that $\sigma^{-1}(X') = X$ and $(q_1, X', q_2) \in T^\sigma$; by the definition of T^σ it follows that $\big(q_1, \sigma^{-1}(X'), q_2\big) \in T$, hence $(q_1, X, q_2) \in T$.



Let us now assume that $h\colon \Lambda'{\restriction_\sigma} \to \Lambda$ is another homomorphism of automata, with $\Lambda' = (Q', \mathcal{P}(A'), T', I', \mathcal{F}')$. Then for any transition $(q_1, X', q_2) \in T'$, by the definition of the model functor, we have $\big(q_1, \sigma^{-1}(X'), q_2\big) \in T'{\restriction_\sigma}$. Since h is a homomorphism, it follows that $\big(h(q_1), \sigma^{-1}(X'), h(q_2)\big) \in T$, which further implies, by the definition of T^σ, that $\big(h(q_1), X', h(q_2)\big) \in T^\sigma$. As a result, the map h is also a homomorphism of automata $\Lambda' \to \Lambda^\sigma$. Even more, it is the unique homomorphism $\Lambda' \to \Lambda^\sigma$ (in the category of automata over $\mathcal{P}(A')$) such that $h; 1_Q = h$ in the category of automata over $\mathcal{P}(A)$. This concludes the proof. □

Proposition 8. *For any* MA-$\underline{\text{LTL}}$*-signature* A*, the category of Muller automata over* $\mathcal{P}(A)$ *has products.*

Proof. Let $\{\Lambda_i \mid i \in J\}$ be a family of Muller automata over $\mathcal{P}(A)$, with Λ_i given by $\big(Q_i, \mathcal{P}(A), T_i, I_i, \mathcal{F}_i\big)$, for $i \in J$. We define the automaton $\Lambda = \big(Q, \mathcal{P}(A), T, I, \mathcal{F}\big)$ by

– $Q = \prod_{i \in J} Q_i$,
– $T = \big\{(q_1, X, q_2) \mid \big(q_1(i), X, q_2(i)\big) \in T_i \text{ for all } i \in J\big\}$,
– $I = \prod_{i \in J} I_i$, and
– $\mathcal{F} = \{S \subseteq Q \mid \pi_i(S) \in \mathcal{F}_i \text{ for all } i \in J\}$,

where $\{\pi_i \colon Q \to Q_i\}_{i \in J}$ are the projections given by Cartesian product $\prod_{i \in J} Q_i$. It immediately follows from the construction of Λ that π_i is a homomorphism of automata $\Lambda \to \Lambda_i$, for any $i \in J$.

Let us now consider another family of homomorphisms $\{h_i \colon \Lambda' \to \Lambda_i\}_{i \in J}$, with $\Lambda' = (Q', \mathcal{P}(A'), T', I', \mathcal{F}')$. One can easily see that the map $h \colon Q' \to Q$ given by $h(q')(i) = h_i(q')$, for any $q' \in Q'$ and $i \in J$, defines a homomorphism $\Lambda' \to \Lambda$. For instance, given any transition $(q'_1, X, q'_2) \in T'$, it holds that $\big(h_i(q'_1), X, h_i(q'_2)\big) \in T_i$, for any $i \in J$ (because h_i is a homomorphism). Therefore $\big(h(q'_1), X, h(q'_2)\big) \in T$. In addition, by the properties of the product $\prod_{i \in J} Q_i$, h is the unique homomorphism satisfying $h; \pi_i = h_i$, for all $i \in J$. □

Proposition 9. *In* MA-$\underline{\text{LTL}}$ *every model homomorphism reflects any sentence, i.e. for any* MA-$\underline{\text{LTL}}$*-signature A, any A-model homomorphism $h \colon \Lambda \to \Lambda'$ and any A-sentence ρ, Λ satisfies ρ whenever Λ' satisfies ρ.*

Proof. Let us assume that $\Lambda = \big(Q, \mathcal{P}(A), T, I, \mathcal{F}\big)$ and $\Lambda' = \big(Q', \mathcal{P}(A), T', I', \mathcal{F}'\big)$. According to the definition of the satisfaction relation we need to prove that any trace λ accepted by Λ satisfies ρ.

