WEB SERVICE COMPOSITION AS A COMPOSITION OF VALID
AND ROBUST SEMANTIC LINKS

FREDDY LÉCUÉ⁎1,2, ALEXANDRE DELTEIL1, ALAIN LÉGER1
1France Telecom R&D
4 rue du clos courtel, F-35512 Cesson Sévigné, France
{freddy.lecue, alexandre.delteil, alain.leger}@orange-ftgroup.com

OLIVIER BOISSIER2
2École Nationale Supérieure des Mines de St-Etienne
158, cours Fauriel, F-42023 Saint-Étienne, France
freddy.lecue@gmail.com, olivier.boissier@ense.fr

Received (Day Month Year)
Revised (Day Month Year)

Automated composition of Web services or the process of forming new value added Web services is one of the most promising challenges facing the Semantic Web today. Semantics enables Web service to describe capabilities together with their processes, hence one of the key elements for the automated composition of Web services. In this paper we focus on the functional level of Web services i.e., services are described according to some input, output parameters semantically enhanced by concepts in a domain ontology. Web service composition is then viewed as a composition of semantic links wherein the latter links refer to semantic matchmaking between Web service parameters (i.e., outputs and inputs) in order to model their connection and interaction. The key idea is that the matchmaking enables, at run time, finding semantic compatibilities among independently defined Web service descriptions. By considering such a level of composition a formal model to perform the automated composition of Web services i.e., Semantic Link Matrix is introduced. The latter model is required as a starting point to apply problem-solving techniques such as regression (or progression)-based search for Web service composition. The model supports a semantic context in order to find correct, complete, consistent and robust plans as solutions. In this paper an innovative and formal model for an AI (Artificial Intelligence) planning-oriented composition is presented. Our system is implemented and interacting with Web services dedicated on Telecom scenarios. The preliminary evaluation results showed high efficiency and effectiveness of the proposed approach.

Keywords: Web service, composition, semantic Web, automated reasoning, AI planning.

⁎corresponding author
1. Introduction

Service oriented computing (SOC) is an emerging cross-disciplinary paradigm for distributed computing that is changing the way software applications are designed, architected, delivered and consumed. Services are autonomous, platform-independent computational elements that can be described, published, discovered, orchestrated and programmed using standard protocols to build networks of collaborating applications distributed within and across organizational boundaries. Web Services are the current most promising technology to provide the feature richness, flexibility and scalability needed by enterprises to manage the Service oriented architecture (SOA) and SOC challenges. Therefore Web services provide the basis for the development and execution of business processes that are distributed over the network and available via standard interfaces and protocols.

The commonly accepted and minimal framework for services, referred to as SOA, consists of the following basic roles: i) the service provider, which is the subject (e.g., an organization) providing services; ii) the service directory, which is the subject providing a repository/registry of service descriptions, where providers publish their services and requestors find services; and iii) the service requestor, also referred to as client, which is the subject looking for and invoking the service in order to fulfill some goals. A requestor discovers a suitable service in the directory, and then connects to the specific service provider in order to invoke the service.

Research on Web services spans over many interesting issues. In this paper, we are particularly interested in semantic Web services and more particularly in automated composition of semantic Web services. The semantic Web i.e., the Web of meaning is considered as the new vision and extension of the current Web that tries to give semantic to the Web resources. Semantic Web aims at improving the technology to organise, search, integrate, and evolve Web-accessible resources (e.g., Web documents, data) by using rich and machine-understandable abstractions for the representation of resources semantics. Ontologies are proposed as means to address semantic heterogeneity among Web-accessible information sources and services. They are used to provide meta-data for the effective manipulation of available information including discovering information sources and reasoning about their capabilities. In the context of Web services, ontologies promise to take interoperability a step further by providing rich descriptions and modelling of service properties, capabilities, and behaviour. Web services in the semantic Web are enhanced using rich description languages (through Description Logics DLs) such as the Web Ontology Language (OWL). In this way semantic Web services are Web services that have been enhanced with formal semantic descriptions where OWL-S, the Web Service Modelling Ontology (WSMO), Semantic Annotation for WSDL (SA-WSDL) or the Semantic Web Services Language (SWSL) may be used to describe them. Services requestor (e.g., other services) can, in turn, use these descriptions to rea-

\[http://www.daml.org/services/swsl/\]
son about Web services and automate their use to accomplish goals specified by end-users including intelligent and automated discovery, selection, composition and invocation. In particular Web service composition enhanced by semantic technologies is currently one of the most hyped and addressed issue in the two major trends in Web technologies i.e., Web services and semantic Web. Web service composition is the ability to combine existing services together in order to provide new functionalities. Starting from an initial set of services, Web service composition aims at selecting and inter-connecting services provided by different partners according to a goal to achieve.

Most of the work in semantic Web service composition has focused on two main levels of composition: functional and process levels. The former level considers Web services as “atomic” components or black boxes described in terms of their inputs, outputs, preconditions and effects (IOPEs) where the execution follows a simple request-response step. Such a level of composition focuses on a correct (partial) ordering of Web services depending on a given criteria (e.g., semantic links, causal laws between Web services). The latter level supposes Web services as stateful processes with a complex interaction protocol involving more complex conversations. In such a level of composition, behaviour of Web service as State Transition Systems are composed, by controlling their Cartesian product. Process level composition investigates the problem of synthesizing a complex behaviour from an explicit goal and a set of candidate services which contribute to a partial solution of the complex problem of Web service composition.

The main focus in this paper is on semantic Web service composition at functional level. In order to address this issue in an effective and well-founded way, first of all we conduct a study of composition methods based on eight specific technical requirements (i.e., $R_1$: Service Expressivity and Category, $R_2$: Composition Expressivity, $R_3$: Composition Outcome, $R_4$: Composability Criteria, $R_5$: Assumptions, $R_6$: Flexibility and Dynamicy, $R_7$: Optimization Criteria, $R_8$: Semantic Reasoning). These requirements need to be met by composers to make automated web service composition a success. Towards such requirements our approach first focuses on a general formal model for representing semantic connections between Web services (i.e., Requirement $R_4$) since the latter connections are considered as the main components to provide suitable composition of Web services at functional level. Indeed they ensure sequence composability of Web services by means of their functional parameters. In this direction our first contribution is the introduction of semantic link. Works published in the literature do not address the issue of Web service composition as a composition of semantic links. By means of such links Web services can be semantically well-chained according to a semantic criteria. This criteria is depending on a quality of matchmaking between service parameters. In this

\begin{footnote}{In this paper the term Semantic Link refers to the Causal Link term, first introduced by in the research domain of Web services. Their semantics are the same, only the terminology has changed from the previous work.}

\end{footnote}
way Web service composition is reduced to a partial order of basic but semantic connections.

In addition to a clear definition of what a semantic link is, our framework provides a formal setting for a precise characterization of the new problems of validity and robustness of semantic links in Web service composition. The latter issues are considered as essential to provide robust and valid composition of Web services. However validity and robustness of semantic links in Web service composition are not addressed in this research area to the best of our knowledge. Once the problem of robustness introduced the second contribution presents two different approaches to overcome this problem in semantic Web service composition.

The third contribution of the paper is an effective technique for automated semantic Web service composition i.e., an automated process of chaining Web services according to their functional description, namely Ra4C (Regression-based Approach for Composition). In particular, we specialize the general framework to the case where Web services are specified by means of input and output parameters together with preconditions and effects, and we present the Semantic link matrix (SLM from now) as an innovative and formal model to apply problem-solving techniques such as regression (or progression)-based search for Web service composition. The SLM aims at not only storing all relevant Web services in a semantic way but also pre-chaining Web services by computing of semantic links. The suggested model supports a semantic context in order to find correct, complete and consistent plans as solutions of the AI planning-oriented composition.

As a forth contribution to the paper, we present the prototype design, architecture and development of an open source software tool implementing our composition technique Ra4C*. Our system is implemented and interacting with Web services dedicated on real Telecom scenarios. Practical experimentation conducted over the latter Telecom scenarios with the prototype showed high efficiency and effectiveness of the tool to build composite semantic Web services, despite the inherent exponential computational complexity of service composition, depending on the number of Web services involved in the composition process (number of Web services which is bounded in our real scenario). The prototype tool that we present in this paper shows exactly the feasibility and effectiveness of our algorithm.

The rest of the paper is organized as follows. In Section 2 we give some background in area of Web service composition, in particular Functional Level Composition, and consider related research work. Section 3 presents one of Telecom scenarios where we integrate the suggested approach. In Section 4 we focus on the notion of semantic links together with their validity and robustness features in Web service composition. Web service composition as a composition of semantic links is

\[ \text{cf. the SME3-Pro (ScMantic wEb sErviE PROject) Open Source Project licensed under the GPL license available at https://sws-orange labs.elibel.tm.fr/} \]
also introduced in this section. Section 5 presents the SLM as a formal model to describe semantic Web services at functional level. In Section 6, an AI planning-oriented method is presented to solve a (robust or not) Web service composition, given a specific SLM. In Section 7, we present our prototype tool together with the preliminary evaluation results based first on the motivating example, and second on random scenarios. Finally Section 8 draws some conclusions and talk about possible future directions.

2. Background, Motivation & Related Work

Recently, automated Web service composition has attracted great interest in the research community. As previously said literature approaches can be classified into two main and complementary categories\textsuperscript{19}: Functional Level Composition and Process Level Composition (FLC and PLC from now). However it is obvious that compositions of Web services that do not expose their behaviours (i.e., stateless Web services) can be achieved by leaving out the PLC step and adopting only FLC.

2.1. Background and Motivation on Functional Level Composition

Due to the high level of abstraction considered, the result of a Web service discovery process may actually be inadequate, and even unusable without a composition approach. For instance some discovery processes e.g.,\textsuperscript{20} retrieved a unordered list of Web services wherein some of them required to be composed to make it really useful for end-users. However first, the latter list may include services useless to satisfy the user request $Q$ e.g., some retrieved Web services may be not required by potential final compositions. Second, some of the discovered services may turn out not to be usable, since, in order to work, they require some specific preconditions to hold and these may not be guaranteed. Therefore some refinement is required to make the list of discovered services easy to use for end-user.

For these reasons, we need to recur to a capability-based composition in order to check that the discovered services really cover, at a functional level, the intended end-user request. In a complementary task FLC is responsible of pruning away useless Web service retrieved by the Web service discovery component. Compositions based on such a functional view of services are known as functional level compositions or data flow driven compositions\textsuperscript{21,8,13,22,14}. In the following we will invest on a novel FLC technique that allows partial matches (aka semantic similarities) of input and output parameters as concepts in a domain ontology, postulating at least that the intersection of the involved types is satisfiable. The search involved in FLC is sensibly more complex than the one performed during discovery, since FLC requires considering the interaction between the conditions enabled by the application of a service and those required by the services executed later on. In the following by FLC we mean a problem of selecting a set of Web services that, combined in a suitable way, are able to perform a composition goal i.e., an abstract service described by its IOPEs. Since Web service discovery and FLC are very close...
processes, the latter tasks are usually combined together to find a chain of suitable services. The composition goal (end-user request in our context) defines the overall functionality that the composed service should implement, again in terms of its IOPEs. Such description of services and goals can be provided for instance by means of some standard proposals such as the OWL-S (formerly DAML-S) service profile, WSMO service capability model, or SA-WSDL specification (formerly WSDL-S). By considering such a level of description, FLC is very much related to traditional AI planning. The available Web services correspond to planning operators that require certain input parameters and preconditions and provide some output parameters and effects. The planning results is a partial ordering of Web services arranged in a simpler version of a workflow in order to fulfil the composition goal. The latter ordering is composed of Web services together with semantic dependences i.e., semantic links between some of them. At functional level semantic connections between Web services are considered as the main issue to form new value-added services. These connections are required to semantically link output to input parameters of Web services. Such connections enable to model a composition of Web services as a simple sequence.

2.2. Related Work
Since our main contribution is about FLC, we did not focus deeply on PLC. For further information about PLC the reader can invest in the following relevant works. In the following we give an overview of the related FLC methods.

2.2.1. A Semi-Automated Composition Approach
FLC of Web services has been discussed in. Their authors provides a composition tool that supports the end user to compose Web services. To this end the end user is required at each service selection step of the composition process. The end user selects her best Web services (depending on her criteria) for each abstract activity in the composition. Upon selecting a Web service, the Web services that can produce output that could be fed as the input of the selected service are listed after filtering based on profile descriptions. The user can manually select the service that she wants to fit in at a particular activity. After selecting all the services, the system generates a composite process in DAML-S. The final flow specification to semantically link Web services is then created in a semi automated way. The execution is done by calling each service separately and passing the results between services according to the flow specifications.

Such an approach does not consider automated composition of Web service but obviously presents a semi automated method since the user is required to select a Web service in a restricted list. Our model of composition presented in this paper aims at automating the process of Web service selection by means of a semantic connections (and their valuation) between their functional parameters (i.e., technically inputs and outputs).
2.2.2. A Logic Based Approach

Towards this issue, \textsuperscript{25,26} introduces a method for automatic composition of semantic web services using Linear Logic theorem proving. On the one hand, Linear Logic, as a resource conscious logic, is used to formally define static definition of web services i.e., functional and non functional parameters (e.g., qualitative and quantitative values) of web services. On the other hand, Linear Logic, as a logic for specifying concurrent programming, has close relationship \textsuperscript{27,28} with $\pi$-calculus (i.e., the formal foundation of many Web service composition languages). Such a logic provides higher expressive powers in the modelling of dynamic behaviour of composite web services than classical logic. Their method uses DAML-S as a semantic web service language to describe them. From the DAML-S service description they retrieve Linear Logic extra logical axioms for each service. These axioms describe the functional description of services in terms of their input and output parameters. Besides the latter axioms, \textsuperscript{25,26} extends their initial Linear Logic theory by introducing additional axioms related to semantic similarity between output and input parameters of services. To this end they exploit a semantic reasoner to detect the subsumption relationships between functional parameters i.e., DAML concepts in their approach. Therefore they make service composition more flexible by considering a connection between two services, even if the output parameters of one service does not match exactly the input parameters of another service. In this direction it is still semantically safe to transfer the more specific output to the more general input (i.e., PlugIn matchmaking). Such axioms can be represented by Linear Logic implication, thus they can be asserted into their Linear Logic theorem prover as axioms. By considering the two latter axioms their approach is twofold: i) first the process model of a composite service can be generated directly from the complete proof, given these axioms and ii) in case all subsumption relations have been detected by the semantic reasoner, the web service composition system can deal with matchmaking of parameters during the composition process. Thus their Linear Logic theorem prover can deal with both the service specification and the semantic web information. Finally the authors consider the non-functional parameters (e.g., quality of services) directly in the theorem proving process in order to satisfy the user requirements.

2.2.3. A Matchmaker Based Approach

More recently the authors of \textsuperscript{29} have addressed in more detail the problem of interleaving Web service discovery and composition by means of their input and output parameters, but have considered only simple workflows where Web services have one input and one output parameter. In such a case their breath-first search approach returns a plan of Web service restricted to a sequence of limited Web services hence a linear workflow of Web services. In this way their solution retrieved a sequence of semantic links between Web services hence a linear and total order of services. With the aim of generating a composite service plan out of existing services, \textsuperscript{30} propose a composition path, which is a sequence of operators that compute data,
and connectors that provide data transport between operators. The search for possible operators to construct a sequence is based on the shortest path algorithm on the graph of operator space. However, they only considered two kinds of services operator and connector with one input and one output parameter (i.e., the simplest case for a service composition).

Contrary to the model proposed in this paper also considers services with more than one input and output parameter.

2.2.4. A Graph Theory Based Approach

In the authors consider a composition of services as a directed graph where nodes refer to Web services. Those nodes are linked by edges referring to a matching compatibility (a degree of match i.e., Exact, Subsume, PlugIn, Disjoint introduced by) between an input and an output parameter of two Web services (i.e., nodes). According to this graph they retrieve the shortest sequence of Web services from the initial requirements to the goal. At the end the ordering set of Web services matches all expected output parameters given the inputs provided by an end user. Their model is based on the degree of match introduced by to value semantic relations between service parameters.

Even if the latter model takes advantage of Exact, PlugIn, and Disjoint to infer semantic connections between services, the Subsume match level is not appropriate in composition of Web services. Using the Subsume match level to value a semantic connection means that the output parameter is not specific enough and then under specified to be exploited by the input parameter. In this way some inconsistencies are retrieved i.e., some information required by the input parameter is missing. Therefore the Subsume matchmaking type should not be directly applied in FLC. This issue is related to the robustness in Web service composition. By computing non robust connections and providing a method to overcome the problem of robustness in semantic Web service composition, our approach will ensure that returned compositions are robust enough to be executed.

