A FORMULA DRIVEN INCREMENTAL CONSTRUCTION OF WEB SERVICE COMPOSITIONS

Antonella Santone*, Gigliola Vaglini+ and Maria Luisa Villani*
* *Dipartimento di Ingegneria, University of Sannio, Benevento, Italy*

+ *Dipartimento di Ingegneria della Informazione, University of Pisa, Italy*

Keywords: Model checking, Temporal logic, Tableaux, Web services.

Abstract: We present a modular approach to system specification to support the realization of web services. In particular, we solve the following problem: given the formal specification of the (incomplete) system, say p, already built, what is a characterization of the sub-systems that can collaborate with p, through a given communication interface $\mathcal{L}$, so that the complete system satisfies a given property φ? An automatic procedure is defined to identify the formula ψ such that, for each process q satisfying ψ, the parallel composition of p and q through $\mathcal{L}$ satisfies φ. For applicability of the method to web service compositions the formula ψ should specify, as much as possible, only the communication actions that allow p to correctly fulfill φ.

1 INTRODUCTION

The Service Oriented Architecture (SOA) model has led to rethinking the way software systems are developed: systems are conceived as collaborations of simpler services, whose concrete realizations will be selected or even discovered at run-time. Also, these systems can reconfigure themselves to recover from problems that may occur during execution. Thus, proper mechanisms to specify the behaviors of the required services are essential, to enable automatic search and compatibility checks of the services that can be bound to the composition. Most importantly, validity of the composition with respect to global objectives must be ensured not only at design time, but also at execution time, when a binding with some service might be changed with another one.

We present a modular approach to system specification to support the realization of such systems. In particular we solve the following problem: given the formal specification of the (incomplete) system, say p, already built, what is a characterization of the subsystems that can collaborate with p, through the given communication interface $\mathcal{L}$, so that the complete system satisfies the property φ?

In this paper, properties are described by temporal logic formulae expressed, for the purpose of simplicity, through the Selective Hennessy-Milner logic (Barbuti *et al.*, 1999), and systems by CCS processes (Milner, 1989). If φ is the formula to be satisfied by the complete system, an automatic procedure is defined to identify a formula ψ such that, for each process q satisfying ψ, $(p \mid q)\backslash\mathcal{L}$ satisfies φ. Moreover, the description of the lacking component through a logic formula guarantees correctness of the integration with p of any process that exhibits a behavior compliant with the inferred formula. This behavior consists in a skeleton of communications so that processes having the same skeleton are all able to be successfully integrated with p.

The synthesized formula could be generally obtained with the partial model checking technique presented in (Andersen, 1995), aiming to reduce the model checking problem of a complex process to that of smaller size processes. Andersen's method moves the cause of the possible exponential complexity of the model checking of the concurrent processes with respect to φ from the number of states of the transition system of $(p \mid x)\backslash\mathcal{L}$ to the number of operators of a new formula ψ, which includes also the specification of all possible behaviors of p. Thus, the resulting formula does not highlight the missing behavior in p with respect to φ, and the method does not scale when evolving the system with new components.

Our aim is, even at the price of non-completeness of the method, to define a formula for q, whose com-

Santone A., Luisa Villani M. and Vaglini G. (2009).
A FORMULA DRIVEN INCREMENTAL CONSTRUCTION OF WEB SERVICE COMPOSITIONS.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 13-22
DOI: 10.5220/0002246800130022

plexity depends mainly on the complexity of the original formula φ and of the chosen communication interface; moreover, the result of the model checking of p is taken into account so that ψ includes only the part of φ that is not satisfied by p. Finally, the efficiency of the method is tackled by exploiting the Selective mu-calculus logic and the local model checking methodology; logic and methodology that allow us to consider (and build) only the part of the transition system of p needed for the verification.

In the following section, the basics of the specification language we refer to are recalled, together with the temporal logic through which the system properties are defined. Section 3 shows the core of the approach, while Section 4 presents an application of the methodology through a known example in the field of Web Services. Finally, considerations and comparisons with some related work are given in Section 5.