We know that a trace λ is accepted by Λ if and only if there exists a run $r \in Q^\omega$ of Λ on λ such that $r(0) \in I$ and $\mathrm{Inf}(r) \in \mathcal{F}$. By the definition of the homomorphisms of automata, it follows that $r; h \in Q'^\omega$ is a run of the automaton Λ' on λ that verifies $(r; h)(0) = h(r(0)) \in I'$ and $\mathrm{Inf}(r; h) = h(\mathrm{Inf}(r)) \in \mathcal{F}'$. Therefore, the trace λ is also accepted by Λ', thus implying (by hypothesis), that λ satisfies ρ.

The remaining part of the paper is dedicated to the proofs of the properties presented in Sect. 2 and 3 that have been omitted due to lack of space.

Proof (of Prop. 4). Let $\delta \colon \alpha \to \alpha'$ be a morphism of ARNs, $\iota' \colon \alpha' \to \beta'$ an α'-interpretation and $@_x\, \rho$ an α-sentence. Assuming that $\Lambda'_{\iota'^v(\delta^v(x))} = \Lambda'_{(\delta;\iota')^v(x)}$ is the observed automaton at $\iota'^v(\delta^v(x))$ in β', we obtain the following equivalences.

$$
\begin{array}{lll}
\iota' \models^{\mathrm{SOC}}_{\alpha'} \mathsf{Sen}^{\mathrm{SOC}}(\delta)(@_x\, \rho) & & \\
\quad \text{iff} \quad \iota' \models^{\mathrm{SOC}}_{\alpha'} @_{\delta^v(x)}\, \delta^{pt}_x(\rho) & & \big(\text{by the definition of } \mathsf{Sen}^{\mathrm{SOC}}\big) \\
\quad \text{iff} \quad \Lambda'_{\iota'^v(\delta^v(x))}{\restriction}_{\iota'^{pt}_{\delta^v(x)}} \models^{\text{MA-}\underline{\text{LTL}}} \delta^{pt}_x(\rho) & & \big(\text{by the definition of } \models^{\mathrm{SOC}}\big) \\
\quad \text{iff} \quad \Lambda'_{\iota'^v(\delta^v(x))}{\restriction}_{\iota'^{pt}_{\delta^v(x)}}{\restriction}_{\delta^{pt}_x} \models^{\text{MA-}\underline{\text{LTL}}} \rho & & (\text{by the sat. cond. of MA-}\underline{\text{LTL}}) \\
\quad \text{iff} \quad \Lambda'_{\iota'^v(\delta^v(x))}{\restriction}_{\delta^{pt}_x;\iota'^{pt}_{\delta^v(x)}} \models^{\text{MA-}\underline{\text{LTL}}} \rho & & (\text{because } \mathsf{Mod}^{\text{MA-}\underline{\text{LTL}}} \text{ is a functor}) \\
\quad \text{iff} \quad \Lambda'_{(\delta;\iota')^v(x)}{\restriction}_{(\delta;\iota')^{pt}_x} \models^{\text{MA-}\underline{\text{LTL}}} \rho & & (\text{by the definition of ';' in } \mathbb{A}RN) \\
\quad \text{iff} \quad \delta; \iota' \models^{\mathrm{SOC}}_{\alpha} @_x\, \rho & & \big(\text{by the definition of } \models^{\mathrm{SOC}}\big) \\
\quad \text{iff} \quad \mathsf{Mod}^{\mathrm{SOC}}(\delta)(\iota') \models^{\mathrm{SOC}}_{\alpha} @_x\, \rho & & \big(\text{by the definition of } \mathsf{Mod}^{\mathrm{SOC}}\big)
\end{array}
$$

□

Lemma 1. *In* $\underline{\text{SOC}}$ *the satisfaction of any sentence is preserved along any model homomorphism, i.e. for any ARN α, any α-sentence $@_x\, \rho$ and any α-model homomorphism $\zeta \colon \iota_1 \to \iota_2$, if ι_1 satisfies $@_x\, \rho$ then ι_2 satisfies $@_x\, \rho$ as well.*

Proof. Let $\zeta\colon \iota_1 \to \iota_2$ be a homomorphism between two α-models $\iota_1\colon \alpha \to \beta_1$ and $\iota_2\colon \alpha \to \beta_2$, and let $@_x\,\rho$ be an α-sentence satisfied by ι_1. With respect to the notations, we assume that the sub-ARN of β_i determined by $\iota_i^v(x)$ (for $i \in \{1,2\}$) is the ground ARN $\beta_i^x = \big(X_i, P_i, C_i, \gamma^i, M^i, \mu^i, \Lambda^i\big)$, with the signature given by the co-limiting co-cone $\xi^i\colon D_{\beta_i^x} \Rightarrow A_{\beta_i^x}$.