2.2.5. A Golog Based Approach

In and in a Situation Calculus-based framework for services is proposed, where a service is seen by the client as an atomic action, thus presenting an input/output behaviour. A situation tree (i.e., a kind of process flow in the theory of Situation Calculus) is associated with such an atomic action. Services are specified as Golog procedures and a tool for automatic composition is presented: a user presents her goal to the system, expressed as a kind of generic (i.e., skeleton) procedure with user constraints and preferences. Such a user specification cannot be executed “as is”: it should be made executable by an agent that, exploiting an OWL-S ontology of services, automatically instantiates the user specification with services contained in such an ontology, by possibly pruning the situation tree corresponding to the
generic procedure in order to take user preferences and constraints into account. Such an instantiated user specification is a sequence of atomic actions (i.e., services) which are then executed by a Golog interpreter. In their approach a Web service is either an information-providing or world-altering service, but not both. A service with only outputs models strictly an information-providing web service (aka sensing action) whereas a service with only effects models an world-altering Web service. Roughly speaking world-altering services are services that change the world (e.g., any buying service) and information-gathering services are services that provide some actions of sensing to gather relevant information (e.g., a weather forecast service).

In the same way as in our approach services are seen as procedure calls. The main difference with our technique is related to the semantic level of potential links between Web services. Contrary to\cite{33} that considers preconditions and effects of Web services, we focus more on utilization of output parameters by input parameters of other Web services by means of semantic connections. In our approach the latter connections consider different semantic levels (Satisfiability, Subsumption, Intersection\cite{21}, Abduction\cite{35}, Concept Difference\cite{36}) of direct relations between Web services whereas\cite{33,12} check only Satisfiability between some effects and preconditions of Web services. Another difference is that in\cite{33,12} the final compositions are restricted to a sequence (total order) of services whereas our approach composes services as a partial order by means of sequences and concurrent execution (i.e., parallel branches) of Web services. By considering parallel branches of services more than one service can be used to provide information required by another Web service. Even if\cite{37} suggest to use an extended version of Golog i.e., ConGolog\cite{38} to overcome the problem of concurrent execution in semantic Web services, this approach still does not consider any degree of semantic level between Web service, hence no way to value Web service compositions at semantic level. Finally, in\cite{12} the outcome of the composition is not a service, in the sense that it cannot be re-used by another client, whereas in our work the composition produces a reusable specification.

2.2.6. An HTN Based AI Planning Approach

In\cite{39} the authors describe SHOP2, a hierarchical planning formalism for encoding the composition domains. This approach is more efficient but it doesn’t support complex constructs like loops. SHOP2 is a domain-independent HTN planning system. HTN planning is an AI planning methodology that creates plan by task decomposition. This is a process in which the planning system decomposes tasks into smaller and smaller subtasks, until primitive tasks are found that can be performed directly. The concept of task decomposition in HTN is very similar to the concept of process decomposition in OWL-S. One difference between SHOP2 and most other HTN planning systems is that SHOP2 plans for tasks in the same order that they will later be executed. Planning for tasks in the order they will be
performed makes it possible to know the current state of the world at each step in the planning process, which makes it possible for SHOP2's precondition evaluation mechanism to incorporate significant inference and reasoning power, including the ability to call external programs. This allows SHOP2 to integrate planning with external information sources as in the Web environment. In order to do planning in a given planning domain, SHOP2 needs to be given the knowledge about that domain. SHOP2s knowledge base contains operators and methods. Each operator is a description of what needs to be done to accomplish some primitive task, and each method tells how to decompose some compound task into partially ordered subtasks.

First, the model and their results are mainly dependent on the knowledge base of the domain. Such a constraint makes automated composition very hard to achieve since decomposition of Web services required to be known at composition time. Conversely our approach does not assume any required decompositions, and builds semantic connections (more or less methods and operators of SHOP2) between Web services of a composition on the fly. Second, given a composition problem the authors do not address methods to retrieve the whole set of feasible solutions, and even less criteria to compute best composition of Web services. In our work all valid compositions are returned. Moreover by exploiting semantic connections in the area of Web services, we provide a means to value semantics of Web service composition.

2.2.7. Summary

The latter forms of composition model are tightly related to Classical Planning in AI, and have been adopted by many other works. These, although different in the kind of goals and initial conditions, are all based on the idea of sequential compositions of the available services, which are considered as black boxes.

2.3. Web Service Composition Challenges

Although the previous approaches, combined with semantic reasoning and/or AI planning, offers practical approaches to perform web services composition, latter approaches encounter some limitations and require some extensions. The potentially enormous search space and the difficulty in fully and accurately representing real-world problems are two key challenges for the composition systems.

In this section we pose eight specific technical requirements, summarized in Table 1, that need to be met by AI planning and matchmaking based systems to make automatic web service composition at functional level a real success. The requirements are as following: i) Web Service Expressivity ($R_1$), ii) Composition Expressivity ($R_2$), iii) Composition Outcome ($R_3$), iv) Semantic Matchmaking between Output and Input Parameters ($R_4$), v) Non Persistence of Information ($R_5$), vi) Flexibility and Dynamicity ($R_6$), vii) Plan optimization ($R_7$), viii) DL Reasoning at Design Time ($R_8$).
2.3.1. Web Service Expressivity ($R_1$)

Any FLC based approach requires to consider not only i) preconditions and effects of services but also ii) their input and output parameters. Therefore i) semantic matchmaking between output and input parameters and ii) causal laws (e.g., Successor State Axioms in $^{33,12}$) between effects and preconditions are two main composability criteria.

Towards these issues $^{41}$ and $^{42}$ modelled input and output parameters together with preconditions and effects to describe semantic Web services. Unlike these methods, other models such as $^{43,39}$ and $^{33,44}$ consider web services with either effects or outputs, but not both. Therefore they restrict their composition models by assuming a complete independence between information providing and world-altering services. Such an assumption is very restrictive in the open world of web services.

In the same way as $^{25,26,24,45,31}$ do not reason on preconditions and effects for retrieving final and correct compositions. Indeed their models is mainly oriented by semantic matchmaking between output and input parameters of services. Such a limitation can affect the final composition result, by for instance, retrieving some compositions with open preconditions or inappropriate effects. $^{29,30}$ present a more restrictive model by focusing on services with one input and one output parameter.

It is obvious that the number of services together with their number of inputs, outputs, preconditions and effects have a direct impact on the performance of composition algorithms. However, to the best of our knowledge, no work focus on a trade-off between service expressivity and composition performance.

2.3.2. Composition Expressivity ($R_2$)

Structured composite services prescribe the order in which a collection of activities (here Web services) takes place. They describe how a service is created by composing the basic activities it performs into structures that express the control patterns, data flow, handling of faults and external events, and coordination of message exchanges between service instances. In this work, the expressivity of a composition states the control constructs the composition can handle. We identify the following three groups of control constructs for assembling primitive actions into a complex actions that collectively comprise an applications:

- Sequential ordering;
- Nondeterministic choice (i.e., conditional compositions);
-Concurrency, namely: parallel split, synchronization and exclusive choice.

While most of approaches support sequential ordering of services, a few supports both sequential and conditional compositions e.g., $^{41,42,29,30}$ and $^{24}$. Indeed alternative and conditional compositions cannot be retrieved at composition time since $^{33,44}$ retrieve only online compositions and then execute service at composition time. Therefore the main limitation of $^{33,44}$ is related to their composition result since only simple linear sequences of web services (viewed as a total order) can be
retrieved by their adaptation of Golog. Even if SHOP2 \cite{43,39} supports conditional effects in composition, the returned plan returned is a sequence of selected and executed service that achieve the goal. Thus they also do not address methods to retrieve the whole set of feasible solutions. Conditional compositions are also not considered by \cite{25,26,45,31} since their services consisted only of determinist output parameters.

\cite{45,31,25,26,24} and \cite{41} consider the concurrency construct as a key construct whereas \cite{43,39}, \cite{33,44}, and \cite{42} do not address this issue. Indeed such plans (i.e., composition) are often disregarded in AI planning based FLC. Indeed there is no mechanism to handle the control constructs related to concurrency in \cite{43,39}. This imposes a serious limitation on the usefulness of these methodologies. At the moment this is resolved by enumerating every possible flow in the process using conditional expressions in the method descriptions. This increases the complexity of search space, and planning.

2.3.3. Composition Outcome ($R_3$)

As the control constructs of web service composition, the outcome of the composition process need to be considered to value composition approaches. On the one hand \cite{43,39}, \cite{25,26} and \cite{24} modelled the outcome of their composer according to the OWL-S formalism or its former form. \cite{41} defined a formal model we can re-use once composition is achieved. On the other hand, the outcome of the composition in \cite{33,44} is not a service, in the sense that it cannot be re-used by another client.

2.3.4. Semantic Matchmaking between Functional Parameters ($R_4$)

Since FLC requires semantic matchmaking between output and input parameters of services, AI planning systems require some adaptations to consider the value of the latter matchmaking between output and input parameters of services. \cite{42}, \cite{33,44}, and \cite{46} compile composition under the assumption of exact matches between functional parameters, hence no way to value connections between web services at semantic level. In \cite{47}, techniques are introduced that disambiguate concept names during composition. In \cite{43,39} and \cite{25,26}, they adapted their work on the open-world description logics semantics of OWL-S, under a further match i.e., PlugIn. In \cite{24,29,30} and \cite{45,31} authors extends the latter works by retrieving semantic connections valued by a Subsume match level. However, as previously pointed out, the Subsume matchmaking type between an output and an input parameter requires some refinements to be fully efficient for FLC. In \cite{22}, partial matches are treated in a fully automatic approach to web service composition. In a nutshell, the approach uses numeric intervals to encode a sub-concept hierarchy, and then performs matching (as well as service discovery) based on those intervals, simply by asking whether one interval (a service output) is fully contained in the union of a set of intervals (service inputs). Even if \cite{41} suggest also partial matches to overcome web service
composition, they do not study in detail the limitation of their matchmaking type e.g., if the partial match of n output parameters subsume an input parameter of another service, matchmaking of those parameters is not appropriate to perform a composition. Moreover no detail about the computation of partial matches is presented in 41. Even if a DL reasoner is used to perform this task, we do not know if DL reasoning is pre-computed or achieved at composition time.

Finally, even if partial and standard matchmaking levels are considered between functional parameters of services in some works, non standard matchmaking such as Abduction 35, Contraction 48, Difference 36 are not considered in the latter works.

2.3.5. A Restrictive Assumption: Non Persistence of Information (R5)

In the open and dynamic world of web services, there are many scenarios (e.g., in Telecom domain) where the reasonable persistence of information, as suggested by 33,44 (and re-used by 43,39), cannot hold. Indeed information has a limited temporal extent associated with it, which may affect composition. Therefore we cannot make such an assumption in our context. The reasonable persistence of information assumption in 33,44 prevents the planner to change (i.e. simulate the changes) the information gathered from external sources.

2.3.6. Flexibility and Dynamicity (R6)

Since our work addresses automated web service composition, automation and dynamicity of the model is one of the most important requirement. 34 does not consider such features in their approach. In contrary they present a semi automated method since the user is required to select a Web service in a restricted list.

The model introduced by 43,39 together with its composition result are mainly dependent on the knowledge base of the domain i.e., a detailed description of compound tasks decomposed into partially ordered primitive tasks. Such a constraint makes automated composition very hard to achieve since decomposition of web services required to be known at composition time.

2.3.7. An Optimization Criterion for Web Service Composition (R7)

On the one hand Web services consume resources, such as network bandwidth, and may have a monetary cost associated with their execution. On the other hand, the semantic form of web service enables different levels of matchmaking between their functional parameters. Therefore different metrics can be used to optimize service selection as well as the resulting composition. 49,50 focused on QoS-aware service composition. To this end, they suggest a QoS-driven approach to select candidate services valued by non functional criteria such as price, execution time, and reliability. As 49 the authors of 25,26 only studied non functional parameters of services to retrieve optimal composition. This approach is then, unfortunately, only non functional parameters oriented (e.g., quality of service). Even if such an
optimization is adapted for web service compositions, it is far from convenient for “semantic” web service compositions. Unlike the previous approaches, consider semantic optimization by valuing semantic connections between services. However the optimization is first, performed by end-user, and secondly, only local. Indeed, during the composition process, the end-user may only choose the best local semantic web services, depending on its semantic connection with its previous services. Even if suggest a method to prune the enormous search space by considering preference constraints, no criteria are suggested to distinguished similar services and compositions. In the same way, neither nor address the problem of optimization in web service composition.

2.3.8. DL Reasoning at Design Time (Rs)

In most of matchmaking based approaches DL reasoning is done at run time. In case the domain ontology is complex, computing subsumption relationship between functional parameters at composition time can be a challenging task.

2.4. Discussion

This section has aimed to give an overview of recent progress in automated Web services composition techniques. To this end, we focused more specially on FLC and we presented main methods studied in this level of composition i.e., AI planning and Matchmaking based approaches. From the latter study we suggested eight specific technical requirements (i.e., ) that need to be met by FLC composers to make automatic web service composition a success.

In the following we will invest on a novel FLC technique that fulfil a subset of these requirements by:

1. considering information-gathering services to gather relevant information. Therefore input and output parameters will be considered to retrieve compositions of services (part of Requirement ). Moreover the number of functional parameters of the latter services will not be limited, or only in cases of performance research;
2. returning compositions as a partial order of relevant services which support not only sequential ordering but also Nondeterministic choice and concurrency (Requirement );
3. producing an abstract composition result as a reusable specification we can adapt for interfacing with standard proposals of web services orchestration such as OWL-S, BPEL4WS (Requirement );
4. considering semantic connection between web services (through their input and output parameters) as a key concept to perform FLC (Requirement ). In a nutshell the technique relies on domain-dependent ontology for calculating semantic similarity scores between the concepts in service descriptions, and applies this score to guide the searching process of the planning algorithm. Moreover our
### Table 1. Some Functional Level Based Web Service Composition Approaches

<table>
<thead>
<tr>
<th>Composition Approaches</th>
<th>Formalism</th>
<th>Independent</th>
<th>Linear Logic</th>
<th>Similarity with HTN</th>
<th>Graph Theory</th>
<th>Situation Calculus</th>
<th>Hierarchical Task Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composition Algorithm</td>
<td>No Automation</td>
<td>Theorem Proving</td>
<td>Breadth-first Search</td>
<td>Dynamic Programming</td>
<td>Golog + World Simplest Breath First Planner (wshfp)</td>
<td>SHOP2</td>
<td>51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Composition Outputs</th>
<th>Service Description</th>
<th>OWL-S</th>
<th>DAML-S</th>
<th>RDF</th>
<th>DAML-S, SA-WSDL</th>
<th>OWL-S</th>
<th>OWL-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>Service Category (R1)</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Information-gathering or World-altering</td>
<td>Information-gathering or World-altering</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Composition Results</td>
<td>Composition Expresiveness (R2)</td>
<td>Sequence</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Non Determinism (R3)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Concurrency (R4)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Composition Outcome (R5)</td>
<td>OWL-S</td>
<td>DAML-S</td>
<td>Independent</td>
<td>Independent</td>
<td>✓</td>
<td>OWL-S</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Composability Criteria</th>
<th>Semantic Function (R1)</th>
<th>Matchmaking Exact, PlugIn Subsume, Disjoint</th>
<th>Exact, PlugIn Subsume, Disjoint</th>
<th>Exact, PlugIn Subsume, Disjoint</th>
<th>Exact, PlugIn Subsume, Disjoint</th>
<th>Exact</th>
<th>Exact, PlugIn Subsume, Disjoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL Reasoning (R6)</td>
<td>Design Time</td>
<td>Design Time</td>
<td>Design Time</td>
<td>Run Time</td>
<td>✓</td>
<td>Design Time</td>
<td></td>
</tr>
<tr>
<td>Causal Laws (R7)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Successor State Axioms (R8)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Assumption Persistence (R9)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Additional Flexibility (R10)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Local, Manual on semantic connections (R11)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Number of services User Preference</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Legend: ✓ = fully supported, × = not supported.
proposal is characterized by the fact that semantic robustness of any composition is checked during the composition process, which is unique to our proposal. The experimental results will demonstrate that such an approach performs superior compared to the traditional planning based techniques;

(5) considering that information has a limited temporal extent associated with it (Requirement $R_5$) i.e., by not assuming that information is persistent. In other words the suggested approach will provide conditional compositions at design time;

(6) returning composition of services on the fly (Requirement $R_6$). Contrary to many approaches (e.g., $^{43,39}$) we will do not assume any required decompositions, and builds semantic connections (more or less methods and operators of SHOP2) of a composition on the fly;

(7) introducing a model for facilitating computation of semantic connections between services and also their composition (Requirement $R_8$). Thus performing DL reasoning on semantic connections between services will be done at Design Time.