2 PRELIMINARIES

In a web service composition, the implementation details are hidden, as web services are black box components running on the provider servers, but their interface specifications, described by standard languages like WSDL (W3C Working Group, 2007) and "abstract" WS-BPEL (Andrews *et al.*, 2003), could be public and they are automatically accessible. These descriptions include the incoming/outcoming messages for each service operation and the interaction protocol for their usage. This work considers a set of WS-BPEL processes, which can be translated into CCS processes, and a global formula to be satisfied by the integrated system: the presented approach can be used to deduce the behavioural specification of the missing partner, i.e., a partner providing the operations required by the existing part of the system to correctly satisfy the global formula. Mappings of WS-BPEL constructs to CCS are discussed in (Breugel and Koshkina, 2006) and (Martinelli and Matteucci, 2007), while arguments sustaining CCS modelling of web services compared to, for example, pi-calculus are given in (Bao *et al.*, 2006).

2.1 The Calculus of Communicating Systems

The Calculus of Communicating Systems (CCS) (Milner, 1989) is an algebra suitable for modelling and analyzing processes. The syntax of *processes* is the following:

$$p ::= nil \mid \alpha.p \mid p+p \mid p|p \mid p\backslash L \mid p[f] \mid x$$

where α ranges over a finite set of *visible actions* $\mathcal{V} = \{a, \overline{a}, b, \overline{b}, ...\}$. Input actions are labelled with "non-barred" names, e.g. a, while output actions are "barred", e.g. $\overline{a}$. The set L, in processes with the form $p\backslash L$ ranges over sets of *visible actions*, f ranges over functions from actions to actions, while x ranges over a set of *constant* names: each constant x is defined by a constant definition $x \stackrel{\text{def}}{=} p$.

The *operational semantics* is given in Appendix.

In the following, given a process p, the *sort* of p is the subset of $\mathcal{V}$ containing the actions that p can perform. The reader can refer to (Milner, 1989) for the precise definition of the syntactically based version of the sort of p.

Let $\delta \in \mathcal{A}^*$: if $\delta = \alpha_1 \dots \alpha_n, n \geq 1$, $p \xrightarrow{\delta} q$ means $p \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} q$; if $\delta = \lambda$, where λ is the empty sequence, $p \xrightarrow{\delta} q$ iff $p = q$. A process q such that there is a computation $p \xrightarrow{\delta} q$ is a δ-*derivative* of p with $\longrightarrow$ (or simply a *derivative* of p).

2.2 Model Checking and Selective mu-calculus Logic

In the model checking framework (Clarke *et al.*, 2000), systems are modelled as automata (often called transition systems) and requirements are expressed as formulae of some temporal logic. The selective mu-calculus is a branching temporal logic to express behavioral properties of systems (Barbuti *et al.*, 1999). It is equi-expressive to mu-calculus (Stirling, 1991), but they differ in the definition of the modal operators. Given a set $\mathcal{A}$ of actions and a set *Var* of variables, selective mu-calculus formulae are defined as follows:

$$\begin{array}{rcl} \varphi & ::= & \mathtt{tt} \mid \mathtt{ff} \mid Z \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid [K]_R \varphi \mid \\ & & \langle K \rangle_R \varphi \mid \nu Z.\varphi \mid \mu Z.\varphi \end{array}$$

where $Z \in Var$ and $K, R \subseteq \mathcal{A}$. The operators $\mu Z.\varphi$ and $\nu Z.\varphi$ are fixed point operators: $\mu Z.\varphi$ is the least fixed point of the recursive equation $Z = \varphi$, while $\nu Z.\varphi$ is the greatest one. In the formula $\mu Z.\varphi$ ($\nu Z.\varphi$) μZ (νZ) *binds* the occurrences of Z in φ. A variable that is not bounded by any fixed point operators is called *free*. A formula without free variables is called *closed*. From now on only closed formulae are considered. The precise definition of the satisfaction of the closed formula φ by the process p is given in the Appendix.

An interesting property, which will be used in the successive sections, is expressed by the formula below, where $S \subseteq \mathcal{A}$, $\gamma = \alpha_1 \dots \alpha_n$ and $\alpha_i \in \mathcal{A}$ for all $1 \leq i \leq n$:

$$\begin{aligned} \mathit{event\_seq}(\gamma,S,\psi) = \; & \mathit{eventually}(\alpha_1,S) \wedge [\alpha_1]_{\mathcal{A}} \\ & (\mathit{eventually}(\alpha_2,S) \wedge [\alpha_2]_{\mathcal{A}} (\cdots \wedge \\ & [\alpha_{n-1}]_{\mathcal{A}} (\mathit{eventually}(\alpha_n,S) \wedge \\ & [\alpha_n]_{\mathcal{A}} \psi) \cdots)) \end{aligned}$$