By the definition of the satisfaction relation of $\underline{\text{SOC}}$ we know that

$$\iota_i \models^{\underline{\text{SOC}}}_{\alpha} @_x\,\rho \quad \text{if and only if} \quad \Lambda^i_{\iota_i^v(x)}{\restriction}_{\iota^{pt}_{i,x}} \models^{\text{MA-}\underline{\text{LTL}}} \rho \ ,$$

where $\Lambda^i_{\iota_i^v(x)}$ is the observed automaton at $\iota_i^v(x)$ in β_i. By hypothesis, we have that $\Lambda^1_{\iota_1^v(x)}{\restriction}_{\iota^{pt}_{1,x}} \models^{\text{MA-}\underline{\text{LTL}}} \rho$. Since we know that every MA-$\underline{\text{LTL}}$-model homomorphism reflects sentences (by Prop. 9), we deduce that $\Lambda^2_{\iota_2^v(x)}{\restriction}_{\iota^{pt}_{2,x}} \models^{\text{MA-}\underline{\text{LTL}}} \rho$ whenever there exists a homomorphism $\Lambda^2_{\iota_2^v(x)}{\restriction}_{\iota^{pt}_{2,x}} \to \Lambda^1_{\iota_1^v(x)}{\restriction}_{\iota^{pt}_{1,x}}$.

We know by hypothesis that ζ is a morphism between the ARNs β_1 and β_2 that satisfies $\iota_1;\zeta = \iota_2$. Moreover, because β_1 is ground, we infer that ζ^{pt}_z is an identity, for every point z of β_1, which implies that $\iota^{pt}_{2,y} = (\iota_1;\zeta)^{pt}_y = \iota^{pt}_{1,y};\zeta^{pt}_{\iota_1^v(y)} = \iota^{pt}_{1,y}$ for every point y of α. As a result, $\Lambda^2_{\iota_2^v(x)}{\restriction}_{\iota^{pt}_{2,x}} = \Lambda^2_{\iota_2^v(x)}{\restriction}_{\iota^{pt}_{1,x}}$. This means that in order to obtain a morphism $\Lambda^2_{\iota_2^v(x)}{\restriction}_{\iota^{pt}_{2,x}} \to \Lambda^1_{\iota_1^v(x)}{\restriction}_{\iota^{pt}_{1,x}}$, it suffices to obtain one $\Lambda^2_{\iota_2^v(x)} \to \Lambda^1_{\iota_1^v(x)}$.

Let us recall that $\Lambda^i_{\iota_i^v(x)}$ (for $i \in \{1,2\}$) is the reduct $\Lambda_{\beta_i^x}{\restriction}_{\xi^i_{\iota_i^v(x)}}$, where

- $\Lambda_{\beta_i^x}$ is the product $\prod_{e\,\in\, P_i \cup C_i} \Lambda_e^{\beta_i^x}$, with projections denoted $\pi^i_e\colon \Lambda_{\beta_i^x} \to \Lambda_e^{\beta_i^x}$, and
- $\Lambda_e^{\beta_i^x}$ (for $e \in P_i \cup C_i$) is the co-free expansion of Λ^i_e along ξ^i_e, with the universal morphism denoted $\varepsilon^i_e\colon \Lambda_e^{\beta_i^x}{\restriction}_{\xi^i_e} \to \Lambda^i_e$.