Not surprisingly, our approach will have roots in both AI planning systems and semantic matchmaking based approach to meet requirements $R_{4,2\leq i \leq 6}$, and parts of requirement $R_1$ (preconditions of services are supposed to be satisfied). Even if a semantic model is presented in this work to perform FLC, presenting a general and extensible model to evaluate quality of both elementary and composition of semantic connections (i.e., Requirement $R_7$) is not the scope of this work. A global semantic connection selection based approach to retrieve the optimal composition is studied in this direction by $^{53}$. In such an approach evaluating “Semantics” of Web service composition will be possible.

3. A Motivating Telecom Use Case

The suggested approach is being exploited in the VoIE Project, conceived by France Telecom R&D. In the domain of Internet packages, commercial offers proposed by Telecom operators are used to be composed of a set of more technical, optional offers. From now Telecom operators such as France Telecom suggest only a few restricted set of pre-existing commercial offers. These offers are well known as All in one or again Global Internet Access$^d$. The end users can then choose a pre-existing commercial offer among the available offers. Even if designing pre-defined packaged offers have at least one significant benefit for Telecom operators i.e., ease to maintain, this approach is far from convenient for end users. Indeed this set of pre-existing commercial offers does not still satisfy the end user constraints and preferences.

The main motivation of our work is to give to the end user the possibility to create their own commercial offers according to their real needs, without any

$d$http://www.orange.com/english/home.php
kind of assistance. Such an issue is really challenging in the domain of Telecom since it addresses other industrial issues related to dynamic and preference-based packaging. Moreover our proposal aims at improving the return on investment (ROI) of any Telecom operator by reducing the Time-to-product (dynamic and automated process), Time-to-market, with lower price (repetitive tasks), better precision, better end-user satisfaction (no constraint-based packages), the whole with more creativity.

In the case under consideration the result of a dynamic and automated generation of commercial offers is a complete customized offer wherein the end user is only in charge of selecting the offer(s) she wants to subscribe. For instance the offers may be as follows, ADSL eligibility (ELIG), LiveBox, Voice over IP (VOIP), Address Book (AB), Visiophone (VP), Television over IP i.e., Internet telephony (IPTV), Voice Messaging (VM), High Definition television (HDTV), High speed WiFi (HSW), Email service (EMS), Virtual Drive (VD) and so on. With the aim of dynamically generating customized packages each of the latter offers are interfaced by a semantic Web service (e.g., IPTVService interfaces the IPTV offer), facilitating flexible and scalable applications through the loosely coupled features of Web services. The semantic features of semantic Web services enable not only to represent knowledge and improve expressivity levels of their functional parameters i.e., IOPEs, but also to reason about those parameters and their potential semantic connections. Here we focus on this last aspect and will provide some practical examples of our approach. The ultimate goal is to provide a correct composition of the latter Web services in regard to their semantic connections. Figure 1 describes a subset of defined concepts in the $\mathcal{ALE}$ domain ontology $\mathcal{T}$ (305 defined concepts and 117 object properties) used to describe the domain i.e., all input and output parameters of semantic Web services refer to concepts of the ontology $\mathcal{T}$. The description Logic (DL) $\mathcal{ALE}$ is used to model our domain due to its interesting trade-off between expressivity and complexity. In the following example, we consider six Web services i.e., a subset of the 35 Web services included in the real scenario:

- $\text{AdslEligibility-}$, $\text{AdslEligibility}$ and $\text{AdslEligibility+}$, that, starting from a $\text{PhoneNum}$, a $\text{ZipCode}$ and an $\text{Email}$ address, returns respectively the $\text{SlowNetworkConnection}$, $\text{NetworkConnection}$ and $\text{FastNetworkConnection}$ of the desired zone;
- $\text{VoiceOverIP}$, that, starting from a $\text{PhoneNum}$ and a $\text{SlowNetworkConnection}$, returns the $\text{VoIPId}$ of the ADSL line a Telecom operator needs to install the line;
- $\text{TvOverIP}$, that, starting from a $\text{PhoneNum}$ and a $\text{FastNetworkConnection}$, returns a serial number of a $\text{VideoDecoder}$ required to access video over IP;
- a $\text{LiveBox}$ service returns the $\text{Invoice}$ of the commercial offer the user requested, depending on a $\text{PhoneNum}$, $\text{IPAddress}$ and serial number of a $\text{Decoder}$.

**Example 3.1. (Motivating Example Illustration)**
Suppose a client wants to customize and create her own internet package from a set
of available technical and optional offers. The client is then in charge of selecting
the offer she is interesting in e.g., `AdslEligibility` to retrieve information about
her connection details, `VoiceOverIP` for Internet telephony, `TvOverIP` to obtain the
Television access via her internet connection and `LiveBox` to get the invoice and
technical details of her customized internet package. Our goal consists in retrieving
a correct composition i.e., a partial order amongst the selected offers (i.e., services)
since the selected offers required to be ordered. Indeed some constraints between
services are in place to perform the customized internet package. For instance the
service `AdslEligibility` is required to be executed first since such a service is
required to obtain technical details about the internet connection the user could
receive. Then the services `VoiceOverIP` and `TvOverIP` might be launched depending
on the result of the `AdslEligibility` service. Finally the `LiveBox` service is
in charge of aggregating information provided by the `VoiceOverIP` and `TvOverIP`
services to calculate and send the final invoice to the client. The latter partial or-
der is defined by means of semantic constraints between services. As we will see
in the next section such semantic constraints are mainly guided by the functional
parameters of services i.e., input and output parameters. Since the main idea is to
customize commercial offers in an automated way, it seems conceivable to suggest
a solution that computes compositions of services in a dynamic way, depending on
the selected offers.

In such a scenario automation of Web service composition is a real and still an
open issue, not only for France Telecom but also for any other Telecom operator
since the number of offers i.e., Web services the user can choose is more and more
increasing. Even if this number of services is relatively closed and bounded in our
scenario, it can be conceivable that a service be offered by any Telecom operator
or other service provider. In this way the end-user will be able to compose its own
internet package from varied services providers. Such cases will cause a duplication
of services (from different sources) involved in the compositions, hence an exponen-
tially increase of the number of service compositions. Indeed the more offers
the harder the composition will be. That is why we suggest to compose automat-
ically Web services depending on the user requirements (through the commercial
offer she subscribed) and the service compatibilities (i.e., through their semantic
connections).

4. Semantic Links between Web Services

In this section we introduce the definition of semantic links in semantic Web service
composition i.e., a formal concept for representing semantic connections between
services (Requirement $R_4$). As previously introduced semantic connections (i.e.,
semantic links) are required as essential in the context of Web service composition
since they highlights the fact that an output parameter of a Web service can be
exploited by an input parameter of another Web service (Requirement $R_1$).
4.1. Definitions, Semantic Context and Assumptions

4.1.1. Ontology, Terminological and Assertional Boxes

The formal model required to represent semantics on functional parameters of Web services is the Ontology. Formally the ontology we used is based on a DL $\mathcal{ALC}$ wherein a distinction is drawn between the so-called Terminological Box (TBox) and the Assertional Box (ABox). The TBox contains intentional knowledge in the form of a terminology, and the ABox contains extensional knowledge or so called assertional knowledge. The key role in both the TBox and the ABox is description logic that offers inference capability based on terminology and assertions. The ontology of our domain is defined as $\mathcal{T} := \langle \mathcal{T}, \mathcal{A} \rangle$ of the studied domain wherein $\mathcal{T}$ be the TBox and $\mathcal{A}$ be the ABox. Figure 1 shows a subpart of the ontology $\mathcal{T}$ we consider in our motivating scenario i.e., the TBox $\mathcal{T}$.

4.1.2. Semantic Web Services in a Nutshell

In the considered approach, all parameters of semantic Web services referred to concepts in the TBox $\mathcal{T}$ of a domain ontology $\mathcal{T}$. In other words syntactic Web services have been enhanced with semantic annotation of their functional parameters. Since we operate with information-gathering services, such services require and provide some information. On the one hand Web services require some instances (defined in $\mathcal{A}$) of their input parameters (defined in $\mathcal{T}$) to be executed. On the other hand Web services return some instances (still defined in $\mathcal{A}$) of each output parameters of the latter services.
Example 4.1. (A Semantic Web Service at Design and Run Time)
Suppose the motivating example introduced in Section 3 with the Web service AdslEligibility. Such a Web service is semantically enhanced by means of semantic annotation of its functional parameters i) input parameters i.e., Phone Number, ZipCode, EMail and ii) output parameters i.e., Network Connection (Figure 2(a)), the whole with respect to the domain ontology $T$ and TBox $T$ (Figure 1). Once its execution is performed we obtain an instance of Network Connection which is in the ABox i.e., Id_{NC,Fast,1024,EligibleMax} (Figure 2(b)). To this end the three input parameters of the service has been first instantiated.

Roughly speaking, given a set of instances in an ABox, semantic Web services are able to generate some instances of (potentially new) concepts, depending on their semantic definitions.

4.1.3. Assumptions on Parameters of Web Services
In this work, some assumptions related to the functional description of Web services has been required. First of all we restrict (without loss of generalization) the semantic scope of input and output parameters of Web services. Secondly preconditions of any Web services is supposed to be, a priori, satisfied by effects of other services.
or in the initial situation.

- **A Common Ontology for Functional Parameters of Services and Goals:**
  
  Since our approach focuses on semantic links based Web service composition, we assume that any output parameter of any service can be semantically compared with any input parameter. To this end, functional parameters of services are annotated by concepts using a common ontology or Terminology $T$. In the same way user goals (i.e., an abstract services defined in terms of their input and output parameters) are described completely and sufficiently by concepts chosen from the ontology. Therefore any kind of reasoning (e.g., Subsumption, Difference) on functional parameters can be calculated. This assumption holds along the paper, especially to focus on Web service composition. Distributed ontologies is not considered here but is largely independent of the problem addressed in this paper.

  The interested readers may consult the following book \textsuperscript{54} published recently to overcome issue related to distributed ontologies.

- **Assumptions on Preconditions and Effects of Web Services:**
  
  Even if we focus more on input and output parameters of services we keep in mind that preconditions and effects have to be also considered in a composition model. Thus our composition approach deals with a part of requirement $R_1$. In this direction we assume for sake of simplicity that preconditions of all Web service involved in a composition are satisfied by some effects of some preceding Web services in the composition. Disregarding open issues related to preconditions and effects in this work, first does simplify the composition process, second enables to highlight open issues related to data flow in Web service composition such as robustness, and third allows scaling up to large sets of capabilities; in turn, a relevant part of the interaction taking place amongst services is ignored.

### 4.2. Semantic Links

Here we address the composition problem as a discovery of semantic connections between Web services, which justify our focus on semantic links. In the considered context retrieving a semantic connection between two Web services $s_x$ and $s_y$ is similar to discover a semantic similarity between an output parameter $Out_{s_y}$ of $s_y$ and an input parameter $In_{s_x}$ of $s_x$ (or vice versa). Consequently the goal is to find a matchmaking function between two knowledge representations encoded using the same ontology $T$. Semantic links\textsuperscript{e} (first introduced by \textsuperscript{18}) between Web services will be in charge of valuating these semantic matchmaking functions. Therefore semantic links measure the semantic quality of links (i.e., quality of matchmaking functions) between Web services.

\textsuperscript{e}In the research area of AI planning, other kinds of links, well known as causal links or \textbf{protection intervals} \textsuperscript{55,56} are computed to perform planning.
Definition 4.1. (Semantic Link)
A semantic link \(\langle s_y, \text{Sim}_T(\text{Out}_{s_y}, \text{In}_{s_x}), s_x \rangle\) is related to a logical dependency among an output \(\text{Out}_{s_y}\) and input parameter \(\text{In}_{s_x}\) of two different services \(s_y\) and \(s_x\).

Roughly speaking a semantic link describes a semantic relation between an output parameter \(\text{Out}_{s_y} \in \mathcal{T}\) of a Web service \(s_y\) and an input parameter \(\text{In}_{s_x} \in \mathcal{T}\) of a Web service \(s_x\). Thereby \(s_x\) and \(s_y\) are semantically and partially linked according to a matchmaking function \(\text{Sim}_T(\text{Out}_{s_y}, \text{In}_{s_x})\). By definition a semantic link \(\langle s_y, \text{Sim}_T(\text{Out}_{s_y}, \text{In}_{s_x}), s_x \rangle\) implies that i) \(s_y\) precedes \(s_x\) since an output of \(s_y\) is exploited by an input of \(s_x\); and ii) no Web service is interleaved between \(s_x\) and \(s_y\). The matchmaking function \(\text{Sim}_T\) informs about the matchmaking type between the two parameters as concepts \(\text{Out}_{s_y}, \text{In}_{s_x} \in \mathcal{T}\). Moreover the latter function is useful not only to value the possible semantic connections between two Web services but also to compare them. Indeed it is obvious that some matchmaking types will be preferred and other will be disregarded with respect to their semantic quality. Let \(s_y\) and \(s_x\) be two Web services with their respective output parameters \(\text{Out}_{s_y}\) and \(\text{Out}_{s_x}\). Suppose \(s_x\) such that \(\text{Out}_{s_y}\) and \(\text{Out}_{s_x}\) semantically match with \(\text{In}_{s_x}\). \(\text{Sim}_T\) is required to value the two connections \((\text{Out}_{s_y}, \text{In}_{s_x})\) and \((\text{Out}_{s_x}, \text{In}_{s_x})\).
and also to order them.

Example 4.2. (Semantic Link Illustration)
Suppose AdslEligibility (ELIG) and VoiceOverIP (VOIP) be two Web services introduced in the motivating example (Section 3). We assume without loss of generality that the AdslEligibility service precedes the VoiceOverIP service. According to the previous definition the two latter Web services are connected by two semantic links i.e., (i) $sl_a$ described by $\langle \text{ELIG}, \text{Sim}_T(\text{NetworkConnection}, \text{SlowNetworkConnection}), \text{VOIP} \rangle$ (Figure 3(a)); (ii) $sl_b$ described by $\langle \text{ELIG}, \text{Sim}_T(\text{NetworkConnection}, \text{PhoneNumber}), \text{VOIP} \rangle$ (Figure3(b)). By (i) and (ii) two semantic connections valued by $\text{Sim}_T$ between an output parameter of AdslEligibility service and an input parameter of VoiceOverIP service are conceivable. However it is obvious that the semantic links $sl_a$ and $sl_b$ are not valued by the same matchmaking function with respect to the domain ontology $T$ and TBox $T$ (Figure 1).

4.3. Valuation of Semantic Links
Despite some existing methods \cite{32,21}, solving a mapping problem is hard because the syntactic form of two knowledge representations rarely matches exactly. Indeed the semantic matchmaking $\text{Sim}_T$ retrieved between an output and input parameter does not still refer to an Exact match e.g., $sl_a$ in the previous example. That is why $\text{Sim}_T$ aims at expressing which matching type is used to chain Web services. In many Web service composition models \cite{31,43,57,25} this function is often reduced to the five well known matchmaking functions introduced by \cite{32} with the extra match level Intersection of \cite{21} (Requirement $R_4$):

- **Exact** ($\equiv$) If the output parameter $\text{Out}_{s_y}$ of $s_y$ and the input parameter $\text{In}_{s_x}$ of $s_x$ are equivalent concepts; formally, $\langle T, A \rangle \models \text{Out}_{s_y} \equiv \text{In}_{s_x}$.
- **PlugIn** ($\subseteq$) If $\text{Out}_{s_y}$ is sub-concept of $\text{In}_{s_x}$; formally, $\langle T, A \rangle \models \text{Out}_{s_y} \subseteq \text{In}_{s_x}$.
- **Subsume** ($\supseteq$) If $\text{Out}_{s_y}$ is super-concept of $\text{In}_{s_x}$; formally, $\langle T, A \rangle \models \text{Out}_{s_y} \supseteq \text{In}_{s_x}$.
- **Intersection** ($\cap$) If the intersection of $\text{Out}_{s_y}$ and $\text{In}_{s_x}$ is satisfiable; formally, $\langle T, A \rangle \models \text{Out}_{s_y} \cap \text{In}_{s_x} \subseteq \bot$.
- **Disjoint** ($\perp$) Otherwise $\text{Out}_{s_y}$ and $\text{In}_{s_x}$ are incompatible i.e., $\langle T, A \rangle \models \text{Out}_{s_y} \cap \text{In}_{s_x} \subseteq \perp$.

For instance, the PlugIn match means that an output parameter of a service $s_y$ is subsumed by an input parameter of the succeeding service $s_x$ whereas the Subsume match means that an output parameter of a service $s_y$ subsumes an input parameter of the succeeding service $s_x$. According to the previous matchmaking functions, a valid semantic link is defined by means of Definition 4.2.
Definition 4.2. (Valid Semantic Link)
A semantic link \( \langle s_y, \text{Sim}_T(\text{Out}_{s_y}, \text{In}_{s_x}), s_x \rangle \) is valid iff \( \text{Sim}_T(\text{Out}_{s_y}, \text{In}_{s_x}) \neq \text{Disjoint} \), i.e., \( \langle T, A \rangle \neq \text{Out}_{s_y} \cap \text{In}_{s_x} \subseteq \bot \).

