Towards Constructive Evidence of Data Flow-oriented Web Service Composition*

Freddy Lécué

IBM Research, Smarter Cities Technology Centre
Damastown Industrial Estate, Dublin, Ireland
{(firstname.lastname)@ie.ibm.com}

Abstract. Automation of service composition is one of the most interesting challenges facing the Semantic Web and the Web of services today. Despite approaches which are able to infer a partial order of services, its data flow remains implicit and difficult to be automatically generated. Enhanced with formal representations, the semantic links between output and input parameters of services can be then exploited to infer their data flow. This work addresses the problem of effectively inferring data flow between services based on their representations. To this end, we introduce the non standard Description Logic reasoning *join*, aiming to provide a "constructive evidence" of why services can be connected and how non trivial links (many to many parameters) can be inferred in data flow. The preliminary evaluation provides evidence in favor of our approach regarding the completeness of data flow.

Keywords: Semantic Web, Web Service, Service Composition, Data Flow, Automated Reasoning, Non Standard Reasoning.

1 Introduction

The Semantic Web [1] is considered to be the future of the current Web. In the Semantic Web, Web services [2] are enhanced using rich description languages e.g., OWL the Web Ontology Language [3]. The underlying descriptions, expressed by means of Description Logic (DL) concepts [4] in domain ontologies, are used to describe the semantics of services e.g., their functional inputs, outputs parameters. Intelligent software agents can, then, use these descriptions to reason about Web services and automate their use to accomplish goals specified by the end-user including intelligent tasks e.g., discovery, selection, composition and execution.

We focus on composition and more specially on its *data flow* i.e., links (or connections) which explain how data is exchanged among services (*Right Panel* in Fig.1). While most approaches [5, 6] derive *control flow* of compositions (i.e., a partial order on services - *Left Panel* in Fig.1) according to a goal to achieve, its data flow remains implicit [7] through opaque and pre-defined assignments from incoming to outgoing

* The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement ID 318201 (SIMPLI-CITY).

services. Usually it is up to developers to provide their details e.g., through BPEL (Business Process Execution Language) assign types or filtering/merging operators. Existing approaches mainly focus in ordering services in a control flow rather than generating its data flow in an automated way. The latter limits flexibility of service oriented computing [8]. Therefore the following are example of open questions in the Web of service community: how to dynamically re-generate data flow specification of "*built-in*" compositions in case of late change of services? Which data is required from which services to turn a composition in its executable state? Does it require data transformation from one description to another? This work investigates the benefits of having semantic descriptions of services a la SA-WSDL [9], OWL-S [10] or WSMO [11] to derive a data flow description of any control flow-based service composition in an automated way.

Towards these issues, some methods [12] exploit expressive DLs to link services through their descriptions, impacting the tractability of the approach. Other approaches [13] limit the expressivity of description through syntactic representation, making data flow very difficult to be automatically derived. In both contexts, complex data links (e.g., filtering, merging) between services cannot be generated in an automated way, providing either abstract or incomplete composition specification. Despite some efforts for pre-defining [14] and inferring [15, 16] compatibilities between services parameters, it remains difficult to derive how data is actually "flowing" from one description to another. In addition, data flow is mainly studied between single outputs and inputs (aka. trivial links). Such links are not appropriate for modeling data flow of complex compositions, limiting their application in real world scenarios. This work tackles this problem.

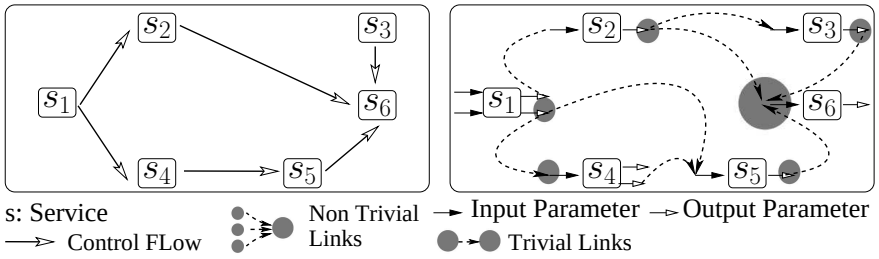

Fig. 1. Control Flow (Left) vs. Data Flow (Right) Views.

Suppose some Semantic Web services[1] being organized in a partial order (based on their overall goals): how to effectively infer their non trivial data flow (e.g., filtering, merging). First of all we define non standard DL reasoning *join* to provide a "constructive evidence" of why services can be connected and how non trivial links can be inferred in data flow. The concept *join* is required to exhibit descriptions J from output parameters Out (of services) which properly ensure Out to be compatible with any input parameter In (of services). In other words the description J is constructed for "glue"-ing outputs and inputs parameters of services, and more importantly used for understanding how data is flowing among services in a composition. Then we describe

[1] Polymorphic services (i.e., exposing several functions depending on inputs combinations) are not investigated here, but can be addressed though conditional compositions [6].

how non trivial data flow can be generated, checked and repaired using concept *join* in order to ensure flexible data flow construction. Service descriptions are formalized in $\mathcal{EL}^{++}$, where subsumption and satisfiability are decidable [17]. For the sake of clarity, we assume compositions without open preconditions. Our work assumes that relevant services are already identified and discovered [18]. Control [7] and data flow [19] based composition techniques, combined with the method introduced in the paper, are then applied to derive ready-to-be-executed compositions.

The remainder of this paper is organized as follows. First of all we summarize data flow-oriented composition, its semantic links and limits. Then we present the DL reasoning *join* to provide a "constructive evidence" of why services can be connected. The next sections (i) describe how *join* can be adapted to simulate and construct complex data flow, and (ii) report some experimental results through comparisons with state-of-the-art approaches. Finally we comment on related work and draw some conclusions.

2 Data Flow-oriented Service Composition

2.1 Service, Semantic Link and Composition

In the Semantic Web, input and output parameters of services are described according to a common ontology or Terminology $\mathcal{T}$ (e.g., Fig.2), where the OWL-S profile, WSMO capability or SA-WSDL can be used as encoding[2], also known as fixed data type or description. Semantic links [19] are defined between output and input parameters of services, based on semantic similarities of their DL encoding. Fig.2 sketches a description of the axioms that are used in the ontology in which the input and output parameters are expressed. Similarities are judged using a matching function between two knowledge representations encoded using the same terminology.