By the description of ARNs defined by a point we have that ζ can be restricted to a morphism of ARNs $\beta_1^x \to \beta_2^x$. Since β_1^x is ground, it follows that

- $D_{\beta_1^x}(z) = D_{\beta_2^x}(\zeta^v(z))$ for every point $z \in X_1$,
- $D_{\beta_1^x}(e) = D_{\beta_2^x}(\zeta^e(e))$ for every hyperedge $e \in P_1 \cup C_1$, and
- $D_{\beta_1^x}(\langle c, z, \beta_1^x\rangle) = D_{\beta_2^x}(\langle \zeta^e(c), \zeta^v(z), \beta_2^x\rangle)$ for every $c \in C_1$ and $z \in \gamma^1_c$.

This allows us to define a co-cone $\xi\colon D_{\beta_1^x} \Rightarrow A_{\beta_2^x}$ by

- $\xi_z = \xi^2_{\zeta^v(z)}$ for $z \in X_1$,
- $\xi_e = \xi^2_{\zeta^e(e)}$ for $e \in P_1 \cup C_1$, and
- $\xi_{\langle c,z,\beta_1^x\rangle} = \xi^2_{\langle \zeta^e(c),\zeta^v(z),\beta_2^x\rangle}$ for $c \in C_1$ and $z \in \gamma^1_c$.

Since ξ^1 is the co-limit of $D_{\beta_1^x}$ it follows that there exists a (unique) morphism of co-cones $\sigma\colon \xi^1 \to \xi$, i.e. an MA-$\underline{\text{LTL}}$-signature morphism $\sigma\colon A_{\beta_1^x} \to A_{\beta_2^x}$ that satisfies, in particular, $\xi^1_e;\sigma = \xi_e$ for every hyperedge $e \in P_1 \cup C_1$.

We obtain in this way, for every hyperedge $e \in P_1 \cup C_1$, the composite morphism $\pi^2_{\zeta(e)}{\restriction}_{\xi^2_{\zeta(e)}}; \varepsilon^2_{\zeta(e)}$ from $\Lambda_{\beta^x_2}{\restriction}_{\xi^2_{\zeta(e)}} = \Lambda_{\beta^x_2}{\restriction}_{\xi_e} = \Lambda_{\beta^x_2}{\restriction}_{\sigma}{\restriction}_{\xi^1_e}$ to $\Lambda^1_e = \Lambda^2_{\zeta(e)}$.

$$\begin{array}{ccccc}
\Lambda^1_e = \Lambda^2_{\zeta(e)} & \xleftarrow{\ \varepsilon^1_e\ } & \Lambda^{\beta^x_1}_e{\restriction}_{\xi^1_e} & & \Lambda^{\beta^x_1}_e \\
\uparrow{\scriptstyle \varepsilon^2_{\zeta(e)}} & & \uparrow{\scriptstyle h_e{\restriction}_{\xi^1_e}} & & \uparrow{\scriptstyle h_e} \\
\Lambda^{\beta^x_2}_{\zeta(e)}{\restriction}_{\xi^2_{\zeta(e)}} & \xleftarrow[\ \pi^2_{\zeta(e)}{\restriction}_{\xi^2_{\zeta(e)}}\ ]{} & \Lambda_{\beta^x_2}{\restriction}_{\xi^2_{\zeta(e)}} = \Lambda_{\beta^x_2}{\restriction}_{\sigma}{\restriction}_{\xi^1_e} & & \Lambda_{\beta^x_2}{\restriction}_{\sigma}
\end{array}$$

Since $\Lambda^{\beta^x_1}_e$ is the co-free expansion of Λ^1_e along ξ^1_e, we deduce that there exists a (unique) morphism $h_e \colon \Lambda_{\beta^x_2}{\restriction}_{\sigma} \to \Lambda^{\beta^x_1}_e$ such that the above diagram is commutative. This implies, by the universality property of the product $\Lambda_{\beta^x_1}$, the existence of a (unique) morphism $h \colon \Lambda_{\beta^x_2}{\restriction}_{\sigma} \to \Lambda_{\beta^x_1}$ such that $h; \pi^1_e = h_e$ for every $e \in P_1 \cup C_1$.

$$\begin{array}{ccc}
\Lambda^1_e & \xleftarrow{\ \pi^1_e\ } & \Lambda_{\beta^x_1} \\
 & \nwarrow^{h_e} & \uparrow{\scriptstyle h} \\
 & & \Lambda_{\beta^x_2}{\restriction}_{\sigma}
\end{array}$$