Roughly speaking, a valid semantic link between two Web services describes a potential match between two parameters (i.e., \( \langle T, A \rangle \neq \text{Out}_{s_y} \cap \text{In}_{s_x} \subseteq \bot \)), hence a valid and potential link between these Web services to form a composition. A valid composition consisted of only valid semantic links.

Example 4.3. (Valid Semantic Link Illustration)
Consider the previous example with the semantic links \( sl_a \) and \( sl_b \). It is trivial that the matchmaking function referred by \( sl_a \) is Subsume whereas the matchmaking function referred by \( sl_b \) is Disjoint. Indeed \( \langle T, A \rangle \models \text{NetworkConnection} \sqsubseteq \text{SlowNetworkConnection} \) and \( \langle T, A \rangle \models \text{NetworkConnection} \cap \text{PhoneNumber} \sqsubseteq \bot \) with respect to the domain ontology \( T \). According to Definition 4.2 \( sl_a \) is a valid semantic link whereas \( sl_b \) is not.

In the following the five previous semantic matching functions considered to value semantic links will be ordered to ease their comparison. Such an ordering is required to infer which semantic matching function i.e., semantic link is more general or specific than another. To do this we suggest to first order them with the logical implication operator \( \Rightarrow \) as presented in Theorem 4.1. Second the semantic matching functions will be discretized according to the previous partial order (Table 2). The discretization of the matchmaking function enables to estimate and represent the semantic quality of a semantic links.

Theorem 4.1. (Partial Order on Semantic Matchmaking Functions)
The partial order on the matchmaking functions Exact, PlugIn, Subsume, Intersection is defined by the relations (i) and (ii) where \( \Rightarrow \) refers to the binary and logical implication between \( \text{Out}_{s_y} \setminus \{\bot\} \) and \( \text{In}_{s_x} \setminus \{\bot\} \).

(i) Exact \( \Rightarrow \) PlugIn \( \Rightarrow \) Intersection
(ii) Exact \( \Rightarrow \) Subsume \( \Rightarrow \) Intersection

Proof. The proof of this theorem is divided into four different steps. Each step follows a trivial logical implication. \( \square \)

The function of matchmaking described by \( \text{Sim}_T(\text{Out}_{s_y}, \text{In}_{s_x}) \) between an output parameter \( \text{Out}_{s_y} \) of \( s_y \) and an input parameter \( \text{In}_{s_x} \) of \( s_x \) is close to the \( \text{degOfMatch}(\text{Out}_{s_y}, \text{In}_{s_x}) \) function introduced by \(^{32}\). However this function is considered in Web service composition and not in discovery. Moreover \( \text{Sim}_T \) is extended with the Intersection matchmaking function to consider more degrees of

\(^{32}\)Since \( \forall \text{out}_{s_y}, \bot \sqsubseteq \text{out}_{s_y} \not\Rightarrow \bot \cap \text{out}_{s_y} \), we required to consider any \( \text{out}_{s_y} \) different from \( \bot \) in our theorem. In the same way any \( \text{In}_{s_x} \) is different from \( \bot \).
Match Type | Logic meaning | Discrete $\text{Sim}_T(\text{Out}_{s_y}, \text{In}_{s_x})$
---|---|---
Exact ($\equiv$) | $(T, A) \models \text{Out}_{s_y} \equiv \text{In}_{s_x}$ | $1$
Plug-in ($\sqsubseteq$) | $(T, A) \models \text{Out}_{s_y} \sqsubseteq \text{In}_{s_x}$ | $\frac{1}{2}$
Subsume ($\sqsupseteq$) | $(T, A) \models \text{Out}_{s_y} \sqsupseteq \text{In}_{s_x}$ | $\frac{1}{2}$
Intersection ($\sqcap$) | $(T, A) \not\models \text{Out}_{s_y} \sqcap \text{In}_{s_x} \sqsubseteq \bot$ | $\frac{1}{4}$
Disjoint ($\perp$) | $(T, A) \models \text{Out}_{s_y} \sqcap \text{In}_{s_x} \sqsubseteq \bot$ | $0$

Table 2. Discretization of Semantic matching functions described by $\text{Sim}_T$.

semantic matching. The suggested approach introduced also a partial order based on the logical implication relation to compare semantic links and their values.

4.4. Robust Semantic Link

The five match levels are far from enough to bring Web service composition as a semantic links composition to its full potential. The Exact match is clearly appropriate to chain two Web service parameters since they refer to equivalent concepts. The Plug-in match is also a possible match to plug an output parameter in an input parameter of another Web service since the output parameter provides more information than the input parameter required. The Disjoint match informs about the incompatibility of two Web service parameters hence an invalid semantic link. Even if the matchmaking Exact, Plug-in, and Disjoint can be used without any change to value semantic links in a Web service composition, the match levels Intersection and Subsume need some refinements to be fully efficient for semantic links composition. Suppose a semantic link $\langle s_y, \text{Sim}_T(\text{Out}_{s_y}, \text{In}_{s_x}), s_x \rangle$ valued by a Subsume match level i.e., $(T, A) \models \text{Out}_{s_y} \sqsubseteq \text{In}_{s_x}$. It is obvious that such a semantic link should not be directly applied in a Web service composition since the output parameter $\text{Out}_{s_y}$ is not specific and specified enough to be exploited by the input parameter $\text{In}_{s_x}$. We say also that $\text{In}_{s_x}$ is over specified and $\text{Out}_{s_y}$ is under specified. In other words the output parameter $\text{Out}_{s_y}$ requires an Extra Description to obtain a composition of these two Web services. In the same way a semantic link valued by an Intersection match needs a comparable refinement. That is, two different kinds of semantic links requires more attention: robust and non robust semantic links.

Definition 4.3. (Robust Semantic Link)

A semantic link $\langle s_y, \text{Sim}_T(\text{Out}_{s_y}, \text{In}_{s_x}), s_x \rangle$ is robust if and only if $\text{Sim}_T(\text{Out}_{s_y}, \text{In}_{s_x})$ is either Exact or Plug-in.

Example 4.4. (Some Limits of standard Matching functions)

Let $s_l$ illustrated in Figure 3(a) be the valid semantic link defined by $\langle \text{ELIG}, \text{Sim}_T(\text{NetworkConnection}, \text{SlowNetworkConnection}), \text{VOIP} \rangle$. By Definition 4.3 the valid semantic link $s_l$ is not robust (Figure 4). Indeed the match level of $s_l$ is Subsume since $(T, A) \models \text{NetworkConnection} \sqsubseteq \bot$.
SlowNetworkConnection. It is obvious that this semantic link can not be applied in a composition since the output parameter NetworkConnection is under specified to be exploited by the input parameter SlowNetworkConnection. The output parameter NetworkConnection requires an Extra Description to enable a composition of these two services.

Once the definition of robust semantic links introduced the notion of robust composition of Web services is as following.

Definition 4.4. (Robust Web Service Composition)
A composition of Web services is robust if and only if all its semantic links are robust.

Robust semantic links in a Web service composition are key components since they enable to obtain robust Web service composition. In the opposite Web service composition with non robust semantic links can fail since some information is missing.

4.5. Extension of Semantic Matching Functions
A possible way to state the problem obtained for non robust semantic links \( \langle s_y, \text{Sim}(\text{Out}_s, s_x) \rangle \) valued with an Intersection or a Subsume match level is to find the information contained by the input parameter \( \text{In}_s \) and not by the output parameter \( \text{Out}_s \). In other words some descriptions have to be retrieved to transform a non robust semantic link in its robust form.

To do this, we exploit a non-standard inference match level for DLs i.e., the difference (well known as subtraction operation) introduced by 58 for comparing DL descriptions and adapt it to the problem of semantic matching between Web service parameters. The difference operator, first introduced by 36, enables to remove from a given description all the information contained in another description. The difference between two concept descriptions \( C \) and \( D \) with \( C \sqsubseteq D \) is given by (1)

\[
C - D := \max\{B|B \cap D \equiv C\}
\]
In other words, \( B \) represents an explanation on why \( D \) is not classified by \( C \) with respect to \( T \). This definition requires that the second argument subsumes the first one. However the difference \( C - D \) between two incomparable descriptions \( C \) and \( D \) (i.e., \( C \) is not subsumed by \( D \)) can be given by constructing the least common subsumer of \( C \) and \( D \), that is, \( C - D := C - lcs(C, D) \). From this, \(^{58}\) proposed a refinement of definition (1) by taking the syntactic minimum (w.r.t. a subdescription ordering \( \preceq_a \)) instead of a semantic maximum. Thus they defined the difference between two (in)comparable concept descriptions \( C \) and \( D \) as (2)

\[
C \triangleq D := \min_{\preceq_a} \{ B \mid B \cap D \equiv C \cap D \}
\]  

(2)

Even if (1) captures the real semantic difference between two concept descriptions, (2) has two main advantages. Firstly its result does not contain redundancies and secondly but not of least importance, its result is more readable by a human user.

**Remark 4.1.** Considering an \( \mathcal{ALN} \) DL, Concept Abduction \(^{60,35}\) is also able to compute a concept expression \( B \) representing what is underspecified in \( D \) in order to completely satisfy \( C \) taking into account the information modelled in a TBox \( T \). Concept Abduction may appear similar to Concept Difference \(^{36}\), yet it is not so. Indeed performing a difference operation requires a subsumption relation between descriptions to be matched since Concept Difference (1) requires the extra strict condition \( C \sqsubseteq D \). This strict condition may make Concept Difference (1) hard to use in a matchmaking process (e.g., Intersection match level), where descriptions overlap is usually a sufficient condition to start the process. However Concept Difference (1) and (2) are more accurate than Concept Abduction since (1) and (2) perform an equivalence between two concept descriptions \( (\langle T, A \rangle \models B \cap D \equiv C \cap D) \) whereas the Concept Abduction computes a subsumption of concept descriptions \( (\langle T, A \rangle \models B \cap D \sqsubseteq C) \).

In the following we suggest to use (2) since it is defined for \( \mathcal{ALE} \) descriptions logics. Moreover (2) is also defined for incomparable descriptions without change in their definitions. In this direction the Concept Difference is primarily performed to capture in a logical way the reason why an output parameter \( \text{Out}_{s_y} \) of a \( s_y \) and an input parameter \( \text{In}_{s_x} \) of \( s_x \) may not be chained by a robust semantic link. The idea behind our approach is the following. In case a semantic link \( \langle s_y, \text{Sim}_{T}(\text{Out}_{s_y}, \text{In}_{s_x}), s_x \rangle \) is valid (not valued by a Disjoint matchmaking) but not robust (i.e., valued by neither an Exact nor a PlugIn matchmaking), we compare \( \text{Out}_{s_y} \) and \( \text{In}_{s_x} \) to obtain two kinds of information:

(i) the **Extra Description** \( \text{In}_{s_x} \setminus \text{Out}_{s_y} \) that refers to information required but not provided by \( \text{Out}_{s_y} \) in order to semantically link it to the input \( \text{In}_{s_x} \) of \( s_x \);  
(ii) the **Common Description** \( \text{Out}_{s_y} \cap \text{In}_{s_x} \) that refers to information required by \( \text{In}_{s_x} \) and effectively provided by \( \text{Out}_{s_y} \).

As previously said semantic links concerned by the computation of concept difference are links valued by either a Subsume or an Intersection matching i.e., non
robust semantic links. In the Subsume case we compute the Extra Description $B$ contained in $In_s x$ such that the matching between $B \cap Out_s y$ and $In_s x$ be Exact. By (2) this Extra Description $In_s x \setminus Out_s y$ is defined by $\min_{\preceq d} \{ B | B \cap Out_s y \equiv In_s x \}$ since $Out_s y \supseteq In_s x$. In case the semantic link is valued by an Intersection match (i.e., $\neg (Out_s y \cap In_s x \sqsubseteq \bot)$) we compute the Extra Description $B$ that is not specified in $Out_s y$ to reach a PlugIn match between $B \cap Out_s y$ and $In_s x$.

**Example 4.5. (Extra Description Illustration)** Let $sl_b$ illustrated in Figure 4 be the non robust semantic link defined by $(ELIG, Sim_T (NetworkConnection, SlowNetworkConnection), VOIP)$. Such a semantic link requires a semantic refinement to be robust enough in order to be applied in a composition of Web services. In the first hand the description missing in NetworkConnection to be plugged in the input parameter SlowNetworkConnection is referred by the Extra Description i.e., $SlowNetworkConnection \setminus NetworkConnection$ i.e., $\forall netSpeed.Adsl1M$. In the other hand the common description defined by the conjunction of the output parameter of ELIG and the input parameter of VOIP is referred by the information required by $SlowNetworkConnection$ and effectively provided by $NetworkConnection$. In this way we remark that the intersection between the output parameter $NetworkConnection$ and the Extra Description $SlowNetworkConnection \setminus NetworkConnection$ i.e., $\forall netSpeed.Adsl1M$ is an Exact match with $SlowNetworkConnection$.

We illustrated the rationale of our approach by computing what is required in order to replace a non robust semantic link by its robust form. In particular, we are able to change an Intersection by a PlugIn match, and a Subsume by an Exact match in order to obtain robust semantic links (modelled with ✓ in Table 3). We could also consider other substitutions of matchmaking functions e.g., find a way to change a Subsume by a PlugIn match, or an Intersection by an Exact match and so on. However these substitutions are out of interest in Web service composition since

(i) some substitutions required the computation of $Out_s y \setminus In_s x$ (modelled with $\mathcal{K}_{i1}$ in Table 3) e.g., from PlugIn to Exact;
(ii) some others are not relevant because they implied a loss of matchmaking quality (modelled with $\mathcal{K}_{i4}$ in Table 3) e.g., from PlugIn to Subsume.

Table 3 summarizes these different levels of substitution. Suppose the substitution of a PlugIn by an Exact match in order to improve a semantic link valued by a PlugIn match level. The case under consideration is i) since we have to compute $B'$ such that $Out_s y \equiv B' \cap In_s x$. $B'$ is defined by $Out_s y \setminus In_s x$ to model the exact match. Unfortunately the description $B'$ can not be added to $In_s x$ since input parameters of services are supposed static without possible alteration. In the opposite output parameters of services may be enhanced by some Extra Descriptions ($Out_s y \cap B$) in order to be chained with input parameters of other services. Now
suppose the case ii) wherein the Subsume match is replaced by a PlugIn match. Consequently, the Extra Description $B'$ is computed as $B' \sqsubseteq B$ such that $B$ is defined by $In_{s_x} \setminus Out_{s_y}$. By the way $B' \cap Out_{s_y} \sqsubseteq In_{s_x}$ whereas $B \cap Out_{s_y} \equiv In_{s_x}$. It is obvious that $B$ is more appropriate than $B'$. The former enables an Exact match whereas the latter changes the Subsume by the PlugIn match.

In case some semantic links $\langle s_y, Sim_T(Out_{s_y}, In_{s_x}), s_x \rangle$ are not robust enough but valid, we are able to compute an Extra Description from $In_{s_x}$ in order to substitute the previous link by its robust form. In other words semantic links valued by a Subsumes or an Intersection match level move to robust semantic links in case their Extra Descriptions are provided. The latter description is essential to retrieve a robust service composition. The Extra Description returned by difference (2) is not only necessary to explain where a semantic link composition may fail but also why a semantic link failed and how to improve it. A composition failure is due to non robust semantic links since the matchmaking between Web services parameters is not robust enough.

### 4.6. Some Concluding Remarks

Till now, we have specified the formal context for semantic Web service composition at functional level. In this direction semantic links between Web services have been introduced, defined and illustrated from the motivating and industrial scenario (Section 3). The latter links are considered as a key concept to form compositions of Web services. Moreover we presented important criteria related to validity and robustness of semantic links. Valid semantic links refer to semantic links that can be useful for any Web service composition whereas robust semantic links are required to form robust Web service composition. In addition we addressed different solutions to overcome the problem of robustness in semantic links such as the Concept Abduction 61 or Concept Difference 62.

### 5. Semantic Link Matrix for Web Service Composability

In this section we address the problem of elaborating and computing a formal model with relevant and pre-computed semantic links (Requirement $R_8$). The idea consists in computing all valid semantic links from a set of services. The result of such a

<table>
<thead>
<tr>
<th>Substituted Match Type</th>
<th>Potential Substitute Match Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact</td>
<td>Exact</td>
</tr>
<tr>
<td>PlugIn</td>
<td>PlugIn</td>
</tr>
<tr>
<td>Subsume</td>
<td>Subsume</td>
</tr>
<tr>
<td>Intersection</td>
<td>Intersection</td>
</tr>
</tbody>
</table>

Table 3. Substitution of Match Levels for Web Service Composition.
computation will be stored in an innovative model so called Semantic Link Matrix.