The property says: "the actions of the sequence γ eventually happen (each action is not interleaved with actions in S) and then ψ holds". The property uses the formula $\mathit{eventually}(\alpha,S)$, i.e. "$\alpha$ eventually happens, not preceded by actions in S", whose formal definition is:

$$\begin{aligned} \mathit{eventually}(\alpha,S) = \; & \mu Z. \langle - \rangle_S \mathtt{tt} \wedge \\ & [S - \{\alpha\}]_{\emptyset} \mathtt{ff} \wedge [-\{\{\alpha\} \cup S\}]_S Z. \end{aligned}$$

To the purpose of explaining the methodology without too much technicality, system properties will be defined through the Selective Hennessy-Milner Logic (SHML) instead of the full selective mu-calculus (Barbuti *et al.*, 1999). SHML is more expressive than the Hennessy-Milner logic (Stirling, 1991) because of the intrinsic recursion of the selective operators. The syntax of such logic is:

$$\varphi ::= \mathtt{tt} \mid \mathtt{ff} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [K]_R \varphi \mid \langle K \rangle_R \varphi.$$

3 THE METHOD

Given a process p and a formula φ, a formula ψ is looked for such that the parallel composition of p and q satisfies φ, for each process q satisfying ψ. In fact, the aim of the present work is the integration of the functionality of an existing process p with new ones: such integration has to maintain certain guarantees, expressed by the formula φ, together with the requirements of the new functionalities. Thus, the logic formula ψ supplies the formal specification of the process q and it is thought to force q to support p to meet φ; obviously, the behavior of p might be such that no formula ψ can be deduced to guarantee a solution to the satisfiability problem of φ. Such a case is detected as unsuccessful by the tableau-based algorithm. In particular, we consider a process p offering a communication interface to cooperate with another process q: if such interface is not sufficient to guarantee, for the part involving p, the satisfaction of the formula (i.e. a required communication is not in the interface of p or it exists, but is not correctly performed) then the algorithm stops with failure. Thus the method hypotheses can be recalled: given a CCS process p,

- all actions of p are either communication actions, τ_l, performed inside p or visible actions that are proposed for communicating with q; only the communication actions with q, together with the corresponding dual actions, constitute the set $\mathcal{L} \subseteq sort(p)$;
- in the global formulae, only actions τ_α will occur, both performed inside p or between p and q.

We need also the definition of the following set containing the corresponding actions through which a communication occurs, beyond the internal communication actions performed by p.

Definition 3.1. *Given the set of communication actions $\mathcal{L}$ and $R \subseteq \{\tau_l \mid \tau_l \in \mathcal{A}\}$,*

$$R_\tau^{\mathcal{L}} = \{l, \overline{l} \mid (\tau_l \in R) \wedge (l, \overline{l} \in \mathcal{L})\} \cup \{\tau_l \mid (\tau_l \in R) \wedge (l, \overline{l} \notin \mathcal{L})\}.$$

When clear from the context we use R_τ instead of $R_\tau^{\mathcal{L}}$.

We propose a tableau-based method since such method permits the exploration (and then requires the construction) of only the part of the transition system of the process involved in the verification of a given formula. In our tableau two parts are distinguished: the goal and the environment. Intuitively, at each intermediate stage while producing the solution, the goal says what remains to be done and the environment records the solution produced so far along a branch of the tableau itself. The tableau works on sequents on which a set of rules can be applied; sequents are defined as follows.

Definition 3.2. *A sequent is an expression of the form: $\langle p, \mathcal{L}, x_\psi, \mathcal{B} \rangle \vdash_{\mathcal{E}} \varphi$, s.t.:*

- *φ is a SHML formula;*
- *p is a CCS process (the existing process or one of its derivatives);*
- *$\mathcal{L}$ is the communication interface offered by p;*
- *$\mathcal{E}$ is the environment (i.e. a set of incomplete selective mu-calculus formulae representing the until now performed path on the tree);*
- *x_ψ represents the unknown formula that will be possibly built through the tableau branches as the search will go onward;*
- *$\mathcal{B}$ is a boolean value* **yes** *or* **no***: $\mathcal{B}$ is* **yes** *when the last selective operator, being examined when producing ψ, is a box operator; otherwise $\mathcal{B}$ is* **no**.