It follows that the reduct $h{\restriction}_{\xi^1_{\iota^v_1(x)}}$ is a morphism $\Lambda_{\beta^x_2}{\restriction}_{\sigma}{\restriction}_{\xi^1_{\iota^v_1(x)}} \to \Lambda_{\beta^x_1}{\restriction}_{\xi^1_{\iota^v_1(x)}}$. The argument is completed by noticing that

- $\Lambda_{\beta^x_2}{\restriction}_{\sigma}{\restriction}_{\xi^1_{\iota^v_1(x)}} = \Lambda_{\beta^x_2}{\restriction}_{\xi_{\iota^v_1(x)}} = \Lambda_{\beta^x_2}{\restriction}_{\xi^2_{\zeta^v(\iota^v_1(x))}} = \Lambda_{\beta^x_2}{\restriction}_{\xi^2_{\iota^v_2(x)}} = \Lambda^2_{\iota^v_2(x)}$ and
- $\Lambda_{\beta^x_1}{\restriction}_{\xi^1_{\iota^v_1(x)}} = \Lambda^1_{\iota^v_1(x)}$.

□

Proof (of Prop. 5). Let us first focus on the direct implication and assume that (α, Q) is a satisfiable query. According to the semantics of service queries it follows that there exists an α-interpretation $\iota \colon \alpha \to \beta$ that satisfies every sentence in Q. We argue that ι is a solution for (α, Q). In order to check this property we consider an arbitrary β-interpretation $\kappa \colon \beta \to \beta'$ and an α-sentence $@_x\, \rho \in Q$. Since κ defines an α-model homomorphism $\iota \to \iota; \kappa$ and ι satisfies $@_x\, \rho$, we deduce by Lemma 1 that $\iota; \kappa$ satisfies $@_x\, \rho$ as well. But $\iota; \kappa$ is just the reduct $\mathsf{Mod}^{\mathrm{SOC}}(\iota)(\kappa)$; therefore, by Prop. 4, it follows that κ satisfies $\mathsf{Sen}^{\mathrm{SOC}}(\iota)(@_x\, \rho)$. We conclude the first part of the proof by noticing that β has at least one interpretation, namely the identity 1_β.

For the reverse implication let us consider that $\theta \colon \alpha \to \alpha'$ is a solution of the query (α, Q) such that $\mathsf{Mod}^{\mathrm{SOC}}(\alpha')$ is not empty. We fix an interpretation $\iota' \colon \alpha' \to \beta'$ (which exists by hypothesis). Since θ is a solution of (α, Q) it follows that ι' satisfies $\mathsf{Sen}^{\mathrm{SOC}}(\theta)(@_x\, \rho)$, for every sentence $@_x\, \rho \in Q$. As a result, by Prop. 4, the α-interpretation $\mathsf{Mod}^{\mathrm{SOC}}(\theta)(\iota')$ satisfies every sentence $@_x\, \rho \in Q$. □

Lemma 2. *If $\langle \theta_1, \theta_2 \rangle$ is a unifier of an α_1-sentence $@_{x_1}\, \rho_1$ and an α_2-sentence $@_{x_2}\, \rho_2$, with $\theta_1 \colon \alpha_1 \to \alpha$ and $\theta_2 \colon \alpha_2 \to \alpha$, then*

$$\mathsf{Sen}^{\mathrm{SOC}}(\theta_2)(@_{x_2}\, \rho_2) \models^{\mathrm{SOC}}_{\alpha} \mathsf{Sen}^{\mathrm{SOC}}(\theta_1)(@_{x_1}\, \rho_1) \ .$$

Proof. Let $\iota\colon \alpha \to \beta$ be an α-interpretation that satisfies $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_2)(@_{x_2}\,\rho_2)$. Assuming that $\Lambda_{\iota^v(\theta_2^v(x_2))}$ is the observed automaton at $\iota^v(\theta_2^v(x_2))$ in β, we obtain