$NetwConnection \equiv \exists netSpeed.Speed$ *// Netw: NetworkConnection*
$Speed \equiv \exists\, mBytes.NoNilSpeed, \quad HighReliable \sqsubseteq Reliable$
$SlowNetwConnection \equiv NetwConnection \sqcap \exists netSpeed.Adsl1M$
$USProvider \equiv \exists to.US,\ UKProvider \equiv \exists to.UK,\ UK \sqcap US \sqsubseteq \bot$
$EUProvider \equiv \exists to.EU, UK \sqsubseteq EU, EU \sqcap US \sqsubseteq \bot, Business \sqsubseteq \top$
$Adsl1M \equiv Speed \sqcap \exists\, mBytes.1M, \quad 1M \sqsubseteq NoNilSpeed$

Fig. 2. DL $\mathcal{EL}^{++}$ Axioms used for representing Output and Input Parameters.

In this context, data flow-oriented service composition consists in retrieving semantic links $sl_{i,j}$:

$$sl_{i,j} \doteq \langle s_i, Sim_{\mathcal{T}}(Out, In), s_j \rangle \tag{1}$$

between an output parameter Out of service s_i and input parameter In of service s_j, where both Out and In are DL descriptions. Thereby s_i and s_j are partially linked according to a matching function $Sim_{\mathcal{T}}$, specifying its data flow. Given a terminology

[2] In case of multiple ontologies used for services descriptions, alignment techniques [20] need to be investigated.

$\mathcal{T}$, the range of $Sim_{\mathcal{T}}$ is determined by five matching types following [21, 22]: i) *Exact* i.e., $Out \equiv In$, ii) *PlugIn* i.e., $Out \sqsubseteq In$, iii) *Subsume* i.e., $In \sqsubseteq Out$, iv) *Intersection* i.e., $\neg(Out \sqcap In \sqsubseteq \bot)$ and v) *Disjoint* i.e., $Out \sqcap In \sqsubseteq \bot$. The cases i)-iv) identify compatible descriptions while the case v) identifies incompatible descriptions Out and In.

2.2 Limitations

As stated in Introduction, models such as (1) are mainly considered for representing trivial semantic links i.e., (boolean) one-to-one compatibility (though matching types) between single output and input parameters. Towards this issue, we generalize (1) by considering In and Out respectively as a conjunction of inputs and outputs of services. Semantic links between "any" output and input at a time i.e., non trivial data flow, can be then represented in (1), which is more appropriate for modeling complex data flow.

However such a model is still limited to understand how data is "flowing" from services to services. Indeed, how data is properly manipulated and adapted between services to ensure data flow? Which part of services descriptions is the most relevant? Is it maximal, minimal, effective and how? These are general questions which remain open in the join domains of Semantic Web and Web of services.

This work suggests concept join as a constructive reasoning to provide a "constructive evidence" of why services can be connected and how complex data flow can be inferred in services composition.

3 Towards Constructive Evidence of Data Flow

Towards the issue of explaining why services can be connected and how non trivial links can be inferred in data flow, Section 3.1 introduces the innovative concept join (Definitions 1, 2 and Propositions 1,2) between data descriptions. Section 3.2 follows the methodology of [23] and [24] to prove the computational complexity of the join reasoning. In particular Proposition 3 is inspired from [23], but highly adapted to concept join (which constructs different descriptions - see Section 6.2). Section 3.3 combines in an innovative way state-of-the-art abduction (Definition 3) and contraction (Definition 4) reasoning techniques to extend the applicability of concept join in a (i) context of service composition, and (ii) when Proposition 1 does not hold (Algorithm 2). Importantly, Section 3.3 explains how non standard reasoning abduction and contraction can be used for enriching the number of joins between services in a composition.

3.1 Concept Join: Definitions and Propositions

We are interested in descriptions in Out which ensure Out and In to be compatible. Therefore we aim at extracting J (Join - Definition 1) from Out such that $J \sqsubseteq In$ remains true in $\mathcal{T}$ (Definition 1). The descriptions R (Remainder), part of Out, such that $Out \equiv R \sqcap J$ will need to be removed from Out since they move Out away from In under subsumption $\sqsubseteq_{\mathcal{T}}$. J highlights descriptions which could be properly joined with In in order to compose outputs Out and inputs In while R points out descriptions which are not required by In.

Definition 1 ***(Concept Join)***
Let $\mathcal{L}$ be a DL, Out, In be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$ such that $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$. A Concept Join Problem, denoted as $CJP\langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ (shortly $Out \blacktriangleright In$) is finding a pair of concepts $\langle R, J\rangle \in \mathcal{L} \times \mathcal{L}$ such that i) $\mathcal{T} \models Out \equiv R \sqcap J$ and ii) $\mathcal{T} \models J \sqsubseteq In$. Then J (or $\blacktriangleright_J$), which is not symmetric, is a join between Out and In in $\mathcal{T}$.

We use $\mathcal{P}$ as a symbol for a $CJP\ \langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ and we denote with $SOLCJP(\mathcal{P})$ the set of all solutions of the form $\langle R, J\rangle$ to a $CJP\ \mathcal{P}$. In case $\mathcal{T} \not\models Out \sqsubseteq In$, the $CJP\ \mathcal{P}$ has no solution at all, as stated formally in Proposition 1.

Proposition 1. ***(No Solution of a CJP)***
Let $\mathcal{P} = \langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ be a CJP such that $\mathcal{T} \not\models Out \sqsubseteq In$. The set $SOLCJP(\mathcal{P})$ is defined by $\emptyset$.

Proof. Since Out can be rewritten as $R \sqcap J$ (condition (i) in Definition 1) with $R = \top$ and $J = Out$ without loss of generality, then $\mathcal{T} \not\models Out \sqsubseteq In$ (Proposition 1) becomes $\mathcal{T} \not\models J \sqsubseteq In$. The latter contradicts $\mathcal{T} \models J \sqsubseteq In$ (condition (ii) in Definition 1), so no possible solution of a $CJP\ \mathcal{P}$ in case $\mathcal{T} \not\models Out \sqsubseteq In$.