Each rule is of the form:

$$\frac{\langle p, \mathcal{L}, x_\psi, \mathcal{B} \rangle \vdash_{\mathcal{E}} \varphi}{\langle p_1, \mathcal{L}, x_{\psi_1}, \mathcal{B} \rangle \vdash_{\mathcal{E}'} \varphi_1 \quad \cdots \quad \langle p_n, \mathcal{L}, x_{\psi_n}, \mathcal{B} \rangle \vdash_{\mathcal{E}'} \varphi_n}$$

where $n > 0$ and side conditions may exist. The premise sequent is the goal to be achieved, the consequents are the sub-goals which are determined by the structure of the formula and by the possible derivatives of p.

Table 1: The Rules.

$$\mathbf{dia}_1 \quad \frac{\langle p,\mathcal{L},x_{\psi_j},\mathbf{no}\rangle \vdash_{\mathcal{E}_i} \langle\tau_l\rangle_R\,\varphi}{\langle p',\mathcal{L},x_{\psi_{j+1}},\mathbf{no}\rangle \vdash_{\mathcal{E}_{i+1}} \varphi} \quad \left\{ \begin{array}{l} p \overset{\gamma\alpha}{\Longrightarrow}_{R_\tau\cup\mathcal{L}} p',\ \alpha\in\{l,\overline{l}\},\ \alpha\in\mathcal{L},\ \gamma\in(\mathcal{L}-R_\tau)^* \\ \mathcal{E}_{i+1}=\mathcal{E}_i\cup\{\psi_j=\langle\overline{\gamma}\rangle_{R_\tau\cup\mathcal{L}}\langle\overline{\alpha}\rangle_{R_\tau\cup\mathcal{L}}\psi_{j+1}\} \end{array} \right\}$$

$$\mathbf{dia}_2 \quad \frac{\langle p,\mathcal{L},x_{\psi_j},\mathbf{no}\rangle \vdash_{\mathcal{E}_i} \langle\tau_l\rangle_R\,\varphi}{\langle p',\mathcal{L},x_{\psi_{j+1}},\mathbf{no}\rangle \vdash_{\mathcal{E}_{i+1}} \varphi} \quad \left\{ \begin{array}{l} p \overset{\gamma\alpha}{\Longrightarrow}_{\{\tau_l\}\cup R_\tau\cup\mathcal{L}} p',\ \alpha=\tau_l,\ l\notin\mathcal{L},\ \gamma\in(\mathcal{L}-R_\tau)^* \\ \mathcal{E}_{i+1}=\mathcal{E}_i\cup\{\psi_j=\langle\overline{\gamma}\rangle_{R_\tau\cup\mathcal{L}}\psi_{j+1}\} \end{array} \right\}$$

$$\mathbf{dia}_3 \quad \frac{\langle p,\mathcal{L},x_{\psi_j},\mathbf{yes}\rangle \vdash_{\mathcal{E}_i} \langle\tau_l\rangle_R\,\varphi}{\langle p',\mathcal{L},x_{\psi_{j+1}},\mathbf{no}\rangle \vdash_{\mathcal{E}_{i+1}} \varphi} \quad \left\{ \begin{array}{l} p \overset{\gamma\alpha}{\Longrightarrow}_{R_\tau\cup\mathcal{L}} p',\ \alpha\in\{l,\overline{l}\},\ \alpha\in\mathcal{L},\ \gamma\in(\mathcal{L}-R_\tau)^* \\ \mathcal{E}_{i+1}=\mathcal{E}_i\cup\{\psi_j=\mathit{event\_seq}(\overline{\gamma},R_\tau\cup\mathcal{L},\langle\overline{\alpha}\rangle_{R_\tau\cup\mathcal{L}}\psi_{j+1})\} \end{array} \right\}$$

$$\mathbf{dia}_4 \quad \frac{\langle p,\mathcal{L},x_{\psi_j},\mathbf{yes}\rangle \vdash_{\mathcal{E}_i} \langle\tau_l\rangle_R\,\varphi}{\langle p',\mathcal{L},x_{\psi_{j+1}},\mathbf{no}\rangle \vdash_{\mathcal{E}_{i+1}} \varphi} \quad \left\{ \begin{array}{l} p \overset{\gamma\alpha}{\Longrightarrow}_{\{\tau_l\}\cup R_\tau\cup\mathcal{L}} p',\ \alpha=\tau_l,\ l\notin\mathcal{L},\ \gamma\in(\mathcal{L}-R_\tau)^* \\ \mathcal{E}_{i+1}=\mathcal{E}_i\cup\{\psi_j=\mathit{event\_seq}(\overline{\gamma},R_\tau\cup\mathcal{L},\psi_{j+1})\} \end{array} \right\}$$