### 5.1. Sequence Composability of Web Services

In a nutshell a semantic link is a very basic composition of two Web services since it links two Web services as a simple sequence. From this we can easily extend the basic composition to the trivial composition $s_x \circ s_y$ wherein $s_x \circ s_y$ (Figure 5) simply means that $s_y$ precedes $s_x$ and there exists a strictly positive value of $\text{Sim}_T$ between each input parameter of $s_x$ and some output parameters of $s_y$. Such simple compositions come from a special case of the sequence composability definition introduced by 18.

**Definition 5.1. (Sequence Composability)**

The sequence composability between two Web services $s_y$ and $s_x$ is defined as a composition $s_x \circ s_y$ if and only if an output of $s_y$ is exploited by an input of another Web service $s_x$.

![Fig. 5. Illustration of the Simplest Case of a Web Service Composition $s_x \circ s_y$.](image)

### 5.2. A Model for Composing Services: Motivation and Assumption

#### 5.2.1. Motivation

It is obvious that trivial compositions do not require any more complex models than semantic links to perform compositions of Web services. Indeed such a level of composition can be compared to a simple sequence of Web services wherein all input parameters of a Web service $s_x$ can be provided by some output parameters of the direct Web service predecessor $s_y$. Unfortunately Web service composition is more subtle than the trivial case of composition. In real scenarios a composition of Web services can be designed by means of simple sequences (Figure 5), but also more complex parallel branches, and non determinism choice (Requirement $R_2$) of Web services as well (Figure 6). Therefore a formal and appropriate model is required to model all valid semantic links we can potentially find in a composition. The model so called SLM (aka matrices of semantic connections) aims at easing the automated
process of Web service composition and improving its performance by storing and classifying all formal link called “valid semantic link” in a simple and intuitive way. In this direction SLMs are introduced in order to store not only basic \( s_x \circ s_y \) and trivial \( s_x \hat{o} s_y \) but also to easily retrieve more complex compositions of Web services. By classifying all valid semantic links the introduced formal model guarantees to discover all composition of Web services but also gives the opportunity to retrieve best compositions according to an optimization criteria i.e., value of valid semantic links. By introducing the SLM model we consider a simpler composition problem i.e., the semantic link composition. Therefore the Web service composition is mapped to a semantic link composition wherein semantic links inform about semantic connections between Web service. The solutions of Web service composition will be mainly oriented by the SLM of the domain.

5.2.2. Assumption: A Required Step of Web Service Discovery

The SLM, introduced in this section, aims at suggesting a model for composing a finite set of semantic Web services \( S_{W,s} \). To this end the latter set of services is supposed to be discovered in a relevant way, given a composition goal. In other words we assume that a process of discovery such as \(^{8,64}\) has been performed in order to first retrieve a finite set of Web services \( S_{W,s} \). Given this set of Web services, we aim at first computing a formal model to organize these Web services and then using this model to achieve a composition of a subset of the latter retrieved services.
5.3. Some Notations

Since the definition of SLMs requires some new but simple definitions i.e., \( \text{Out}(s_y) \), \( \text{In}(s_y) \), \( \text{Input}(S_{W_s}) \), \( \text{Output}(S_{W_s}) \) and \( \beta \), we suggest to introduce and illustrate them in the following. Suppose \( S_{W_s} \) be the set of Web services with the upcoming services \( s_x \) and \( s_y \). \( \text{Out}(s_y) \) refers to the set of output parameters of the Web services \( s_y \) whereas \( \text{In}(s_y) \) is its set of input parameters. \( \text{Input}(S_{W_s}) \) refers to the set of all input parameters of all services included in the set \( S_{W_s} \). In the same way \( \text{Output}(S_{W_s}) \) refers to the set of all output parameters of all services included in the set \( S_{W_s} \). In the rest of the paper \( \beta \) will refer to the composition goal. In our approach \( \beta \) is simply viewed as a subset of the TBox \( T \). These concepts have to be reached i.e., our ultimate issue is to find a composition of Web services which are able to find an instance of each concept in \( \beta \). \( \beta \) will be required and used during the AI planning-based composition (Section 6).

Example 5.1. \((\text{Out}(s_y), \text{In}(s_y), \text{Input}(S_{W_s}) \text{ and } \text{Output}(S_{W_s}))\)

Let the motivating example be in Section 3 and a set of relevant Web services \( S_{W_s} \) be defined by \texttt{AdslEligibility}, \texttt{VoiceOverIP}, \texttt{TvOverIP} and \texttt{LiveBox} services. Table 4 and Figure 7 illustrates the previous defined elements related to \( S_{W_s} \).

5.4. Semantic Link Matrix

In the following we define a SLM as a matrix containing all enabled, legal and valid transitions for a Web service composition goal. Non valid semantic links are disre-
Web Service Composition as a Composition of Valid and Robust Semantic Links

Table 4. Out(s_x), In(s_x), Input(S_W_s) and Output(S_W_s) in the Motivating Example.

<table>
<thead>
<tr>
<th>Services s_x</th>
<th>AdslEligibility S_a</th>
<th>VoiceOverIP S_b</th>
<th>TvOverIP S_c</th>
<th>LiveBox S_d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input(s_x)</td>
<td>{PhoneNum, ZipCode, Email}</td>
<td>{PhoneNum, SlowNetworkConnection}</td>
<td>{PhoneNum, FastNetworkConnection, IP Address, Decoder}</td>
<td></td>
</tr>
<tr>
<td>Output(s_x)</td>
<td>{Slow NC}</td>
<td>{Fast NC}</td>
<td>{VoIP Id}</td>
<td>{Invoice}</td>
</tr>
<tr>
<td>Input(S_W_s)</td>
<td>{PhoneNum, ZipCode, Email, SlowNetworkConnection, FastNetworkConnection, IP Address, Decoder}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output(S_W_s)</td>
<td>{Invoice, FastNetworkConnection, NetworkConnection, VoIP Id, VideoDecoder, SlowNetworkConnection}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

garded from a SLM point of view since a semantic link valued by a Disjoint match level (Table 2) refers to an inconsistency between two Web services. In contrary all valid semantic links between Web services are explicitly represented with a value pre-computed by means of the \( \text{Sim}_T \) function (Table 2). The latter value is based on the semantic quality of the valid semantic link. The more valid semantic links there are, the better it is for a functional composition problem.

**Definition 5.2. (Semantic Link Matrix SLM)**

The set of \( p \times q \) Semantic Link Matrices is defined as matrices \( M_{p,q}(P(S_W_s \times (0,1])) \). Their columns \( c_{j, \in \{1,...,q\}} \) are labelled by concepts in \( (\text{Input}(S_W_s) \cup \beta) \subseteq T \), i.e., the inputs parameters of services \( \text{Input}(S_W_s) \) in \( S_W_s \) and the concepts described by the goal set \( \beta \subseteq T \). Rows \( r_{i, \in \{1,...,p\}} \) are labelled by \( \text{Input}(S_W_s) \), the inputs parameters of services in \( S_W_s \). Each entry \( m_{i,j} \) of a SLM \( \mathcal{M} \) is defined as a set of pairs \( (s_y, \text{score}) \in S_W_s \times (0,1] \) such that

\[
(s_y, \text{score}) := (s_y, \text{Sim}_T(\text{Out} s_y, c_j)) \text{ if } s_y \in S_W_s, \text{Out} s_y \in \text{Out}(s_y) \tag{3}
\]

with \( r_i \in T \cap \text{In}(s_y) \subseteq \text{Input}(S_W_s) \) is the label of the \( i^{th} \) row,

with \( c_j \in T \cap (\text{Input}(S_W_s) \cup \beta) \) is the label of the \( j^{th} \) column.

A SLM is seen as a matrix with entries in \( P(S_W_s \times (0,1]) \). Each entry of a SLM refers to a set of pairs \( (s_y, \text{score}) \) such that the score refers to a semantic similarity \( \text{Sim}_T(\text{Out} s_y, c_j) \) between an output parameter \( \text{Out} s_y \in T \) of a service \( s_y \) and an input parameter \( c_j \in \text{Input}(S_W_s) \cup \beta \) of another service in \( S_W_s \). Indeed columns \( c_j \) of the SLM are labelled by all input parameters of the relevant Web services \( S_W_s \).

**Remark 5.1. (SLM or a Matrix of Valid Semantic Links)**

Since all entries of SLMs are defined on \( P(S_W_s \times (0,1]) \), the only conceivable values of semantic links in a SLM are defined in \( (0,1] \), hence only valid semantic link in a SLM.

\( P(S) \) refers to power set of \( S \).
Remark 5.2. (Key Feature of SLMs)
The innovative feature of SLMs is to label rows and columns together with the same set of concepts in $T$, i.e., input parameters of Web services in $S_{WS}$. The link between a row and column of such a matrix is defined by a possible semantic link between an output parameter of a service and a column of the matrix.

All SLMs in $M_{p,q}(P(S_{WS}\times(0,1]))$ of a given domain are defined by means of their number $p$ of rows and number $q$ of columns. SLMs of a domain are also related to the pre-defined goal $\beta$. By considering $\#(\beta)$ be the cardinality of goals we have the following relations:

$$p = \#(Input(S_{WS}))$$

$$q = p + \#(\beta) - \#(\beta \cap Input(S_{WS}))$$

Definition 5.3. (Dimension of SLMs)
In compliance with the dimension of a semantic link matrix in $M_{p,q}(P(S_{WS}\times(0,1]))$ is defined by:

$$dim_{p,q}(P(S_{WS}\times(0,1])) = p \times q.$$ 

In the general case, SLMs are not square matrices since $q > p$.

<table>
<thead>
<tr>
<th>index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_i$.label</td>
<td>Email</td>
<td>Decoder</td>
<td>FastNC</td>
<td>IPAddress</td>
<td>PhoneNum</td>
<td>SlowNC</td>
<td>ZipCode</td>
<td>Invoice</td>
</tr>
<tr>
<td>$c_j$.label</td>
<td>Email</td>
<td>Decoder</td>
<td>FastNC</td>
<td>IPAddress</td>
<td>PhoneNum</td>
<td>SlowNC</td>
<td>ZipCode</td>
<td>Invoice</td>
</tr>
</tbody>
</table>

Table 5. Labels of the Rows $r_i$ and Columns $c_j$ of the $7 \times 8$ Matrix $M$.

Example 5.2. (Illustration of SLMs indexes and labels)
Suppose the motivating example with the six Web services $AdslEligibility$ ($S_a$), $AdslEligibility-$ ($S_a^-$), $AdslEligibility+$ ($S_a^+$), $VoiceOverIP$ ($S_b$), $TvOverIP$ ($S_c$) and $LiveBox$ ($S_d$) services as elements of $S_{WS}$. $\{Invoice\}$ is supposed to be the only concept in goal $\beta$. The number of rows and columns are respectively equal to 7 and 8 (Table 5) according to equalities (4), (5) and definition of SLMs. Therefore rows and columns of the SLM $M$ of this domain are respectively indexed by $\{1, \ldots, 7\}$, $\{1, \ldots, 8\}$ and labelled by concepts $r_{i,j} \in \{1\ldots, 7\}$, $c_{j} \in \{1\ldots, 8\}$ of the TBox $T$. For instance the row $r_3$ and the column $c_1$ are labelled by the concept Email whereas the row $r_3$ and column $c_3$ are labelled by the concept FastNetworkConnection (Table 5).

5.5. Construction of Semantic Link Matrices
The Algorithm 1 presents the different steps of the SLM construction. Such a construction consists in discovering a semantic similarity score between the output
parameters of all services \( s_y \in S_{Ws} \) and the input parameters of another service in \( S_{Ws} \). In case the value \( score \) is not null, the pair \( (s_y, score) \) is added in the SLM. To do this, all input parameters of \( S_{Ws} \) are parsed two times \( (p \text{ times in line 4 and } q \text{ times in line 6}) \) to build the \( p \times q \) matrix. Moreover all output parameters of \( S_{Ws} \) are parsed \( (\text{line 8}) \) to value the matchmaking function \((\text{Table 2})\) of a potential valid semantic link.

**Algorithm 1:** Semantic Link Matrix Construction.

1. **Input:** \( S_{Ws}, T \).
2. **Result:** The Semantic link matrix \( M \) of the domain.
3. **begin**
   4. \textbf{foreach} row \( r_i \) of the SLM \( M \) do
   5. \quad \textbf{foreach} column \( c_j \) of the SLM \( M \) do
   6. \quad \quad \textbf{foreach} Web service \( s_y \in S_{Ws} \) do
   7. \quad \quad \quad \textbf{if} \( r_i \in In(s_y) \) then
   8. \quad \quad \quad \quad \textbf{if} \( \exists Out_{s_y} \in Out(s_y) \& Sim_T(Out_{s_y}, c_j) \neq 0 \) then
   9. \quad \quad \quad \quad \quad \textbf{m}_{r_i,c_j} \leftarrow \text{m}_{r_i,c_j} \cup (s_y, Sim_T(Out_{s_y}, c_j));
10. \quad \quad \textbf{return} \( M; \)
11. **end**

According to Algorithm 1 the Semantic link matrix construction is mainly function of the cardinality of \( Output(S_{Ws}) \) and \( Input(S_{Ws}) \). The algorithmic complexity of the SLM construction is then

\[
\theta(\#(Input(S_{Ws})) \times \#(Input(S_{Ws})) \times \#(S_{Ws}))
\]

so cubic in the worst case. However an optimal process of the SLM construction can be computed in

\[
\theta(\#(Input(S_{Ws})) \times \#(Output(S_{Ws})))
\]

or \( \theta(\max(\#(Input(S_{Ws})), \#(Output(S_{Ws})))^2) \)

so square in case \#\( S_{Ws} \ll \#(Input(S_{Ws})) \).

**Example 5.3. (Semantic link matrix illustration with Tables 4, 5)**

Suppose the motivating example with the set of Web services \( S_{Ws} \) in Table 4. We can easily compute the SLM of the domain according to algorithm 1 and Table 5 to obtain a SLM \( M \) with entries in \( \mathcal{P}(S_{Ws} \times \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}) \).

\[
M = \begin{pmatrix}
0 & 0 & \{(S_{s_y}^{-}, \frac{1}{4}), (S_{s_y}^{+}, \frac{1}{2}), (S_{s_y}^{+}, 1)\} & 0 & 0 & \{(S_{s_y}^{-}, \frac{1}{4}), (S_{s_y}^{+}, \frac{1}{2}), (S_{s_y}^{+}, 1)\} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \{(S_{s_y}^{-}, \frac{1}{4}), (S_{s_y}^{+}, \frac{1}{2}), (S_{s_y}^{+}, 1)\} & 0 & 0 & \{(S_{s_y}^{-}, \frac{1}{4}), (S_{s_y}^{+}, \frac{1}{2}), (S_{s_y}^{+}, 1)\} & 0 & 0 \\
0 & 0 & \{(S_{s_y}^{-}, \frac{1}{4}), (S_{s_y}^{+}, \frac{1}{2}), (S_{s_y}^{+}, 1)\} & 0 & 0 & 0 & 0 \\
0 & 0 & \{(S_{s_y}^{-}, \frac{1}{4}), (S_{s_y}^{+}, \frac{1}{2}), (S_{s_y}^{+}, 1)\} & 0 & 0 & 0 & 0 \\
0 & 0 & \{(S_{s_y}^{-}, \frac{1}{4}), (S_{s_y}^{+}, \frac{1}{2}), (S_{s_y}^{+}, 1)\} & 0 & 0 & \{(S_{s_y}^{-}, \frac{1}{4}), (S_{s_y}^{+}, \frac{1}{2}), (S_{s_y}^{+}, 1)\} & 0 & 0 \\
\end{pmatrix}
\]
The entry $m_{3,3}$ (i.e., $m_{\text{PhoneNumber,FastNC}}$) is equal to $\{(S_a, \frac{1}{2}), (S_a, \frac{1}{2}), (S_b, 1)\}$. Indeed a Web service $S_a$ with one input parameter $\text{PhoneNumber}$ (row 5) and an output $\text{NetworkConnection}$ semantically similar to $\text{FastNetworkConnection}$ (column 3) exists in $S_{Ws}$. $(S_a, \text{Sim}_T(\text{NetworkConnection, FastNetworkConnection}), S_b)$ is a valid semantic link since the matchmaking function valued by $\text{Sim}_T(\text{NetworkConnection, FastNetworkConnection})$ is a Subsume match level i.e., $\frac{1}{2}$.

**Proposition 5.1.** An entry $m_{i,j}$ of a semantic link matrix $M \in M_{p,q}(\mathcal{P}(S_{Ws} \times (0,1]))$ is different from the empty set if and only if the following conditions is satisfied:

- $\exists s_y \in S_{Ws}$ with at least one input $r_{i,\text{label}} \in T$ and one output $\text{Out}_s \in \text{Out}(s_y) \cap T$ such that $\text{Sim}_T(\text{Out}_s, c_j, \text{label}) \neq 0$ (SLM definition).