$\mathcal{T} \models Out \sqsubseteq In$ implies that there is always the trivial solution $\langle \top, Out\rangle$ to a CJP $\langle \mathcal{L}, Out, In, \mathcal{T}\rangle$.

Proposition 2. ***(Trivial Solution of a CJP)***
If $Out \equiv In$ in $\mathcal{T}$ then $\langle \top, Out\rangle \in SOLCJP(\langle \mathcal{L}, Out, In, \mathcal{T}\rangle)$.

This case refers to an *exact* composition [25] of services s_i and s_j: if we want to proceed s_j, all outputs Out of s_i are required (since J is defined by Out in Proposition 2) to achieve all input In of s_j. Then, no description R has to be removed from Out. On the other hand, when $Out \sqsubset In$ (i.e., $\mathcal{T} \models Out \sqsubseteq In$ and $\mathcal{T} \not\models Out \equiv In$), $\langle \top, Out\rangle$ is also one potential solution of the CJP problem. However, other solutions with R not being $\top$ are possible. Obviously, in order to achieve a composition between Out and In the first case (in Proposition 2) is in a much better shape than the second one. Indeed all descriptions In, which are required by s_j, are provided by Out. If we want to use join to highlight the closest descriptions in Out (i.e., the most general) to In, emphasising the most compatible descriptions in Out for In to compose s_i and s_j, "effective" joins under $\sqsubseteq_{\mathcal{T}}$ need to be defined (Definition 2 adapted from [26]).

Definition 2 ***(Effective Join Solution)***
Let $\mathcal{P} = \langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ be a CJP. The set $SOLCJP_{\sqsubseteq}(\mathcal{P})$ is the subset of $SOLCJP(\mathcal{P})$ whose join concepts J are maximal under $\sqsubseteq_{\mathcal{T}}$. The set $SOLCJP_{\leq}(\mathcal{P})$ is the subset of $SOLCJP(\mathcal{P})$ whose join concepts have minimum length.

Formally the set $SOLCJP_{\sqsubseteq}(\mathcal{P})$ satisfies both Definition 1 and the following condition: $\forall \langle R', J'\rangle \in \mathcal{L} \times \mathcal{L} : \mathcal{T} \models Out \equiv R' \sqcap J' \wedge \mathcal{T} \models J' \sqsubseteq In \Rightarrow J' \sqsubseteq J$. Maximality under $\sqsubseteq_{\mathcal{T}}$ is considered as a effectiveness criterion since no unnecessary joins is assumed between Out and In.

Example 1 ***(Effective Join Solution - Fig.3)***
Let s_1 be an `InternetEligibility` *service which returns as output Out: the NetworkConnection (e.g., Speed, UK Country) of a desired geographic zone together with information about its network provider (Reliability, Business type). Let s_2 be another telecom service which requires a Reliable network provider in UK as input In to be executed. Out and In, as DL representations of functional parameters in Fig.3, ensure $Out \sqsubseteq In$ in $\mathcal{T}$. On the one hand $\exists netSpeed.Adsl1M \sqcap \exists to.UK \sqsubseteq NetwConnection$. On the other hand $HighReliable \sqsubseteq Reliable$. In other words some outputs produced by s_1 can be consumed by some inputs of s_2. The effective join J of Out and In (under $\sqsubseteq_{\mathcal{T}}$) is $\exists netSpeed.Adsl1M \sqcap \exists to.UK \sqcap HighReliable$ while the discarded description R is $Business$. An instance of J is then required to instantiate In (and execute s_2): $SlowNC$ (NC refers to $NetwConnection$), $\exists to.UK$, $Reliable$ while an instance of $Business$ is not. The description J acts as a filter between s_1, s_2 to restrict Out over the data flow. In other words J establishes which descriptions are relevant to link Out to In. The two output instances of s_1 are then practically merged into one instance for s_2 through the construction of J.The latter ensures the executability of s_2.*

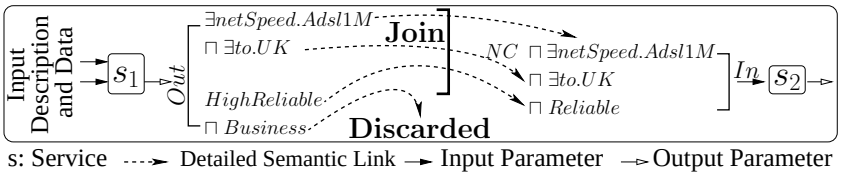

Fig. 3. Effective Join Solution.

In [26] it was proven that $\leq$-minimality is more appropriate for conciseness, but largely depending on $\mathcal{T}$. Indeed, by simply adding axioms $A \equiv R$ and $B \equiv J$, we obtain a $\leq$-minimal solution $\langle A, B\rangle$ for each pair $\langle R, J\rangle \in SOLCJP(\mathcal{P})$.

3.2 Computational Complexity

Since concept join can be considered as an extension of concept subsumption with respect to a TBox, its lower bounds carry over to decision problems related to a CJP.

Proposition 3. ***(Deciding Existence of Join)***
Let $\mathcal{P} = \langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ be a CJP. If concept subsumption with respect to a $\mathcal{T}$ in $\mathcal{L}$ is a problem $\mathcal{C}$-hard for a complexity class $\mathcal{C}$, then deciding whether a pair of concepts $\langle R, J\rangle \in \mathcal{L} \times \mathcal{L}$ belongs to $SOLCJP(\mathcal{P})$ is $\mathcal{C}$-hard.

Proof. Since $\mathcal{T} \models Out \sqsubseteq In$ iff $\langle \top, Out\rangle \in SOLCJP(\mathcal{P})$, such a problem is $\mathcal{C}$-hard.

In our $\mathcal{EL}^{++}$ context, deciding whether a pair of concepts $\langle R, J\rangle$ belongs to $SOLCJP(\mathcal{P})$ is PTIME-hard [27] with respect to both acyclic and cyclic TBoxes $\mathcal{T}$.