$$\mathbf{box}_1 \quad \frac{\langle p,\mathcal{L},x_{\psi_j},\mathcal{B}\rangle \vdash_{\mathcal{E}_i} [\tau_l]_R\,\varphi}{\langle p_1,\mathcal{L},x_{\psi_{j_1}},\mathbf{yes}\rangle \vdash_{\mathcal{E}_{i+1}} \varphi \cdots \langle p_n,\mathcal{L},x_{\psi_{j_n}},\mathbf{yes}\rangle \vdash_{\mathcal{E}_{i+1}} \varphi} \quad \mathit{condition}$$

$$\mathit{condition} = \left\{ \begin{array}{l} \{p_i \mid p \overset{\alpha}{\Longrightarrow}_{R_\tau\cup\mathcal{L}} p_i,\ 1\le i\le n,\ \alpha\in\{l,\overline{l}\},\ \alpha\in\mathcal{L}\} \\ \mathcal{E}_{i+1}=\mathcal{E}_i\cup\{\psi_j=[\overline{\alpha}]_{R_\tau}(\bigwedge_{k=1\ldots n}\psi_{j_k})\} \end{array} \right\}$$

$$\mathbf{box}_2 \quad \frac{\langle p,\mathcal{L},x_{\psi_j},\mathcal{B}\rangle \vdash_{\mathcal{E}_i} [\tau_l]_R\,\varphi}{\langle p_1,\mathcal{L},x_{\psi_{j_1}},\mathbf{yes}\rangle \vdash_{\mathcal{E}_{i+1}} \varphi \cdots \langle p_n,\mathcal{L},x_{\psi_{j_n}},\mathbf{yes}\rangle \vdash_{\mathcal{E}_{i+1}} \varphi} \quad \mathit{condition}$$

$$\mathit{condition} = \left\{ \begin{array}{l} \{p_i \mid p \overset{\alpha}{\Longrightarrow}_{\{\tau_l\}\cup R_\tau\cup\mathcal{L}} p_i,\ 1\le i\le n,\ \alpha=\tau_l,\ l\notin\mathcal{L}\} \\ \mathcal{E}_{i+1}=\mathcal{E}_i\cup\{\psi_j=(\bigwedge_{k=1\ldots n}\psi_{j_k})\} \end{array} \right\}$$

$$\mathbf{and} \quad \frac{\langle p,\mathcal{L},x_{\psi_j},\mathcal{B}\rangle \vdash_{\mathcal{E}_i} \varphi_1\wedge\varphi_2}{\langle p,\mathcal{L},x_{\psi_{j_1}},\mathcal{B}\rangle \vdash_{\mathcal{E}_{i+1}} \varphi_1 \quad \langle p,\mathcal{L},x_{\psi_{j_2}},\mathcal{B}\rangle \vdash_{\mathcal{E}_{i+1}} \varphi_2} \quad \left\{ \begin{array}{l} \mathcal{E}_{i+1}=\mathcal{E}_i\cup\{\psi_j=\psi_{j_1}\wedge\psi_{j_2}\} \end{array} \right\}$$

$$\mathbf{or}_1 \quad \frac{\langle p,\mathcal{L},x_{\psi_j},\mathcal{B}\rangle \vdash_{\mathcal{E}_i} \varphi_1\vee\varphi_2}{\langle p,\mathcal{L},x_{\psi_j},\mathcal{B}\rangle \vdash_{\mathcal{E}_i} \varphi_1}$$

$$\mathbf{or}_2 \quad \frac{\langle p,\mathcal{L},x_{\psi_j},\mathcal{B}\rangle \vdash_{\mathcal{E}_i} \varphi_1\vee\varphi_2}{\langle p,\mathcal{L},x_{\psi_j},\mathcal{B}\rangle \vdash_{\mathcal{E}_i} \varphi_2}$$

3.1 Tableau Rules

A tableau, i.e., a proof tree, is built starting from a root labelled with the following initial goal

$$\langle p,\mathcal{L},x_{\psi_0},\mathcal{B}_0\rangle \vdash_{\mathcal{E}_0} \varphi.$$

where, for any tableau, $\mathcal{B}_0=\mathbf{no}$, $\mathcal{E}_0=\emptyset$ and ψ_0 is the formula to be obtained through the tableau search.