**5.6. Web Service Composition and SLMs**

As previously mentioned basic and trivial compositions of Web services are the first and the most intuitive categories of compositions. That is why it seems relevant to make sure that SLMs are able to simply retrieve these models of compositions. In other words a SLM of a given domain have to satisfy the sequence composability definition introduced in the beginning of this section.

**Theorem 5.1.** (Sequence Composability and SLMs)

Let $M$ be a SLM, and $s_x, s_y$ be two Web services in $S_{Ws}$. $s_x$ and $s_y$ are sequence-composable iff the following condition holds:

- $\exists i \in \{1, \ldots, p\}, \exists j \in \{1, \ldots, q\}, \exists v \in (0,1]$ such that $(s_y, v) \subseteq m_{i,j}$. $c_j, \text{label}$ and $r_{i,\text{label}}$ are respectively inputs of $s_x$ i.e., $\text{In}(s_x)$ and $s_y$ i.e., $\text{In}(s_y)$.

By the Sequence Composability theorem of SLMs we proved that SLMs are able to not only store valid semantic links between Web services but also sequence composable Web services. For further details the proof of the latter theorem is deeply studied in 18.

**Example 5.4.** (Sequence Composability in SLMs)

Let $M$ be the SLM. $S_b$ and $S_a$ are sequence-composable in $S_{Ws}$ iff $S_b \circ S_a$. Indeed there exists a pair $(i,j) = (5,6)$ in $M$ such that $(r_{5,\text{label}}, c_{6,\text{label}}) = (\text{PhoneNumber, SlowNC})$. In the considered case $(S_{a}^{-1}, 1) \subseteq m_{5,6}$ where $c_{6,\text{label}}$ is $\text{SlowNC} \in \text{In}(S_b) \subseteq T$ and $r_{5,\text{label}}$ is $\text{PhoneNumber} \in \text{In}(S_a) \subseteq T$. Therefore an output of $S_a$ is exploited by the input of $S_b$ since $\text{Sim}_T(\text{Out}_s, c_{j,\text{label}}) \neq 0$.

In case of more complex compositions, several services can be required to be chained with $s_x$ in order to produce all input parameters of $s_x$ e.g., Figure 6. So parallelism between some services needs to be considered. In case a service is in several entries of the same column, this means that parallelism can be required to compose with such a service.
Example 5.5. (Parallelism in SLMs)

Let $M$ be the SLM. The intersection of entries $m_{3,2}$ and $m_{5,2}$ is $\{(S_c, 3/4)\}$. This simply means that Web service $S_c$ requires two input parameters $r_{3,\text{label}}$ i.e., $\text{FastNC}$ and $r_{5,\text{label}}$ i.e., $\text{PhoneNum}$ to be achieved. In other words a composition that requires the Web service $S_c$ have to provide two output parameters which will be exploited by the input parameters of $S_c$.

Besides to model sequence composable and parallelism in Web service composition, SLM is also able to model the multiple choice. This case of non determinist compositions are conceivable in case an the number of elements in an entry of a SLM is strictly greater than one.

Example 5.6. (Choice in SLMs)

Let $M$ be the SLM. The entry $m_{1,3}$ of $M$ is $\{(S_a, 1/2), (S_a, 1/2), (S_a^+, 1)\}$. The latter set identifies that three Web services $S_a$, $S_a^-$ and $S_a^+$ are able to provide an output parameter semantically close to the concept $\text{FastNC}$ i.e., $c_3,\text{label}$. In other words a composition that requires the input parameter $\text{FastNC}$ may choose between three different Web services.

According to the previous features of SLM, sequence, choice (aka Non Determinism) and parallelism (aka Concurrency) of Web services can be retrieved and then used to model compositions returned by our approach (Requirement $R2$).

5.7. Semantic Link Matrix and Scalability

In this section the SLM model and its construction have been introduced in order to prepare a FLC of Web services. The key contribution of the SLM is a formal and semantic model to control a set of Web services $S_{Ws}$ which is relevant for a composition goal. Since the flexibility feature of Web service composition model is supposed to be fundamental in volatile environment such as the Web services area (Requirement $R_6$), the SLM model aims at being as flexible as possible. Indeed dynamic process of Web service discovery could be applied to modify a SLM of a given domain. In this direction the SLM model is quite appropriate to support fundamental criteria such as alteration and modification (e.g., insertion, deletion, or update) of Web services in $S_{Ws}$, hence a flexibility of the model. Each new update of $S_{Ws}$ is supported by a SLM revision together with a revision of the domain ontology $T$. For instance the integration of a new Web service is related to the insertion of new labelled rows and columns in the worst case. In the alternative case the integration of a Web service means a simple insertion of this service in the relevant entry(ies) of the specific SLM. Therefore incremental systems wherein new Web services are progressively added, are supported by the SLM model. In the same way deletion of Web services in a given SLM can be easily performed as well.

The set of Web services $S_{Ws}$ involved in the SLM is closed in order to limit its dimension. This assumption seems appropriate to perform Web service composition with a bounded number of Web services such as presented in Section 3. Even if...
the SLM dimension and its computation is only square with the number of input parameters of $S_{WS}$, such a level of complexity may be a real issue to scale with more complex cases of composition e.g., hundreds of thousands of Web services involved in a composition. Computing the SLM in such a domain can really weaken the model used by our composition approach, presented in the next section. One possible direction to overcome this issue is to consider SLMs with a major part of empty entries since SLMs are sparse in most of cases. In such a case the SLM model is replaced by a more subtle model i.e., lists of semantic links wherein we store only valid semantic links. The space complexity of this model is then drastically reduced. However our composition approach presented in Section 6 will require some minor changes since our composition method assumes a SLM of the domain. We remark that scenarios that require a composition of hundreds of thousands of Web services is more than improbable in real industrial scenario since its execution will be a real open issue.

5.8. Some Concluding Remarks

The required background for Web service composition at functional level i.e., valid semantic links and their semantic dependency can be known by means of SLMs (Requirement $R_8$) of the domain. In this way the SLM model can be used to pre-chain Web services according to the latter semantic similarity. Therefore SLMs describe all valued possible interactions between all relevant Web services in $S_{WS}$. This valuation can be used as an interesting feature to classify valid semantic link and retrieve the best links in a possible composition. Since a composition of Web services at functional level mainly consisted of semantic links we suggest to study Web service composition as a semantic links composition. The suggested approach to retrieve composition is mainly oriented by i) the SLM of a given domain, ii) its valid semantic links together with their values since all Web services are semantically well ordered in the latter robust and formal model i.e., SLM.

6. Regression-based Web Service Composition

From a SLM of a given domain with its valid semantic links, together with basic AI planning techniques such as regression, or progression-based search we retrieve complete, correct, consistent and robust plans as solutions of a composition problem. The method consists in retrieving a (or some) composition(s) of services that produces instances of the desired concepts in $\beta \subseteq T$ depending on some individuals in an ABox $A$ (e.g., Figure 8) of the ontology $T$. More formally the planning problem is formalized as a triple $\Pi = \langle S_{WS}, A, \beta \rangle$ so called the AI planning-based Web service composition problem. Our AI planning-based semantic Web service composition is defined by means of a set of Web services $S_{WS}$ which refers to planning operators (i.e., available actions). The main differences with AI planning problem concern first the description of goals and Initial states, second addition
of assumptions on the planning operators, the composition goal and initial conditions. The latter assumptions are required to stage with open issues related to the AI planning-based Web service composition. In this direction the set of Web services $S_{ws}$ is closed by assumption. Whereas goals are described as fluent with a first-order logic representation in AI planning domain, the goal $\beta$ is clearly described as a set of defined concepts in a TBox. $\beta$ informs about composition (or plan) directions in order to retrieve some individuals of concepts in $\beta$. The DL-based description of $\beta$ allows to make reasoning such as satisfiability, subsumption, abduction, and concept difference especially to infer some properties between concepts in $\beta$ and parameters of Web services in $T$. The Initial state of our problem is modelled as a set of instances in an ABox $A$. In other words $A$ status on initial conditions by defining some instances of concepts we can use to instantiate input parameters of some Web services.

From this a composition of Web services is computing in a well-defined domain: goals are explicitly given, initial state is well defined and Web services are strictly defined at functional level. Therefore non determinism, implicit goals, fuzzy Web service descriptions and behaviours are not considered in our composition approach. It does seem possible to directly apply (some) current AI planning methods to our specific problem.

![Email, PhoneNum, ZipCode]

Fig. 8. Sample of the Assertional Box of the $\text{ALE}$ Domain Ontology $T$.

### 6.1. Composition as a Partial Ordering of Web Services

Before mapping a Web service composition as a simpler form of an AI planning problem, we suggest to briefly overview the way we model a composition of Web services. Considering $S_{ws}$ be the set of available and relevant services, the composition result is a partial ordering ($\circ$, $S_{ws}$) of services in $S_{ws}$ arranged in a simpler workflow (mainly sequence, parallel branch and non determinism operators) in order to fulfill the composition goal $\beta$. The latter ordering of Web services is intertwined with valid semantic links. A composition is then defined as a plan of services wherein all services are semantically well ordered and well linked by semantic links.

**Example 6.1.** (Semantic links and Ordering on Web Services)

Suppose the two Web services $S_a$ and $S_b$ in the motivating example. Since $S_b$ and $S_a$ are sequence composable i.e., $S_b \circ S_a$, $S_a$ required to be executed first in order to provide the input parameter required by $S_b$. 

The partial ordering \((\circ, S_{W_s})\) returned by our FLC approach consisted of Web services with either a (or some) predecessor(s), either successor(s), both or none. A Web service \(s_x\) is a predecessor of another Web service \(s_y\) i.e., \(s_y \circ s_x\) in case \(s_x\) provides at least one output parameter to \(s_y\). Reciprocally \(s_y\) is a successor of \(s_x\).

Since the order is not necessarily a total order some Web services in \((\circ, S_{W_s})\) can be incomparable i.e., two Web services can not be ordered since none of both requires the achievement of the other service. Parallel branches of Web services \(s_x\) and \(s_y\) are then conceivable in service composition in case \(s_x\) and \(s_y\) are incomparable and they provide output parameters required by inputs of another service. Therefore the partial ordering of Web services models not only sequences of comparable Web services but also the concurrent execution of incomparable Web services. The following definition is introduced to model a partial order of Web services by means of sequence, multiple choice and parallelism.

**Definition 6.1.** Let \(<S_{W_s}, A, \beta>\) be the AI planning-based Web service composition problem, constructs of Web service composition and their priority order are \(\wedge > \lor > \circ\), such that:

- \(\wedge\) is the conjunction operator i.e., the parallel construct;
- \(\lor\) is the disjunction operator i.e., the multiple choice construct;
- \(s_i \circ s_j\) if \(\exists C_o, C_i \in T\) such that \(<s_j, Sim_T(C_o, C_i), s_i>\) is a valid semantic link i.e., the sequence construct which models sequence-composability.

**Example 6.2.** (Semantic Links and Partial Ordering on Web Services)
Let the three Web services \(S_{y_1}, S_{y_2}, S_{y_3}\) and \(S_x\) illustrated in Figure 6, the partial order \((\circ, \{S_{y_1}, S_{y_2}, S_{y_3}, S_x\})\) is defined by (i) \(S_x \circ S_{y_1}\); and (ii) \(S_x \circ (S_{y_2} \lor S_{y_3})\). The reduced form is \(S_x \circ (S_{y_1} \land (S_{y_2} \lor S_{y_3})))\).

### 6.2. A Regression-based Approach for Web Service Composition

The introduced composition process \(Ra_4C\) (Regression-based Approach for Composition) presented by algorithm 2 follows a recursive and regression-based approach from the set of concepts (without individuals in \(A\)) in \(\beta\) to initial concepts \(A\). The main idea consists in controlling and parsing the SLM of a given domain in an adequate way to obtain a composition of services and their semantic links that will satisfy the goal composition i.e., that retrieves individuals of all concepts in \(\beta\).

The \(Ra_4C\) approach requires a first call of the algorithm 2 i.e., \(Ra_4C(M, \emptyset, (S_{W_s}, A, \beta), \emptyset)\) in order to compute automated Web service composition that provide individuals of the concept \(\beta\). In the trivial case (line 6) wherein all concepts in \(\beta\) have already individuals in \(A\), the goal \(\beta\) is obviously fulfilled by the initial condition. The automated process of composition is then stopped since trivially satisfied by \(A\). In the more complex case (line 8) wherein the ABox \(A\) does not contains any individuals of the concept \(\beta_i\) in \(\beta\), the composition process needs to retrieve at least one Web service \(s_x\) in \(S_{W_s}\) with \(\beta_i\) defined as one of its output parameters. In other words a Web service discovery process eased by the SLM of
Web Service Composition as a Composition of Valid and Robust Semantic Links

Algorithm 2: Regression-based Approach for Composition $Ra_4C$.

1. **Input:** A SLM $M([m_{i,j}])$, a (or disjunction of) plan(s) $\pi$, a planning prb $(S_{Ws}, A, \beta)$, a set of solved goals $G$, a set of non valid goals $\beta_{nv}$.
2. **Result:** A disjunction of consistent plans $\pi$.

```
begin
    $S_c \leftarrow \emptyset$; // Temporary set of pairs in $S_{Ws} \times (0,1]$.
    // Stop condition of the $Ra_4C$ algorithm.
    if $\exists c_k \in A$ such that (($c_k$ is an individual of $C_k$) & (Sim$_T(C_k, \beta) \neq 0$)) then $\pi \leftarrow \beta$;
    // Web services discovery with $\beta$ output.
    foreach $I_i \in Input(S_{Ws})$ do
        if $\exists (s_y, v) \in m_{I_i, \beta}$ then Add($(s_y, v), S_c$);
    // Plan for Web service composition.
    if $S_c \neq \emptyset$ then
        foreach pair $(s_y, v) \in S_c$ such that $s_y \in S_{Ws}$ do
            $\pi \leftarrow \pi \lor s_y$;
            foreach $In(s_y) \in In(s_y)$ do
                if $\beta \in G$ then
                    $\pi \leftarrow \pi \land \emptyset$; // Inconsistent plan
                    Add($\beta, \beta_{nv}$);
                else
                    Add($\beta, G$);
                    $\Pi \leftarrow \langle S_{Ws}, A, In(s_y) \rangle$;
                    $\pi \leftarrow \pi \circ (\Lambda_{In(s_y)}Ra_4C(M, \pi, \Pi, G))$;
                end
            end
        end
    else $\pi \leftarrow \pi \land \emptyset$; // Non correct plan since there is an open goal (input).
    return $\pi$;
end
```

The domain is performed. In case of a discovery success (line 11), the process is iterated with the $s_y$ input parameters as new goals (line 16) of AI planning-based Web service composition. Alternatively (line 22), the process is stopped and the (or a part of the) plan is reduced to an incorrect composition $\emptyset$ since the current goal is considered as open i.e., no composition or individuals in $A$ can solve the current goal. All the previous process is recursive (line 21) until the concepts in $\beta$ and new goals (recursive goals as input parameters of services involved in the composition) have individuals in $A$ (stop condition in line 6). Therefore the algorithm 2 returns a disjunction of consistent plans consisted of valid and sequence-composable semantic links i.e., a (or more) composition(s) of Web services semantically chained by semantic links. For further details the correctness proof of algorithm 2 is detailed...
Formally the computational complexity of the $Ra_4C$ process is polynomial with the number of Web services and their numbers of functional (i.e., here input since we perform a regression-based search) parameters. The latter complexity is obviously related and very depending on the filling rate of the SLM. It is obvious that the sparser the SLM the faster the $Ra_4C$ process is.

In the $Ra_4C$ process we assumed without loss of generality, and for sake of simplicity that $\beta$ refers the unique concept we want to instantiate, and not as a set of concepts. Even if we do not investigated on a subtle method to perform a composition with multi goals, the practical computation of a composition with $n$ concepts in $\beta$ is linear with $n$. However it could be even less in general case e.g., some goals can be satisfied by a composition of services satisfying another goal.

6.3. Main Features of Retrieved Web Service Composition

Depending on a set of services $S_{WS}$, a set of concepts $\beta$ as a goal and an ABox $A$ of a given domain, the set of retrieved compositions has some interesting properties related to the AI planning research area i.e., consistency, completeness and correctness.

6.3.1. Consistency of Web Service Composition

The consistency property in Web service composition is a necessary condition to obtain executable solutions. This condition is satisfied by compositions containing no cycle in the ordering constraints and no semantic link conflicts. Such compositions are retrieved by the $Ra_4C$ algorithm by identifying cycles and conflicts to dispose of inconsistent semantic links. This inconsistency is tackled by a simple update of solved goals (line 19). Therefore a goal is not resolved twice or more during the composition process.