Regarding upper bounds, a simple result can be derived from the fact that $\langle \top, Out\rangle$ is always a solution of the CJP $\langle \mathcal{L}, Out, In, \mathcal{T}\rangle$ if $Out \sqsubseteq In$ in $\mathcal{T}$ (Proposition 2) although not always an effective one for join. Following [23], a total length-lexicographic

order $\prec_{lex}$ can be defined over concepts as follows: given two concepts $Out, In \in \mathcal{L}$, let $Out \prec_{lex} In$ if either $|Out| < |In|$, or both $|Out| = |In|$ and Out is lexicographically before In. Based on this total order, an approach for finding a $\leq$-minimal solution of a CJP, using polynomial space relatively to an oracle for subsumption in $\mathcal{L}$, is presented in Algorithm 1. Algorithm 1 is innovative as it enumerates concept join solutions over a total length-lexicographic ordered concepts.

Algorithm 1: Effective $\blacktriangleright_J$ of a CJP.

```
1 Input: A CJP P = ⟨L, Out, In, T⟩ with T ⊨ Out ⊑ In.
2 Result: A concept x ∈ L such that ⟨R, x⟩ ∈ L × L is in SOLCJP≤(P).
3 begin
4     x ← ⊤; // Initialisation
5     while |x| < |Out| do
6         if T ⊨ Out ⊏ x and T ⊭ x ⊑ In then
7             return x;
8         x ← next concept following x in ≺lex;
9     x ← Out; return Out;
```

Algorithm 1 uses polynomial space (considering one call to subsumption as an oracle) since it just tries all concepts with less symbols than Out, and returns Out if it does not find a shorter solution. Thus, it provides an upper bound on the complexity of CJP, depending on the complexity class to which subsumption in $\mathcal{L}$ belongs to. Although this result does not directly lead to a practical algorithm, it provides an upper bound on the complexity of the problem, hence on the complexity of every optimal algorithm.

Theorem 1. ***(Finding a Solution in*** $SOLCJP_{\leq}(\mathcal{P})$***)***
Let $\mathcal{P} = \langle \mathcal{L}, Out, In, \mathcal{T} \rangle$ be a CJP. If concept subsumption with respect to a $\mathcal{T}$ in $\mathcal{L}$ belongs to a complexity class $\mathcal{C}$ that is included in PSPACE then finding a pair of concept in $SOLCJP_{\leq}(\mathcal{P})$ is a problem in PSPACE. Otherwise if PSPACE is included in $\mathcal{C}$, then finding a pair of concept in $SOLCJP_{\leq}(\mathcal{P})$ is a problem in $\mathcal{C}$.

According to Theorem 1, inspired from [26], finding a pair of concept for the problem $SOLCJP_{\sqsubseteq}(P)$ in $\mathcal{EL}^{++}$ is in PSPACE. Theorem 1 simply builds on top of the subsumption properties.

3.3 Incompatible Descriptions in Concept Join

As highlighted by Proposition 1, Definition 1 has no solution if $\mathcal{T} \not\models Out \sqsubseteq In$. This limits the applicability of concept join by restricting services to exchange data (from Out to In) only under $Out \sqsubseteq In$ in $\mathcal{T}$. Even if this is a basic requirement to compose and join services, other potential compositions, which do not satisfy $Out \sqsubseteq In$ [25], would be ignored since their join cannot be derived. Towards this issue, we exploit constructive DL reasoning abduction [28] (Definition 3) and contraction [24] (Definition 4) to respectively consider join if i) In does not subsume Out but have a consistent conjunction i.e., $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$ and ii) their conjunction is inconsistent

i.e., $\mathcal{T} \models Out \sqcap In \sqsubseteq \bot$. While concept abduction derives description which is missing in Out to be subsumed by In, concept contraction [24] retracts specification G (for *Give up*) in Out to obtain a concept K (for *Keep*) such that $K \sqcap In$ is satisfiable in $\mathcal{T}$. The latter extends abduction to unsatisfiable conjunction of Out and In.

Definition 3 ***(Concept Abduction)***
Let $\mathcal{L}$ be a DL, Out, In be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$ such that $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$. A Concept Abduction Problem: $In \backslash Out$ is finding a concept $H \in \mathcal{L}$ such that $\mathcal{T} \not\models Out \sqcap H \equiv \bot$, and $\mathcal{T} \models Out \sqcap H \sqsubseteq In$.

Similarly to concept join, abduction extends subsumption. It also constructs a concept H to ensure $Out \sqcap H$ be subsumed by In. By computing description H using abduction, join can be derived between $Out \sqcap H$ (instead of Out) and In. Abduction is then required to enlarge the scope of Definition 1 i.e., from $Out \sqsubseteq In$ to $\neg(Out \sqcap In \sqsubseteq \bot)$ in $\mathcal{T}$.

Contraction, which extends satisfiability, aims to retract specification G (for *Give up*) in Out to obtain a concept K (for *Keep*) such that $K \sqcap In$ is satisfiable in $\mathcal{T}$.

Definition 4 ***(Concept Contraction)***
Let $\mathcal{L}$ be a DL, Out, In be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$ where both Out and In are satisfiable in $\mathcal{T}$. A Concept Contraction Problem, denoted as $In \lrcorner Out$ is finding a pair of concepts $\langle G, K \rangle \in \mathcal{L} \times \mathcal{L}$ such that $\mathcal{T} \models Out \equiv G \sqcap K$ and $\mathcal{T} \not\models K \sqcap In \sqsubseteq \bot$. Then K (or $\lrcorner_K$) is a contraction of Out according to In and $\mathcal{T}$.

By computing (1) contraction $\lrcorner_K$: a part of Out which ensures $\lrcorner_K \sqcap In$ to be satisfiable in $\mathcal{T}$ (i.e., validating conditions of Definition 3), and then (2) abduction $In \backslash \lrcorner_K$ which ensures $\lrcorner_K \sqcap (In \backslash \lrcorner_K) \sqsubseteq In$, join can be derived between $\lrcorner_K \sqcap (In \backslash \lrcorner_K)$ and In. Thus contraction can be applied to enlarge the scope of Definition 1: from $Out \sqsubseteq In$ to $Out \sqcap In \sqsubseteq \bot$ in $\mathcal{T}$.