Then, the construction proceeds by applying specific rules to successively simplify the goal and extend the environment until terminal sequents are reached. Namely, the sequents labelling the immediate successors of a node are determined by the rules in Table 1, while terminal sequents (successful and unsuccessful) are identified in Table 2.

The rules take into account the structure of the formula present in the premise of each sequent (the first modal operator), and some "context" information: specifically, context information regard the last examined logical operator. The rules **and** and **or** specify recursion on one or both the component formulae.

In Table 1 the shorthand $\langle\overline{\gamma}\rangle_S\varphi$ is used to represent the sequence $\langle\overline{\delta}_1\rangle_S\cdots\langle\overline{\delta}_n\rangle_S\varphi$ when $\gamma=\delta_1\cdots\delta_n$ and $n\ge 1$. If $\gamma=\lambda$, then $\langle\overline{\gamma}\rangle_S\varphi$ is equal to φ. The rules **dia** and **box** have a case regarding the actions in the interface $\mathcal{L}$ and one regarding communications inside p; in the case in which p does not perform either the required action in the interface or the internal communication, instead of forcing q to substitute p,

Table 2: Successful/unsuccessful terminal sequents.

success$_1$	$\dfrac{\langle p, \mathcal{L}, x_{\psi_j}, \mathcal{B}\rangle \vdash_{\mathcal{E}_i} \mathtt{tt}}{}$	$\mathcal{E}_{i+1} = \mathcal{E}_i \cup \{\psi_j = \mathtt{tt}\}$
succes$_2$	$\dfrac{\langle p, \mathcal{L}, x_{\psi_j}, \mathcal{B}\rangle \vdash_{\mathcal{E}_i} [\tau_l]_R \varphi}{}$	$\left\{ \begin{array}{l} \{p' \mid p \stackrel{\alpha}{\Longrightarrow}_{\{\alpha\}\cup R_\tau \cup \mathcal{L}} p',\ \alpha \in \{l, \overline{l}, \tau_l\}\} = \emptyset \\ \mathcal{E}_{i+1} = \mathcal{E}_i \cup \{\psi_j = \mathtt{tt}\} \end{array} \right\}$
unsucces$_1$	$\dfrac{\langle p, \mathcal{L}, x_{\psi_j}, \mathcal{B}\rangle \vdash_{\mathcal{E}_i} \mathtt{ff}}{}$	
unsucces$_2$	$\dfrac{\langle p, \mathcal{L}, x_{\psi_j}, \mathcal{B}\rangle \vdash_{\mathcal{E}_i} \langle \tau_l \rangle_R \varphi}{}$	$p \not\stackrel{\gamma\alpha}{\Longrightarrow}_{\{\alpha\}\cup R_\tau \cup \mathcal{L}} p',\ \alpha \in \{l, \overline{l}, \tau_l\}, \gamma \in (\mathcal{L} - R_\tau)^*$

we have chosen to produce a failure of the algorithm to represent, in some sense, a failure of the formula verification. Now the rules are explained by case analysis:

- **dia**$_1$**, dia**$_2$**.** These rules are applicable when the previously examined logical operator is not a box one. The rule **dia**$_1$ considers the action α be an offered communication, while **dia**$_2$ considers α as a communication inside p. If α is preceded by a sequence γ of actions not in R_τ, q must perform an equal sequence of corresponding dual actions; obviously, if $\alpha = \tau_l$ no corresponding action is required of q. Note that when more than one move exists for p, each one produces the specification of a different set of candidate processes q. It is out of the scope of this paper the definition of a possible strategy for choosing the "most suitable" set of candidate processes.
- **dia**$_3$**, dia**$_4$**.** When the last examined formula operator is a box one, the sequence $\overline{\gamma}$ of actions must be performed in all paths of q before performing α, if different from τ_l. Otherwise only the sequence $\overline{\gamma}$ must be performed in all paths. In such a way the connected behaviors of p and q are synchronized.
- **box**$_1$**, box**$_2$**.** These rules take account of the ability of p of performing k actions α: for each one a branch of the tableau is opened to verify the formula φ.
- **and.** The rule says that the constructions of the two sub-formulae are carried on separately, and the results are composed.
- **or**$_1$ and **or**$_2$: straightforward.