Example 6.3. (Set of consistent compositions)

Let $\mathcal{M}$ be the SLM and $\Pi = \langle \{S_a^-, S_n, S_b, S_c, S_d\}, A, \{\text{Invoice}\} \rangle$ be the planning-oriented service composition problem. By considering $A$ illustrated in Figure 8, the composition goal is to obtain an Invoice of a customized service. From this the composition result of $Ra_4C$ is a disjunction of nine consistent compositions (or simpler plans) depicted in Figure 9:

\[
\pi_{S_a, S_b} := S_d \circ \left[ \begin{array}{ll}
\text{PhoneNum} \\
\land (S_c \circ (S_a((\text{Email}, \text{PhoneNum}, \text{ZipCode}) \land \text{PhoneNum}))) \\
\land (S_b \circ (S_g((\text{Email}, \text{PhoneNum}, \text{ZipCode}) \land \text{PhoneNum})))
\end{array} \right]
\]

In a previous step the end user has selected a set of offers (i.e., services) she want to subscribe, together with some information such as her email address, Phone number and the ZipCode of the desired phone line.
wherein $S_x$ and $S_y$ can be any Web service in $\{S_a, S_{-a}, S_{+a}\}$.

![Diagram showing Web service compositions and semantic links]

**Fig. 9. Ra4C Result on the Motivating example.**

### 6.3.2. Correctness of Web Service Composition

Adapted from the AI planning domain, we define a correct composition as a composition wherein every input of every Web service can be provided by an output parameter of another Web service or by an individual occurring in the ABox $\mathcal{A}$. In other words correctness of resulting compositions is guaranteed in case they do not contain any open inputs. In our approach, all non correct compositions are identified in the Ra4C process in line 22 of algorithm 2. Therefore compositions with open inputs are removed from the set of potential solutions.

**Example 6.4. (Set of correct compositions)**

The nine composition results returned by algorithm 2 are correct since none of them contains any open goals.

### 6.3.3. Completeness of Web Service Composition

Here we study the completeness properties of compositions computed by algorithm 2. By definition, a SLM contains all required information about complete plans since a SLM explicitly stores all valid semantic links between sequence composable services. According to the Ra4C process the compositions refinement follows a backward chaining strategy from the goal $\beta$ to initial states $\mathcal{A}$ by means of a domain SLM and its semantic links. Therefore by definition of SLM, all compositions of services consist of valid semantic links $(s_y, Sim_T(Out_{s_y}, \beta), s_x)$ hence complete compositions.
Example 6.5. (Set of complete compositions)
The set of compositions returned by $Ra_4C$ is complete by definition.

By means of the $Ra_4C$ process a set of correct, complete and consistent compositions is returned.

6.4. Robust Semantic Web Service Composition

Even if $Ra_4C$ is able to compute correct, complete and consistent compositions of Web services, some of them can be not robust enough. Indeed some compositions returned by $Ra_4C$ can contain non robust semantic links (Definition 4.3) hence non robust compositions (Definition 4.4).

Example 6.6. (Non Robust Composition of Web Services)

Suppose $\pi_{S_a,S_a}$ (from $\pi_{S_x,S_y}$ in Example 6.3) illustrated in Figure 10 be a composition result of the motivating example. Its formalization is as following:

$$\pi_{S_a,S_a} := S_d \circ \left[ \text{PhoneNum} \right.$$  

$$\land (S_c \circ (S_a(\text{Email}, \text{PhoneNum}, \text{ZipCode}) \land \text{PhoneNum}))$$  

$$\land (S_b \circ (S_a(\text{Email}, \text{PhoneNum}, \text{ZipCode}) \land \text{PhoneNum})) \right]$$

It is obvious that such a composition of Web services is not robust.

In the considered case a complete automation of semantic Web service composition is still not a reality, especially when a Web service composition comprises non robust semantic links. An intuitive but naive method would be to not consider non robust semantic links i.e., semantic links valued by a Intersection and Subsume match level. Therefore the composition approach would be the same as proposed.

![Fig. 10. A Non Robust Composition of the Motivating example.](image-url)
by the $RaC$ process with a restricted SLM in $M_{p,q}(P(S_{Ws} \times \{1, \frac{3}{4}\}))$. It is obvious that such a method is far from convenient since it would consider no more than two matchmaking levels to value semantic links, hence a loss of expressivity in semantic links. Another approach consists in replacing non robust semantic links of a composition with their robust forms as suggested in Section 4.5. The Web service composition process is still automatic in case the Extra Description required by the non robust semantic links is automatically retrieved. In the following we study two main methods to obtain this Extra Description i.e., the first performs robust Web service composition in an automated way whereas the second acts in a semi-automated way.

### 6.4.1. The Perfect Case

An intuitive method to immediately retrieve the Extra Description consists in discovering services that return this description by means of their output parameters. In case a non robust semantic link is retrieved in a composition, the Extra Description of the link is computed according to Concept Difference (or Concept Abduction). The Extra Description is then exposed to a Web service discovery process which is in charge of retrieving relevant Web services. The discovered services will be able to provide the Extra Description by means of their output parameters. It is obvious that the Extra Description can be reached by one or a conjunction of Web services, depending on the Extra Description and the discovery process. However the main constraint of this method is related to the computational complexity of composition. Indeed each input parameter of new discovered Web services has to be either known at run time or linked to an output parameter of another Web service through a robust semantic link. The latter consideration requires that all input parameters are then considered as a new composition goal. The more non robust semantic links in a composition the more important the cardinality of services will be implied in this composition. Such a solution can be employed and implemented in any composition approach.

### 6.4.2. An Alternative: Relaxing Constraints

In case wherein no service can reach the Extra Description there is no way to automatically retrieve this description. In this direction all available information does not guarantee to find a robust Web service composition. Consequently, the latter description has to be retrieved by relaxing some constraints during the composition process. Indeed relaxing some constraints and obtaining a composition of robust semantic links is an interesting trade-off to reach composition. These constraints still guarantee the original feasible solutions and yield additional feasible solutions. Constraints $B_{i,1 \leq i \leq n}$ in Web service composition refer to the Extra Description. More formally the set of relaxing constraints $B$ is expressed by Definition 6.2 where $(s_y, SimT(Out, s_y, In, s_x), s_x)$ refers to semantic links in the composition model.
Definition 6.2. (Set of Relaxing Constraints)
The set of relaxing constraints $\mathcal{B}$ is defined by

$$\inf \left\{ \text{In}_s \backslash \text{Out}_s \mid \langle s_y, \text{Sim}_T(\text{Out}_s, \text{In}_s), s_x \rangle \text{ is a valid semantic link} \right\} \setminus \{\top\} \quad (10)$$

Intuitively, the set of relaxing constraints $\mathcal{B}$ of a Web service composition is defined as being the set of descriptions able to change non robust semantic links into their robust forms. $\mathcal{B}$ gathers the most specific descriptions of the set $\{\text{In}_s \backslash \text{Out}_s\}$ where $\langle s_y, \text{Sim}_T(\text{Out}_s, \text{In}_s), s_x \rangle$ is a valid semantic link. The latter consideration implies that a same description (i.e., the most specific) can be used by a finite set of non robust semantic links to change them in their robust forms. The descriptions used to perform these changes will be the most specific descriptions of the Extra Description. $\mathcal{B}$ does not only explain why the composition process failed but also gives a solution of the robustness problem of semantic links hence a way to reach robust Web service composition.

Proposition 6.1. (Constraints for Robust Semantic Links)
The set of relaxing constraints $\mathcal{B}$ of a Web service composition with only robust semantic links is the empty set.

Proof. Let $\pi$ be a service composition constituted of only robust semantic links $sl_{i \leq i \leq n}$ defined by $\langle s_y, \text{Sim}_T(\text{Out}_s, \text{In}_s), s_x \rangle$. By definition, the match level between $\text{Out}_s$ and $\text{In}_s$ is either Exact or PlugIn, hence $\text{Out}_s \equiv \text{In}_s$ or $\text{Out}_s \equiv \text{In}_s'$. By difference (2) we obtain in the two cases that $\text{In}_s \backslash \text{Out}_s \equiv \top$ i.e., $\mathcal{B}$ is defined by the empty set.

Once the set of Extra Descriptions is retrieved through Concept Difference, the set of relaxing constraints $\mathcal{B}$ (Definition 6.2) is computed to be suggested to the end user in order to be relaxed. This user is then responsible of providing the Extra Description required by the system in order to elaborate the final and robust Web service composition, hence satisfying the initial user request. The suggested method has the advantage of relaxing constraints on the end user side. In the motivating example, Web services and user’s requirements are both not specified enough to focus on the Extra Description and also advantages of relaxing constraints. The relaxing task is of the utmost importance in real scenarios of composition since Web service composition often requires some refinements such as relaxing constraints to turn in an automated composition.

Example 6.7. (Relaxed Semantic links)
The motivating example exposes a Web service composition through 4 semantic links $sl_{i \leq i \leq 4}$ (Figure 10). Three of the four valid semantic links are not robust i.e., $sl_{i \leq i \leq 3}$ are valued by a Subsume match and $sl_3$ is valued by an Intersection match. No Web service may provide the Extra Description necessary to form
robust semantic links. A Relaxing constraints needs to be applied to obtain a composition of robust semantic links. The discovery of the Extra Descriptions $B$ gives directions to obtain robust semantic links. According to the Definition 6.2, $B$ is constituted of an union of three differences in DL i.e., the difference between the concepts (i) FastNetworkConnection and NetworkConnection to change $sl_1$ by a semantic link valued by an Exact match; (ii) SlowNetworkConnection and NetworkConnection to change $sl_2$ in the same way as $sl_1$; (iii) IPAddress and VoIPId to replace $sl_3$ by a semantic link valued by a PlugIn match. Since $sl_4$ is a robust semantic link and $\forall netSpeed.AdslMax \sqsubseteq \forall netSpeed.Adsl1M$, $B$ is defined by $\{\forall netSpeed.AdslMax, \forall protocol.IP\}$.

The Web service composition can be automatically retrieved in case the Extra Description is provided by the end user, other services or any third party, depending on the application we want to automate. For instance $\forall netSpeed.AdslMax, \forall protocol.IP$ can be provided by another Web service in case we want to automate the composition process of the motivating example.

6.4.3. An Approach for Robust Web Service Composition

The composition approach used and extended in this paper is based on the model $Ra_4C$ introduced earlier in the paper, but can be easily applied with many other approaches of functional level composition e.g., $^{31,29,4}$. From some user constraints, a set of Web services and a goal to achieve the $Ra_4C$ approach computes a composition consisted of valid but not necessarily robust semantic links. Indeed some solution proposals may refer to non robust compositions since some semantic links can be valued by a Subsume match level. It is obvious that this approach and most of the functional level composition methods needs refinements to perform robust Web service composition. The algorithm suggested in this section aims at extending not only the $Ra_4C$ approach but also any other functional level composition method in order to overcome the robustness problem in Web service composition. The main idea is as following: the set of relaxing constraints $B$ is progressively evaluated throughout the computation of composition by $Ra_4C$. This method is suitable especially to compare different descriptions and then rapidly prune the worst solutions. Algorithm 3 differs from $Ra_4C$ and other composition approaches, primarily because it does explore non robust semantic links and stop a composition process in case its Extra Description is more specific than one of the pre-computed solutions. The more specific Extra Description the more description have to be provided to enable automation of the composition. That is why the best Web service compositions are supposed to be compositions with the most general Extra Descriptions, i.e., with the least constraints.

The algorithm 3 consists of the following steps. First (line 4 and 5) in the trivial case wherein the algorithm $Ra_4C$ returns a robust composition, the process is stopped. In the more complex case (from line 7), each composition result (line 9) is analysed by algorithm 3. During the computation of each composition result
πᵢ (line 9), the Extra Description of each non robust semantic link involved in πᵢ is computed. By assuming B be the Extra Description of the best current robust composition, we compare the current Extra Description Bᵢ to the best current Extra Description B (lines 12 and 13). In case Bᵢ (line 12) is more specific than B, this means that B is the most appropriate Extra Description. The more the composition process proceeds the more specific will be Bᵢ. Indeed more non robust semantic links can be involved in the composition πᵢ. That is why we stop the process of computing the Extra Description of πᵢ. Otherwise in case Bᵢ is more general than B (line 13) we continue the process of valuating the Extra Description of Bᵢ. At the end of the algorithm 3 (line 15), the best robust Web service composition together with its Extra Description are returned. Roughly speaking the set Bᵢ required by non robust semantic links of the composition π is returned to the end user.

Algorithm 3: Robust Web Service Composition.

1. **Input:** A composition process Ra₄C.
2. **Result:** The best compositions and their Extra Descriptions.
3. **begin**
   4. if Ra₄C returns a robust composition πᵢ then
      5. return {(πᵢ, ∅)};
   6. else
      7. Bₐ₀ ← compute Extra Description of π₀;
      8. sol ← {(π₀, Bₐ₀)};
      9. foreach πᵢ₀ ≠ 0 do
         10. while πᵢ computation is in progress by Ra₄C do
            11. Bᵢ ← current Extra Description of πᵢ;
            12. if Bᵢ ⊆ sol.B then stop πᵢ computation;
            13. if Bᵢ ⊇ sol.B then continue to build πᵢ;
            14. if πᵢ is valid then sol ← sol ⊔ {(πᵢ, Bᵢ)};
      15. return sol;
   16. **end**

According to the set of available Web services i.e., commercial offers pre-selected by the end user, all semantic links are computed first and then Concept Difference reasoning is applied to non robust semantic links. Therefore the computational complexity of the introduced method is the same as approaches without relaxation (e.g., Ra₄C) since the Concepts Differences are solved in a pre-processing phase for each non robust semantic links we consider in the composition. However the computational complexity of the pre-processing step is mainly depending on the match levels of non robust semantic links, and particularly on the computational complexity of all Concepts Differences.

**Example 6.8. (Computing the Best Robust Composition)**

According to the algorithms 2 and 3, the different Extra Description of the nine
compositions $\mathcal{B}_{\pi S_x, S_y}$ can be computed with $S_x, S_y \in \{S^-_a, S_a, S^+_a\}$. The results are depicted in Table 6. According to this Table, it is obvious that the two best robust compositions are $\pi S^+_a, S^+_a$ and $\pi S^-_a, S^-_a$ since they have the most general Extra Description. However the latter compositions are not robust enough and require some Extra Description $\{\forall \text{protocol.IP}\}$.

### 6.5. Some Limitations of our Composition Approach

In computing the SLM, a single output from one service can be related to only a single input to a possible following service (see Definition of semantic link). In our approach this is a necessary restriction since it is SLM oriented. It appears that our use of DL for describing inputs and outputs would allow generalizations of inputs and outputs to be matched. For instance a conjunction of output parameters from different Web services could be semantically matched to one (or more) input parameter(s) of a (or more) services. To this end the semantic link definition requires to be extended. Even if such an extension is straightforward, its SLM adaptation is more complex and even inappropriate for Web service composition. Indeed considering such an extension of semantic links does not longer require a simple matrix but, obviously, a more complex model.

Scalability of Web service composition and discovery models is still an open issue. However the formal model i.e., SLM together with $Ra_4C$ scale well in France Telecom scenarios. Note that the assumption related to Requirement $R_5$ (persistance of information) does not hold in this approach since first we retrieve potential and conditional composition of services and then we execute the composite service.

Another weakness of the latter approach may concern the potential involvement of the end-user during the computation of robust Web service compositions hence a supervision of the robust composition process. The result of the $Ra_4C$ process and
its robust form is a set of service compositions we can model as a partial order of services interleaved by semantic links.

6.6. Some Concluding Remarks
In this section we presented the Ra$_4$C process, which is able to perform correct, consistent and complete services composition by means of a SLM and regression-based approach. Instead a regression-based approach, other problem-solving techniques - called heuristic reasoning - may be applied such as a progression-based approach.

Given a SLM of a domain, most of its entries are not necessarily required to perform “only one” composition goal. However, given this same domain, its SLM can be re-used together with an AI planning approach without new pre-computation of semantic links to achieve more composition goals. Actually only the goal columns of the SLM are required to be updated. This obviously improves the performance of the composition approach in case more than one composition goal requires to be achieved in a domain. In case two composition goals are given on the fly with two different domains, the composition approach is the same but the SLM requires to be updated (Requirement $R_6$ - very simple in most of cases, see Section 5) with the relevant Web services.

Besides an automated method for Web service composition we overcome the problem of robustness in service composition by means of the algorithm 3. Contrary to Ra$_4$C which does not consider non robust semantic links as a special case of semantic links, we have considered a method to obtain more robust compositions of Web services on the fly i.e., throughout the computation of potential (and non robust) compositions (e.g., Ra$_4$C process).

From the definition of robust and valid semantic links, it seems interesting to retrieve optimal compositions in terms of the quality of semantic links involved (Requirement $R_7$).

7. The Composition Tool: Implementation and Experiments
In this section, we discuss the prototype tool that we developed to compute automated semantic Web service composition in our framework. Moreover we give a preliminary evaluation of the suggested approach by analyzing some results obtained with the prototype developed.