Algorithm 2 sketches the approach to enlarge the scope of Definition 1. It ensures that Out and In can be joined by iteratively weakening and strengthening Out through contraction and abduction. Besides the case already supported by Propositions 1 and 2 and its extension to $Out \sqsubset In$ (line 6), abduction (lines 10, 14) is applied if $Out \sqcap In$ is consistent (line 9) in $\mathcal{T}$. Alternatively contraction (line 13) is required beforehand (line 12). The most specific contraction is considered to obtain a description as close as possible to Out. Thus, the join is derived between (1) Out and In in the trivial case $Out \sqsubseteq In$ (line 6), (2) $Out \sqcap (In \backslash Out)$ and In if $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$ (line 9) and (3) $(In \lrcorner_K Out) \sqcap (In \backslash (In \lrcorner_K Out))$ and In if $\mathcal{T} \models Out \sqcap In \sqsubseteq \bot$ (line 12).
The complexity of Algorithm 2 is in PSPACE in $\mathcal{EL}^{++}$. Indeed lines 6, 9, 12 are in PTIME [17], line 13 is in PTIME (Theorem 4 in [24]), lines 10, 14 are in PSPACE (Theorem 1 in [28]), line 15 is in PSPACE (Theorem 1).

4 Composing Services with Concept Join

We present how concept join can be used to compose properly services through complex data flow modelling.

4.1 Join-ing Data and Descriptions of Services

Compositions of any outputs Out with inputs In can be derived using Algorithm 2. The data flow is established by joining their descriptions. In case their join cannot be derived (lines 9 and 12), we apply contraction and abduction to identify data descriptions which need to be removed/added from/to outputs Out of services with respect to inputs In.

Algorithm 2: Computing Join (Case $\mathcal{T} \not\models Out \sqsubseteq In$).

1 **Input**: A $CJP\ \mathcal{P} = \langle \mathcal{L}, Out, In, \mathcal{T} \rangle$.
2 **Result**: A pair $\langle R, J \rangle \in \mathcal{L} \times \mathcal{L}$ which is in $SOLCJP_{\sqsubseteq}(\mathcal{P})$.
3 **begin**
4 $H \leftarrow \top$; *//Initialisation*
5 *// Trivial Case of Subsumption between* Out *and* In.
6 **if** $\mathcal{T} \models Out \sqsubseteq In$ **then**
7 ; *// Propositions 1, 2 and its Extension to* $Out \sqsubset In$.
8 *// Extension to Consistent Conjunction* | $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$.
9 **else if** $\mathcal{T} \not\models Out \sqcap In \sqsubseteq \bot$ **then**
10 $H \leftarrow In \backslash Out$; *// Abduction*
11 *// Extension to Inconsistent Conjunction of* Out *and* In.
12 **else if** $\mathcal{T} \models Out \sqcap In \sqsubseteq \bot$ **then**
13 $Out \leftarrow (In \lrcorner_K Out)$; *// Contraction*
14 $H \leftarrow In \backslash Out$; *// Abduction*
15 $\langle R, J \rangle \leftarrow SOLCJP_{\sqsubseteq} \langle \mathcal{L}, Out \sqcap H, In, \mathcal{T} \rangle$; *// Min. Join*
16 **return** $\langle R, J \rangle$;

In some cases, Semantic Web services *consumed* and *produced* data that does not fit its static semantic description, making semantics of data not as precise as it should be. In this context, we proceed as following: (1) detecting the most accurate semantic description of concrete data values following [20]), (2) expanding the domain ontology with this new description, mainly for reasoning purpose, and (3) applying Algorithm 2 at run time to obtain joins. The steps (1) and (2) ensures that the reasoning at description level (through Algorithm 2) is also valid at a lower (i.e., data) level. This case of non-alignment between data and their description justifies and reinforces the use of non standard reasoning to capture composition. Indeed, more inconsistent joins could occur, limiting the applicability of pure equivalence-based approaches [16].

4.2 Simulating Complex Data Flow Operators

Definition 1, as a way to identify (semantic) link-"*able*" descriptions in composition, can be used to simulate/infer complex data flow operators e.g., "*Data Filter*", "*Merge*". Their benefit is twofold: modeling and explaining how services and their data can be properly manipulated and adapted in data flow-oriented composition. Contrary to [25, 16, 6], among others, automated generation, verification and repair of complex data

flow in composition can be enabled once integrated in a composition engine [29]. In the following the symbol ► will denote the problem in Definition 1 where both (i) effective join solutions (Definition 2) and (ii) maximality under $\sqsubseteq_{\mathcal{T}}$ are considered.

• **Data Filter**: [14] commonly used the data filter operator in data flow-oriented service composition to i) extract some descriptions Y and ii) block the rest In from an incoming description X with respect to a filter (description) Z (see illustration in Fig.4). This operator is simulated by X ► Z and its solution $\langle In, Y\rangle$. $X \sqsubseteq Z$ since Z is used as a filter for X. The effectiveness condition (Definition 2) is crucial to avoid any undesired data in Y e.g., In. The more specific the filter Z (i.e., the closer to X), the less descriptions blocked by Z (the least is $\top$).

Example 2 ***(Data Filter - Fig.4 a))***
Let Y be defined by $\exists to.UK$ and D be defined by $Business$. The descriptions Y and D are respectively extracted and blocked from description X i.e., $\exists to.UK \sqcap Business$ using the filter Z, defined by $\exists to.EU$. Each data instance from X is split along Y and D. Only instance of X is connected to Y.

• **Data Merge**: In [7] it is used to aggregate descriptions X_1 and X_2 into a description Y with respect to a filter Z (see illustration in Fig.4). If X_1 and X_2 are compatible, this operator can be simulated by $(X_1 \sqcap X_2)$ ► Z and its solution $\langle In, Y\rangle$. $X_1 \sqcap X_2 \sqsubseteq Z$ since Z is used as a filter for X_1 and X_2. In refers to descriptions which are blocked from X_1 and X_2 with respect to Z. In case $X_1 \sqcap X_2 \equiv Z$, all descriptions from $X_1 \sqcap X_2$ are merged, ensuring In to be $\top$ i.e., none of descriptions in $X_1 \sqcap X_2$ is blocked from Y. A generalization to n descriptions to merge is straightforward.