A sequent $S = \langle p, \mathcal{L}, x_{\psi_j}, \mathcal{B}\rangle \vdash_{\mathcal{E}_i} \varphi$, S can be either a successful or an unsuccessful terminal. The successful/unsuccessful terminals are clearly defined in Table 2. A tableau is successful if it is finite and all of its leaves are successful terminals. If $\langle p_1, \mathcal{L}, x_{\psi_{j_1}}, \mathcal{B}\rangle \vdash_{\mathcal{E}_{i_1}} \varphi_1 \cdots \langle p_n, \mathcal{L}, x_{\psi_{j_n}}, \mathcal{B}\rangle \vdash_{\mathcal{E}_{i_n}} \varphi_n$ are all the leaves of the tableau for the goal $\langle p, \mathcal{L}, x_{\psi_0}, \mathbf{no}\rangle \vdash_{\mathcal{E}_0} \varphi$, and they are also successful terminals, then the solution is the formula ψ_0, contained in all the environments of the leaves and recursively obtained by substituting the right hand side of each equation $\psi_k = \psi'$ (taken from any terminal environment) each time ψ_k exists in some environment. When ψ' is a SHML formula the procedure terminates. It is worth noting that, while the name ψ_k can appear more than once in the terminal environments, its definition is unique, i.e., in only one leaf environment we have $\psi_k = \psi'$.

The following theorem states the soundness of our approach.

Theorem 3.1. *Consider a CCS process p and a SHML formula φ. Any tableau for the goal $\langle p, \mathcal{L}, x_{\psi_0}, \mathcal{B}_0\rangle \vdash_{\mathcal{E}_0} \varphi$, defines the formula ψ_0 such that: $q \models \psi_0 \Longrightarrow (p \mid q)\backslash L \models \varphi$.*

Proof. *The proof can be done by induction on the length of the formula φ.*

4 AN APPLICATION OF THE METHODOLOGY

In the design of a web service composition, the implementation details of the candidate component services are hidden, but, attached to their WSDL interface descriptions, one may luckily have their "abstract" WS-BPEL processes (Andrews *et al.*, 2003), representing the interaction protocol for their usage. Indeed, this assumption is in line with the facet-based publication process supported by the SeCSE platform, an outcome of the European project SeCSE (Di Penta *et al.*, 2008). In this context, given a set of processes, described, for example in WS-BPEL, which we can translate into CCS processes, and a global formula to be satisfied by the integrated system (that one can finally realize as a choreography in the WS-CDL (W3C Working Group, 2005) specification language), the approach can be used to deduce the behavioral specification of the missing partner, i.e., a partner providing the operations required by the existing part of the

system, to correctly satisfy the global formula.

As an example, let us consider the following scenario, described in (Bertoli *et al.*, 2007), which we have slightly modified to realize it as a choreography: *The employee of a firm is organizing a work trip. He presents a ticket request to the employer's administration, complete with ticket details. He has gathered these data by first interacting with on-line information service. The administration will either accept or refuse such a proposal, and in the former case, it will try to pay for the ticket of the employee, either by credit card, or by cheque, through some payment service. The employee will eventually get the ticket, or a cancel message by the administration due to payment problems, or else a refusal.* The aim is to provide the firm with a service-centric system that automates this process. The authors had identified five services that could be used:

- the *Administration* service: implementing the firm internal process of employees' work trip management;
- the *InfoTrains* and *InfoFlights* services: providing tickets information over train/flight routes, and ticket booking/buying services; and
- the *BuyCheque* and *BuyCard* services: enabling payment by cheque or card respectively.

We added the *Travel* service as the interface with the user. The main requirement for this composition is flexibility with respect to the services that will be actually used at run-time, that is, possibility to seamlessly replace a service with another one (e.g., the transport or payment services). Examples of CCS descriptions of "skeleton" services for this scenario are given in Table 3.

Supposing fixed the Travel and Administration services interfaces, any compatible transport service must provide: (i) an operation to search for available seats with respect to the data provided in the query; (ii) an operation for booking a selected travel, without further data (e.g., payment information); (iii) an operation to be notified of the payment. Instead, a payment service consists of an operation to accept the data for the specific payment mean (either card or cheque), and it is required to notify the interested service of the payment in case of success, or to send an error message back to the requestor.