7.1. Architecture and Implementation
Figure 12 shows the high level architecture wherein we implemented and tested our FLC component i.e., the SLM and the Ra$_4$C approach. In the following we focus on this latter components. Supposing the latter architecture we start from a repository of Web services $S^*_W$, that implements our different scenarios, and which can be seen, therefore, as an advanced version of UDDI. In the suggested framework each entry
Web Service Composition as a Composition of Valid and Robust Semantic Links

wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace{ _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/ontologies/internetPack#",
  dc _"http://purl.org/dc/elements/1.1#",
  foaf _"http://xmlns.com/foaf/0.1/",
  wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
  loc _"http://www.wsmo.org/ontologies/location#"
}

/************* Sa ONTOLOGY ***************/
ontology _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/ontologies/Sa_Ontology"

  dc#title hasValue "WSML ontology of Sa i.e., AdslEligibility"
  dc#subject hasValue "InternetPackage"
  dc#description hasValue "Fragments of the InternetPackage Ontology"
  dc#contributor hasValue
    { _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/~fael8534/foaf.rdf"
  dc#date hasValue _date(2007,08,28)
  dc#format hasValue "text/html"
  dc#language hasValue "en-US"
  dc#rights hasValue _"https://sws-orangeLabs.elibel.tm.fr/privacy.html"
  wsml#version hasValue "$Revision: 1.1 $"

concept ZipCode
concept Email
concept PhoneNum
concept NetworkConnection
  netSpeed ofType Speed
  nonFunctionalProperties
    dc#description hasValue "concept of a NetworkConnection"
endNonFunctionalProperties

/************** WEB SERVICE ***************/
WebService _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/services/Sa"
importsOntology
  _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/ontologies/IPDO.wsml"
capability _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/Eligibility#SaCap"

//Preconditions in WSMO refer to Input parameters in the Web Service Specification
precondition definedBy
  ?pn memberOf PhoneNum
  and
  ?zc memberOf ZipCode
  and
  ?em memberOf Email.

//PostConditions in WSMO refer to Output parameters in the Web Service Specification
postcondition definedBy
  ?nc memberOf NetworkConnection.

Fig. 11. WSML Description of the AdslEligibility Service $S_a$.

of the UDDI registry represents a service in term of both its syntactic interface through a WSDL document, and its semantic description, which can be expressed
in any language that allows to express semantics. We recall that in our framework the focus is on semantics that a service can express. As an example, the functional description (through its WSML interface) of service AdslEligibility $S_a$ can be consulted online on https://sws-orangelabs.elibel.tm.fr/media/2/20071001-Sa.wsml (Figure 11). The client specifies her goal service $s_g$ in term of a pair $\langle A, \beta \rangle$ wherein $A$ refers to individuals occurring in the ABox and $\beta$ is a subset of defined concepts in the TBox $T$. From this the goal $s_g$ is translated into a WSML document ($s_g$ Parsing in Figure 12), expressed by means of its input and output parameters. Therefore both the services in the repository and the goal service $s_g$ are described at semantic level, using the WSML formalism. The parameters of $s_g$ may refer to concepts in the TBox $T$ of the domain ontology (Figure 1), or can use some extra concepts that will extend the initial TBox $T$.

In the considered architecture we assumed that a discovery process (i.e., Service Discovery and Selection Module) is in charge of retrieving relevant Web services $S_{W_s}$ (i.e., a subset of $S_{W_s}^*$) depending on the composition goal $s_g$. The latter module is interacting with the Semantic Reasoning Module since some semantic inferences are performed to retrieve the relevant set of Web services $S_{W_s}$. The Service Discovery and Selection Module aims also at parsing relevant Web services $S_{W_s}$ in order to ease the SLM construction.

The architecture is completed by a Semantic Reasoning Module, which provides a vital infrastructural support to three components of the architecture i.e., the Service Discovery and Selection Module, SLM Construction Module, Robustness Verification Module. The main function of this module is to infer some properties on input and output parameters of Web services (defined as concepts of the TBox $T$). For instance the latter module can check satisfiability or subsumption of Web service parameters by means of a DL reasoner, e.g., Fact++

\[http://owl.man.ac.uk/factplusplus/\]

Pellet 

\[http://dee227.poliba.it:8080/MAMAS-tng/DIG\]

In our approach we use the open source Fact++ reasoner to compute standard reasoning and the MAMAS-tng to compute non standard reasoning such as Concept Abduction. Concept Difference is computed by means of an France Telecom component. The power of such a module is therefore crucial to the performance of the overall architecture.

The Functional Level Composition Module is the core module of the FLA Architecture. It interacts with three important modules i.e., the Service Discovery and Selection Module, SLM Construction Module and the $Ra_4C$ Module. From the set $S_{W_s}$ returned by the Service Discovery and Selection Module and the service goal $s_g$, the SLM Construction Module elaborates the SLM (algorithm 1) of the considered Web service composition problem. The SLM Construction Module is related to the Semantic Reasoning Module since the latter module requires semantic reasoning to infer values of semantic links between Web services i.e., Exact, PlugIn, Subsume, Intersection. The $Ra_4C$ Module is re-
sponsible of computing correct, complete, consistent and robust composition of Web services according to the SLM of the domain and the service goal \( s_g \). The robustness feature of Web service composition is ensured by means of the Robustness Verification Module together with the Semantic Reasoning Module. In this case non standard reasoning such as Concept Difference or Concept Abduction are required. Once correct, consistent, complete and robust Web service composition are returned by the Ra4C Module, the Functional Level Composition Module renders the latter compositions in a BPEL4WS format in order to be executed (Requirement \( R_3 \)). The BPEL4WS output file consisted of sequences, parallel branches of Web services, non determinist choice of Web services and assignments between parameters of services (in case of semantic links). By first retrieving potential and conditional compositions and then executing them, we ensure that assumption related to requirement \( R_5 \) does not hold.

7.2. Results and Preliminary Evaluation

We conducted experiments using the implemented prototype system to evaluate our approach. In the experiments the PC used for running the prototype system had the configuration of Intel(R) Core(TM)2 CPU, 1.86GHz with 512 RAM. The PC runs Linux-gnu (2.6.12-12mdk) and Java 2 Edition v1.5.0_11.

Our Web service composition model (i.e., SLM) and algorithms (i.e., Ra4C and its robust form) have been evaluated on two different experimentations. The first experiment is based on the architecture depicted in Figure 12 wherein we test some
France Telecom scenarios in use in the respective domain of Telecom, E-Tourism and E-HealthCare. The first experiment was therefore designed to evaluate an end to end composition (Discovery and FLC) in real scenarios. In the second experiment our Web service composition approach (Ra4C) has bee tested on a wide set of randomly generated Semantic link matrices and service goals $s_g$. The aim of the second experiment conducted was to further evaluate the Ra4C algorithm and also characterise the scalability of our approach.

7.2.1. An end to end Web service composition with Scenarios in Use

In this practical experiment all modules involved in the architecture depicted in Figure 12 have been evaluated. Each module in the architecture relies on powerful state-of-the art technologies and tools to perform the standard tasks. Discovery is based on the jUDDI implementation of the UDDI specification and FLC is performed by means of our approach. Ontology reasoning adopts the WSMO4J implementation for WSMO parsing, and makes use of the FaCT++ DIG reasoner. The remaining code, which includes the various algorithms developed for the architecture as well as the interfacing to the external tools, is realized in Java and ANSI C, for maximum portability. To enable the direct end to end utilization of our architecture, we also included a final automated deployment phase for the resulting orchestrator, which assumes the presence of an Active Web Flow engine running over a local Apache Tomcat platform.

In this experiment, three scenarios in use in France Telecom have been tested:

(i) one in the Telecom domain (the extended version of the motivating scenario in Section 3) where the number of potential Web services ($#S_{W^*}$) is 35 and the TBox of the $\mathcal{ALE}$ Ontology consists of 305 defined concepts and 117 object properties;
(ii) another in the E-Tourism domain where $#S_{W^*}$ is 45 and the TBox of the $\mathcal{ALE}$ Ontology consists of 60 defined concepts and 19 object properties;
(iii) and finally one in the E-HealthCare domain where $#S_{W^*}$ is 12 and the TBox of the $\mathcal{ALE}$ Ontology consists of 105 defined concepts and 37 object properties.

For the three latter scenarios, we specially investigated on the execution time of SLM Construction, Discovery, FLC through Ra4C, and FLC and Robustness.

In the three considered scenarios, Web services in $S_{W^*}$ have at most three input and three output parameters, and at least one input and one output parameter. By applying the Discovery process on the set $S_{W^*}$, we obtain the set of relevant Web services $S_{W^*}$ we can parse in the SLM. The composition problem and the SLM construction is depending on the service goal $s_g := (\mathcal{A}, \beta)$ where $\#\mathcal{A}$ is the number of individuals in $\mathcal{A}$, and $\#\beta$ is the number of concepts in $\beta$. SLMs are mainly characterized by their filling rates $F_r$, i.e., $\frac{\#\text{NonEmptyEntryset}}{\text{Rows} \times \text{Columns}}$ and their averages of elements by non empty entry $m_{i,j}$. In case Ra4C is executed, we return the number of correct, complete and consistent compositions. Moreover the number
Web Service Composition as a Composition of Valid and Robust Semantic Links

<table>
<thead>
<tr>
<th>Parameters &amp; Processes</th>
<th>Description of Main Parameters &amp; Processes</th>
<th>Application Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web services</td>
<td>#S_Ws</td>
<td>Telecom #T := 305</td>
</tr>
<tr>
<td></td>
<td>#InputMax</td>
<td>E-Tourism #T := 60</td>
</tr>
<tr>
<td></td>
<td>#OutputMax</td>
<td>E-HealthCare #T := 105</td>
</tr>
<tr>
<td>Discovery</td>
<td>Discovery (ms)</td>
<td>99.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>108.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58.5</td>
</tr>
<tr>
<td>Goal s_g</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLM Parameters</td>
<td>SLM Construction (ms)</td>
<td>837.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2128.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>581.5</td>
</tr>
<tr>
<td></td>
<td>File Size (bytes)</td>
<td>2865</td>
</tr>
<tr>
<td></td>
<td>#Rows</td>
<td>3190</td>
</tr>
<tr>
<td></td>
<td>#Columns</td>
<td>1.34</td>
</tr>
<tr>
<td></td>
<td>Filling Rate F_r (%)</td>
<td>23.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20%</td>
</tr>
<tr>
<td>Ra4C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ra4C + Robustness</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End to End Composition</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7. Semantic Web Service Composition Tested on Three Scenarios in Use.

The whole execution of the architecture takes from 0.6 to 2.2 seconds depending on the scenario in use. This is much faster than it would take for a user to look for services, and to interact with each of them. In our FLC approach (SLM + Ra4C), the SLM construction is the most time-consuming and costly phase. For instance it takes two seconds to parse and organize 13 Web services in a 10 matrix whereas the Ra4C process takes less than 20 ms to retrieve robust compositions of Web services. We remark that SLMs of our scenarios are relatively sparse i.e., approximately 80% of entries are empty. That is why the Ra4C process is quite fast in the presented scenarios. The robustness computation does not really perturb the composition process. This sounds correct since robustness is checked only for non robust semantic links.

While in the following we described our architecture instantiation supporting semantic and specific requirements and implementations of services, we stress that the architecture design is language-independent. Indeed different instantiations are
also possible as well.

7.2.2. Ra\textsubscript{4}C with random SLM and service goals

In the second experiment SLMs together with their service goal \( s_g := \langle A, \beta \rangle \) have been randomly generated. The aim of such random generations was to further evaluate the Ra\textsubscript{4}C algorithm, and also characterise the scalability of our approach. In the following tests the generated service goals are not trivial i.e., \( \beta \) can not be directly satisfied by the initial conditions \( A \). In other words the goal \( \beta \) can only be satisfied by a composition of at least one Web service. Given this assumption, we will draw some dependences between the time execution of the Ra\textsubscript{4}C process, and the main features of SLMs i.e., their #Rows, #Columns, #\textit{SW}s (and the number of parameters of Web services), \( s_g \), the filling rate of the SLM \( F_r \) and the average of elements by non empty entry \( m_{i,j} \). According to the practical experiments we obtain the following trivial results concerning the properties of SLMs:

- the more Web services in \( \textit{SW}s \) the higher the filling rates of SLMs.
- the more Web services in \( \textit{SW}s \) the higher the average of elements by non empty entry in the SLM.

Roughly speaking the less Web services involved in the SLM the less complex and the sparser the SLM. We also obtain the following more subtle and interesting results:

- the more expressive is the initial conditions \( A \) of the AI planning-based Web service composition \( \langle \textit{SW}s, A, \beta \rangle \) the faster is the Ra\textsubscript{4}C process (Figure 13(a)).
- the sparser the Semantic link matrix the faster is the Ra\textsubscript{4}C process (Figure 13(b) and 15).
- the less functional parameters of Web services the faster is the Ra\textsubscript{4}C process (Figure 14).

According to the previous results, the SLM formalism coupled with the Ra\textsubscript{4}C approach scales very well in case SLMs are relatively sparse (i.e., \( F_r \leq 40\% \)) and the initial condition \( A \) of AI planning-based Web service composition are expressive enough (i.e., \( \#A > \frac{1}{2} \#\text{Rows} \)). We notice that the most of France Telecom scenarios are in such a configuration e.g., the Telecom scenario considers \( \langle \textit{SW}s, A, \beta \rangle \) with \( \#A = 3 \), and a SLM with 7 rows and a filling rate of 23.2\%.

7.3. Some Remarks on Performance of the Ra\textsubscript{4}C process

The SLM is a very interesting feature to perform Web service composition with good performance, and above all with good properties (correctness, completeness, robustness, consistency) since such a matrix is responsible of retrieving all relevant semantic links (at design time) we could encounter in a composition of services (Requirement \( R_8 \)). However it is obvious that the efficiency of the composition
process presented in this work is related to the performance of the discovery process i.e., the set of retrieved Web services \( S_{W_s} \). Indeed the less Web services involved in \( S_{W_s} \) (and in the SLM), the smaller the size of the SLMs (e.g., number of rows and columns of the SLM) and faster the process of composition \( Ra_4C \). In this work we study FLC by applying a very naive algorithm of Web service discovery. One way to easily improve the composition process \( Ra_4C \) consists first in applying a more efficient Discovery process e.g., \( 64 \).
8. Final Remarks and Future Work

The main contribution of this paper w.r.t. research on service oriented computing and more specially semantic Web service is in tackling simultaneously the following issues: (i) elaborating an architecture wherein Web services can be discovered, selected, and composed at Functional Level; (ii) presenting a formal framework where services are characterized in term of their functional properties; (iii) introducing the semantic link concept in Web service composition; (iv) fitting semantic Web service composition to a semantic links composition by means of the feature of sequence composability between services; (v) overcoming robustness in Web service composition; (vi) providing an AI planning-based technique for computing correct, consistent, complete and robust Web service compositions ($Ra_4C$ by means of a domain SLM); (vii) presenting SME$^3$-Pro, an open source prototype tool that implements our technique for automatically computing compositions of Web services; (viii) illustrating our approaches in different levels of scenario i.e., from scenario in Use in Telecom, E-Tourism or E-HealthCare domain, to random scenario to evaluate scalability of our approach. Moreover the suggested approach aims at satisfying requirements $R_{1}, 1 \leq i \leq 6, R_8$ and parts of $R_1$ to the success of FLC.

Recently, $^{53}$ have focused on an approach to order the composition result in case more than one composition is returned (i.e., Requirement $R_7$). Classifying Web service compositions can be useful, especially for the end users or developers that will use the final composition. To do this we plan to order composition according to the robustness criteria of compositions.

Currently, we are investigating on a generalization of semantic links with more services, more output and input parameters. Moreover an alternative to the SLM is studied and compared with the presented approach to perform AI planning based
Web service composition.

Finally, far-reaching future work may be identified along several directions. One of the most interesting future direction consists in applying an efficient discovery component, and then obtain more relevant and useful semantic links. Second, even if our approach is adequate to perform a composition of multiple goals, it could be useful to investigate on a more subtle method to achieve such kind of composition problems. Third but not of least importance, it could be interesting to consider preconditions and effects together with our model of semantic links composition (Requirement $R_1$). In this direction preconditions and effects could prune some non satisfiable composition of services e.g., compositions with non satisfiable preconditions. Another direction would be to study Non functional parameters such as Quality of services together with functional parameters to perform composition of Web services. An open issue will be related to the computation and evaluation of the best compositions depending on two level parameters.

Acknowledgments

This work was funded by France Telecom R&D and a PhD Grant CIFRE 159/2006 from the French National Association for Research and Technology (ANRT).

References


10. Fensel, D., Kifer, M., de Bruijn, J., Domínguez, J.: Web service modeling ontology (wsmo) submission, w3c member submission. (June 2005)


41–50
Conference on Web Intelligence (WI’04). (September 2004)