$$
\begin{aligned}
&\iota \models^{\underline{\mathrm{SOC}}}_{\alpha} \mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_2)(@_{x_2}\,\rho_2)\\
&\quad \text{iff} \quad \iota \models^{\underline{\mathrm{SOC}}}_{\alpha} @_{\theta_2^v(x_2)}\,\theta^{pt}_{2,x_2}(\rho_2) && \big(\text{by the definition of } \mathsf{Sen}^{\underline{\mathrm{SOC}}}\big)\\
&\quad \text{iff} \quad \Lambda_{\iota^v(\theta_2^v(x_2))}{\restriction}_{\iota^{pt}_{\theta_2^v(x_2)}} \models^{\text{MA-}\underline{\mathrm{LTL}}} \theta^{pt}_{2,x_2}(\rho_2) && \big(\text{by the definition of } \models^{\underline{\mathrm{SOC}}}\big)\\
&\quad \text{iff} \quad \Lambda_{\iota^v(\theta_1^v(x_1))}{\restriction}_{\iota^{pt}_{\theta_1^v(x_1)}} \models^{\text{MA-}\underline{\mathrm{LTL}}} \theta^{pt}_{2,x_2}(\rho_2) && \big(\text{because } \theta_1^v(x_1) = \theta_2^v(x_2)\big)\ .
\end{aligned}
$$

Since by hypothesis $\theta^{pt}_{2,x_2}(\rho_2) \models^{\text{MA-}\underline{\mathrm{LTL}}} \theta^{pt}_{1,x_1}(\rho_1)$, we deduce that the following relations hold as well, thus completing the proof.

$$
\begin{aligned}
&\Lambda_{\iota^v(\theta_1^v(x_1))}{\restriction}_{\iota^{pt}_{\theta_1^v(x_1)}} \models^{\text{MA-}\underline{\mathrm{LTL}}} \theta^{pt}_{1,x_1}(\rho_1)\\
&\quad \text{iff} \quad \iota \models^{\underline{\mathrm{SOC}}}_{\alpha} @_{\theta_1^v(x_1)}\,\theta^{pt}_{1,x_1}(\rho_1) && \big(\text{by the definition of } \models^{\underline{\mathrm{SOC}}}\big)\\
&\quad \text{iff} \quad \iota \models^{\underline{\mathrm{SOC}}}_{\alpha} \mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_1)(@_{x_1}\,\rho_1) && \big(\text{by the definition of } \mathsf{Sen}^{\underline{\mathrm{SOC}}}\big)
\end{aligned}
$$

□

Proof (of Prop. 6). Let $\theta\colon \alpha \to \alpha'$ be a solution of the query (α, Q) and $\iota'\colon \alpha' \to \beta'$ an interpretation of α'. We prove that ι' satisfies $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_1;\theta)(@_x\,\rho)$, for any sentence $@_x\,\rho \in Q_1$, by case analysis.

If the sentence $@_x\,\rho$ is distinct from R_1 then by the construction of Q we have that $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_1)(@_x\,\rho) \in Q$, which further implies, because θ is a solution of (α, Q), that ι' satisfies $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta)\big(\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_1)(@_x\,\rho)\big) = \mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_1;\theta)(@_x\,\rho)$.

Let us now assume that $@_x\,\rho$ is the sentence R_1. Since the morphism θ is a solution of (α, Q) and $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_2)(R_2) \subseteq Q$, it follows that ι' satisfies every sentence in $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta)\big(\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_2)(R_2)\big) = \mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_2;\theta)(R_2)$. Therefore, by Prop. 4, the interpretation $\mathsf{Mod}^{\underline{\mathrm{SOC}}}(\theta_2;\theta)(\iota')$ satisfies R_2. By the satisfiability of $\big(P_2, \alpha_2, R_2\big)$, we further deduce that $\mathsf{Mod}^{\underline{\mathrm{SOC}}}(\theta_2;\theta)(\iota')$ also satisfies P_2, which is equivalent with the fact that $\mathsf{Mod}^{\underline{\mathrm{SOC}}}(\theta)(\iota')$ satisfies $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_2)(P_2)$. Furthermore, since R_1 and P_2 are unifiable (by hypothesis), it follows by Lemma 2 that $\mathsf{Mod}^{\underline{\mathrm{SOC}}}(\theta)(\iota')$ satisfies $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_1)(R_1)$ as well. Hence, by applying Prop. 4 one last time, we conclude that ι' satisfies the sentence $\mathsf{Sen}^{\underline{\mathrm{SOC}}}(\theta_1;\theta)(R_1)$. □