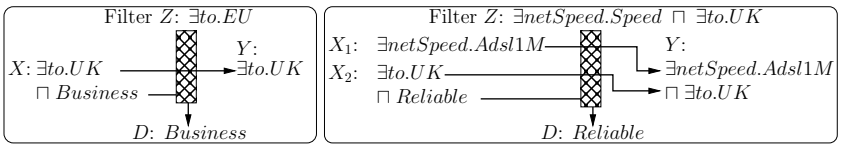

Fig. 4. Simulation of Data Filter and Merge with Join.

Example 3 ***(Data Merge - Fig.4 b))***
$\exists netSpeed.Adsl1M \sqcap \exists to.UK$ is the merging description of X_1, X_2 in Fig.4 b) using the filter $\exists netSpeed.Speed \sqcap \exists to.UK$ while $Reliable$ is the description which is blocked.

Based on a straightforward extension of Algorithm 2 with effective concept join, most common complex data flow operators e.g., *Data Merge*, *Filter* can be derived in any data flow, modeling and explaining how services and their data are adapted. Algorithm 2 can be also used to validate pre-defined links or complete existing ones. More generally effective concept join can be used in any data-based application e.g., as a way to retrieve instances of Z from a large set of data Y given some constraints X i.e., Y ► Z.

5 Experimental Results

In more details we analyze our approach (Algorithm 2 and its extension for data flow simulation) by comparing its performance against existing approaches [5–7] along two dimensions: (i) CPU time (in ms) to generate composition and (ii) completeness of data flow. The second dimension is evaluated by computing the rate: data descriptions connections retrieved against those expected in the optimal composition. This composition, which is manually constructed based on services descriptions and their goal, has no open links (i.e., links reaching to a non executable process) and no redundant links. The experiments have been conducted on Intel(R) Core (TM)2 CPU, 2.4GHz, 2GB RAM.

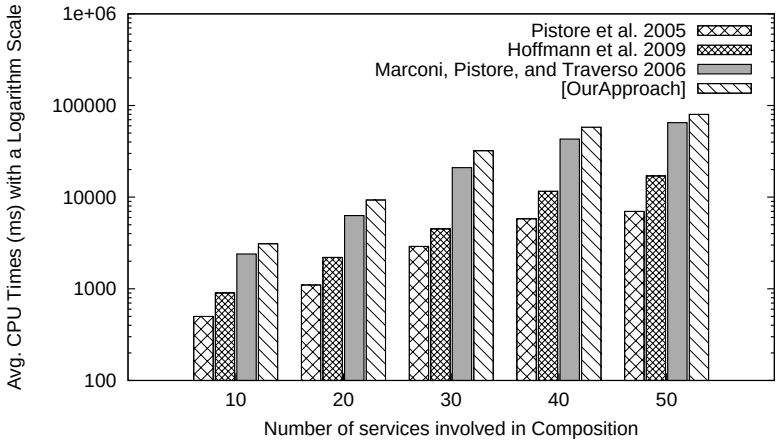

Fig. 5. Computation Time of Composition Approaches.

• **Context:** Compositions with up to 50 services have been extracted from [30] and enriched using a commercial $\mathcal{EL}^{++}$ ontology (1100 concepts, 390 properties: 384 concepts subsume the 716 remaining ones with a maximal depth of 8). The semantic annotations are important for deriving data flow in our approach. SOUR[3] is used for services annotations. The annotation process is costly e.g., 8 person/hours for 50 services (with an average of 5 inputs/outputs) with the latter ontology, but has a positive impact on automation of compositions. For scalability purpose we guided the semantic link detection since each composition is bound by $n \times 2^n$ potential semantic links, with n be the number of services. In more details we limited the number of Out (input of Algorithm 2) to be computed beforehand e.g., by ranking Out with respect to In (e.g., size of their contraction/abduction) and considering only Out which ensures to obtain the top k contraction/abduction. The semantic link detection was required only by our approach, mainly to (i) identify potential data flow in composition and (ii) avoid the computation of an exponential number of join, which strongly reduce the overall com-

[3] http://www.soa4all.eu/tools.html

putation time. The data flow requirements are formalized for [7] while only composition goals are defined for [5, 6].

• **Results - Computation Time:** Fig.5 illustrates the computation costs for constructing compositions with up to 50 services. Our approach is the most time consuming although (i) a control flow-based compositions is pre-defined and (ii) conjunctions of outputs are considered satisfiable. Other approaches, generating control flow-based compositions, are faster. The best approach [5] generates compositions of 50 services in 7.2 seconds.

• **Results - Data Flow Completeness:** Fig.6 sketches the comparison of our approach vs. existing approaches. The same number of compositions has been retrieved in all cases. The only difference is related to its data flow description. On average our approach automatically derives 83% of the final data flow structure (i.e., data filter, merge operators) of a data flow-free composition. The 17% remaining connections, are cyclic-based data flow operators e.g., loop, which is not supported by our current implementation. On average no more than 55% of connections are retrieved with the state-of-the-art approach [7]. The approach of [5] generates an average of 9% of connections. As reported by their authors, this is more appropriate for independent services.

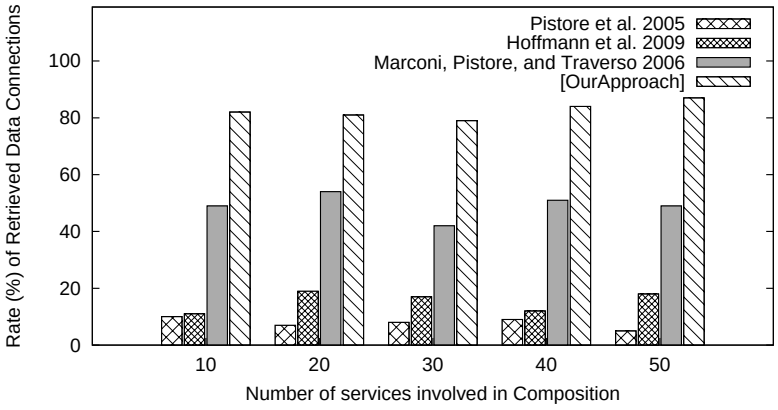

Fig. 6. Data Flow Completeness.

• **Lessons Learned:** Even if state-of-the art approaches are appropriate for fast elaboration of control-flow-based composition, they are not necessarily adequate for (i) detecting connections between services and (ii) connecting their descriptions. The automated construction of complex data flow in $\mathcal{EL}^{++}$ DL has a negative impact on the computation costs but ensures a finer description of compositions, which are ready for execution. The size and the structure of the ontology have a limited impact. The main factors for the increase of computation cost are (i) the expressivity of the DL and (ii) the number of DL conjuncts (and their complexity) used to describe services. The reduction of its expressivity has a positive impact on scalability, but it also decreases the completeness and quality of data flow. The scalability can be improved by considering

only subsumption-based comparisons of descriptions (line 6), removing computation of abduction and contraction. In such a case the rate of data flow completeness is also decreasing. By removing the abduction and contraction parts of Algorithm 2 (from line 9 to 14), our approach is more scalable than state-of-the-art approaches, but only 55% of data flow description is retrieved. According to our experiments a best trade-off is proposed in [7], while [5, 6] fits perfectly independent services with a better scalability for [5].

• **Limitations:** The computed potential connections are all used for defining the data flow of the composition. However if multiple services provide similar output (respectively input) descriptions, they are all equally considered. All their output (respectively input) descriptions are aggregated and subject to a join with other services. This case falls in a special case of "Data Merge" (Fig.4 b where $X_1 \equiv X_2$, with X_1 and X_2 outputs of two distinct services). Additional manual efforts are required if such cases need to be avoided, which were not foreseen in our applications.

6 Related Work

6.1 Data Flow-based Semantic Service Composition

Fig.7 positions existing approaches in relation to 3 dimensions: control flow, data flow, description expressivity. These dimensions aree used to structure the remainder of this section.

Mash-up-based approaches [31, 13] and semantics-based methods [7, 32, 14], positioned in *Front Cluster* of Fig.7, achieve composition by linking services according to different expressivity of static control flow and pre-defined data flow operators (with explicit requirements). They are all limited by the expressivity of service descriptions. Indeed the latter are constrained by RDF/S while the former support only basic XML-based transformation. By embedding compositions with advanced control flow [7], the data flow construction is reduced. [14] provide a more complete (pre-designed) panel of data flow operators, such as *Construct* and *Mix*, which can be simulated by Definition 1, but support only RDF/S, focusing at instance level. Their applicability to expressive semantics and the automated construction of data flow is then limited.

AI planning- [6, 33] and DL-based approaches [15, 12], positioned in *Back Cluster* of Fig.7) elaborate composition of services by reasoning on their descriptions. Despite higher expressivity, only sequence-based data flow is inferred. The approaches of [15, 25, 32] are even more restrictive as they consider (specialized) semantic links between one output and input. More elaborated operators have been presented by [16] towards this issue. Contrary to our approach, data flow is based on concrete values and not their semantic descriptions, which is more flexible for handling misalignment data-description e.g., the instance defined by $(\exists hasConnection.ADSL512KBS)$ where $ADSL512KBS$ is a $SlowNetwConnection$ partially respects the description $SlowNetwConnection \sqcap \exists to.UK$. Indeed no instance of a provider is provided. We address it by using non standard reasoning. Other approaches simulate sequence [33] and conditional-based [6], e.g., through forward effects for the latter, limiting the expressivity of compositions.

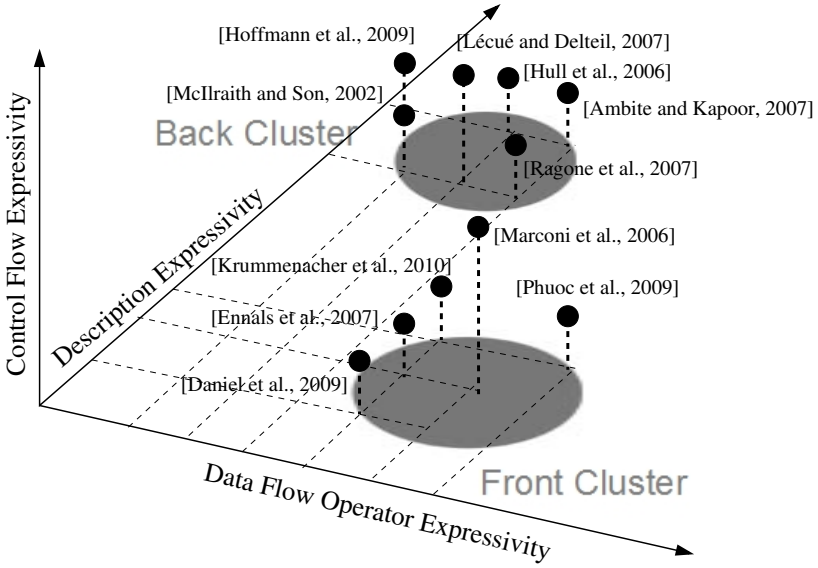

Fig. 7. Classification of Data Flow-oriented Composition.

6.2 Existing Constructive DL Reasoning

While abduction [26] derives description which is missing in Out to be subsumed by In, concept contraction [24] retracts specification G (for *Give up*) in Out to obtain a concept K (for *Keep*) such that $K \sqcap In$ is satisfiable in $\mathcal{T}$. The latter extends abduction to unsatisfiable conjunction of Out and In. Approximate subsumption has been presented by [34]. Such types of reasoning construct concepts which are missing or over-specified in Out to be respectively (1) subsumed by and (2) consistent with In. Concept join constructs more general concepts from Out which are subsumed by In. In particular, its effective solutions (under $\sqsubseteq_{\mathcal{T}}$) refer to the most general description of Out which is subsumed by In. Abduction and approximate subsumption extend Out while join extracts a part of Out for the same objective i.e., being subsumed by In. If $Out \sqsubseteq In$, abduction, contraction and approximate subsumption do not construct any description while concept join does. It explains the way they are joined.

Subsumption between DLs concepts Out and In can be explained by deriving its formal proof (i.e., which descriptions in In subsume which descriptions in Out) in [35]. Concept join does not provide any explanation of subsumption, but instead closer descriptions J (in Out) of In given Out under $\sqsubseteq_{\mathcal{T}}$.

7 Conclusion

In this paper we studied data flow-oriented Web service composition. Our work has been directed to meet the main challenges facing this problem i.e., *how to effectively*

infer data flow between services based on their DL $\mathcal{EL}^{++}$ descriptions? Firstly we introduced the constructive reasoning *join* in $\mathcal{EL}^{++}$, aiming to provide a "constructive evidence" of why services can be connected. Then we described how non trivial data flow can be generated, checked and (potentially) repaired using concept *join*, all ensuring flexible data flow construction. Thus, implications of control flow modification on data flow can be investigated. The experimental results provide evidence in favor of our approach regarding the completeness of data flow.

Future works will focus on modeling data flow operators at instance level [14] i.e., how do loops in control flow work together with data flow? We will also investigate metrics for evaluating data flow precision.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5) (2001) 34–43
2. Sycara, K.P., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated discovery, interaction and composition of semantic web services. J. Web Sem **1**(1) (2003) 27–46
3. Smith, M.K., Welty, C., McGuinness, D.L.: Owl web ontology language guide. W3c recommendation, W3C (2004)
4. Baader, F., Nutt, W. In: The Description Logic Handbook: Theory, Implementation, and Applications. (2003)
5. Pistore, M., Marconi, A., Bertoli, P., Traverso, P.: Automated composition of web services by planning at the knowledge level. In: IJCAI. (2005) 1252–1259
6. Hoffmann, J., Bertoli, P., Helmert, M., Pistore, M.: Message-based web service composition, integrity constraints, and planning under uncertainty: A new connection. J. Artif. Intell. Res. (JAIR) **35** (2009) 49–117
7. Marconi, A., Pistore, M., Traverso, P.: Implicit vs. explicit data-flow requirements in web service composition goals. In: ICSOC. (2006) 459–464
8. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F., Krämer, B.J.: 05462 service-oriented computing: A research roadmap. In: Service Oriented Computing. (2005)
9. Kopecký, J., Vitvar, T., Bournez, C., Farrell, J.: Sawsdl: Semantic annotations for wsdl and xml schema. IEEE Internet Computing **11**(6) (2007) 60–67
10. Ankolenkar, A., Paolucci, M., Srinivasan, N., Sycara, K.: The owl-s coalition, owl-s 1.1. Technical report (2004)
11. Fensel, D., Kifer, M., de Bruijn, J., Domingue, J.: Web service modeling ontology submission, w3c submission. (2005)
12. Ragone, A., Noia, T.D., Sciascio, E.D., Donini, F.M., Colucci, S., Colasuonno, F.: Fully automated web services discovery and composition through concept covering and concept abduction. Int. J. Web Service Res. **4**(3) (2007) 85–112
13. Ennals, R., Brewer, E.A., Garofalakis, M.N., Shadle, M., Gandhi, P.: Intel mash maker: join the web. SIGMOD Record **36**(4) (2007) 27–33
14. Phuoc, D.L., Polleres, A., Hauswirth, M., Tummarello, G., Morbidoni, C.: Rapid prototyping of semantic mash-ups through semantic web pipes. In: WWW. (2009) 581–590
15. Hull, D., Zolin, E., Bovykin, A., Horrocks, I., Sattler, U., Stevens, R.: Deciding semantic matching of stateless services. In: AAAI. (2006)
16. Ambite, J.L., Kapoor, D.: Automatically composing data workflows with relational descriptions and shim services. In: ISWC/ASWC. (2007) 15–29
17. Baader, F., Brandt, S., Lutz, C.: Pushing the el envelope. In: IJCAI. (2005) 364–369

18. Benatallah, B., Hacid, M., Leger, A., Rey, C., Toumani, F.: On automating web services discovery. VLDB Journal (December 2002) 1–26
19. Lécué, F., Léger, A.: A formal model for semantic web service composition. In: ISWC. (2006) 385–398
20. Euzenat, J.: Semantic precision and recall for ontology alignment evaluation. In: IJCAI. (2007) 348–353
21. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic matching of web services capabilities. In: ISWC. (2002) 333–347
22. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: WWW. (2003) 331–339
23. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: Concept abduction and contraction in description logics. In: DL. (2003)
24. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: A uniform tableaux-based method for concept abduction and contraction in description logics. In: ECAI. (2004) 975–976
25. Lécué, F., Delteil, A.: Making the difference in semantic web service composition. In: AAAI. (2007) 1383–1388
26. Noia, T.D., Sciascio, E.D., Donini, F.M.: Semantic matchmaking as non-monotonic reasoning: A description logic approach. J. Artif. Intell. Res. (JAIR) **29** (2007) 269–307
27. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: IJCAI. (2003) 325–330
28. Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: Abductive matchmaking using description logics. In: IJCAI. (2003) 337–342
29. Wu, D., Parsia, B., Sirin, E., Hendler, J.A., Nau, D.S.: Automating DAML-S web services composition using SHOP2. In: ISWC. (2003) 195–210
30. Oh, S.C., Kil, H., Lee, D., Kumara, S.R.T.: Wsben: A web services discovery and composition benchmark. In: ICWS. (2006) 239–248
31. Daniel, F., Casati, F., Benatallah, B.: Hosted universal composition: Models, languages and infrastructure in mashart. In: ER. (2009) 428–443
32. Krummenacher, R., Norton, B., Marte, A.: Towards linked open services and processes. In: FIS. (2010) 68–77
33. McIlraith, S.A., Son, T.C.: Adapting golog for composition of semantic web services. In: KR. (2002) 482–496
34. Stuckenschmidt, H.: Partial matchmaking using approximate subsumption. In: AAAI. (2007) 1459–1464
35. McGuinness, D.L., Borgida, A.: Explaining subsumption in description logics. In: IJCAI (1). (1995) 816–821