We show how the presented approach can be applied to automatically derive properties to be satisfied by any transport and payment services, given some global objectives to be ensured by the composition. Let us first consider the following formula:

$$\begin{aligned}\varphi_1 &= \langle \tau_{fSearch}\rangle_{\emptyset}\,\mathtt{tt} \wedge [\tau_{fSearch}]_{\emptyset}\,\langle \tau_{book}\rangle_{\emptyset} \\ &\quad (\langle \tau_{buy}\rangle_{\emptyset}\,\mathtt{tt} \vee [-]_{\{\tau_{fSearch}\}}\,\mathtt{ff})\end{aligned}$$

which says: *it is possible to search for flights and whenever this operation is required, it is possible to first book a travel and then either buying the ticket or to start another search.*

Suppose that we want this formula to hold for the composition: $S \stackrel{\text{def}}{=} (TS|AD|TInfo|X|BC|BCH)\backslash L$ where X is the missing service interface to be specified, providing the flights information, and L contains all the actions of the CCS process components.

The application of the approach leads to the following (sub)formula providing a characterization of the flights service:

$$\begin{aligned}\psi_1 &= \langle fSearch\rangle_{\emptyset}\,\mathtt{tt} \wedge [fSearch]_L\,\psi_{11} \\ \psi_{11} &= event\_seq(\overline{fSearchStarted}. \\ &\quad \overline{flights}, L, \langle book\rangle_L\,\psi_{12}) \\ \psi_{12} &= \langle buy\rangle_L\,\mathtt{tt} \vee [-]_{\{fSearch\}\cup L}\,\mathtt{ff}\end{aligned}$$

The formula, other than requiring the booking and buying operations from the service to add, ensures that the communication with that service is correct, that is, the booking operation is actually reachable. We note that the FInfo process in Table 3 satisfies ψ_1, and so it is a solution for X, whereas the following process does not:

$$\begin{aligned}FInfoBad &\stackrel{\text{def}}{=} fSearch.\overline{fsearchStarted}. \\ &\quad \overline{flights}.(FInfoBad + BOOKBad) \\ BOOKBad &\stackrel{\text{def}}{=} book.(buy.\overline{ticket}.FInfoBad + \\ &\quad cancel.FInfoBad)\end{aligned}$$

as it requires to explicitly cancel the booking. Now suppose that we want to replace the service *BC* with another one, without affecting the composition. Thus, we consider the system:

$$S' \stackrel{\text{def}}{=} (TS|AD|TInfo|FInfo|X|BCH)\backslash L$$

where X is the missing card payment service, and the global formula:

$$\varphi_2 = \langle \tau_{ticket}\rangle_{\emptyset}\,\mathtt{tt} \wedge \langle \tau_{cancel}\rangle_{\emptyset}\,\mathtt{tt}$$

which says: *either the user will finally get the ticket or the selected trip is cancelled*, that is, both results must be possible. We note that none of the actions of the formula are required from the missing service, that only needs to provide a correct interaction protocol in S' to the satisfaction of φ_2. In this case, the following formula is derived, satisfied by the service *BC* in Table 3:

$$\begin{aligned}\psi_2 &= \langle cardPay\rangle_L\,\langle \overline{receipt}\rangle_L\,\langle \overline{buy}\rangle_L\,\mathtt{tt} \\ &\quad \wedge \langle cardPay\rangle_L\,\langle \overline{invalid}\rangle_L\,\mathtt{tt}.\end{aligned}$$

Table 3: Travel management choreography.

Travel Service

$TS \stackrel{\text{def}}{=} FClient + TClient$

$FClient \stackrel{\text{def}}{=} \overline{fSearch}.fsearchStarted.flights.(BOOK + TS)$

$TClient \stackrel{\text{def}}{=} \overline{tSearch}.tsearchStarted.trains.(BOOK + TS)$

$BOOK \stackrel{\text{def}}{=} \overline{book}.(bookOK.ADClient + bookKO.TS)$

$ADClient \stackrel{\text{def}}{=} \overline{request}.(accepted.(ticket.TS + cancel.TS) + rejected.TS)$

Administration Service

$AD \stackrel{\text{def}}{=} request.(\overline{accepted}.(CPClient + CHPClient) + \overline{rejected}.AD)$

$CPClient \stackrel{\text{def}}{=} \overline{cardPay}.(receipt.AD + invalid.\overline{cancel}.AD)$

$CHPClient \stackrel{\text{def}}{=} \overline{chequePay}.(receipt.AD + invalid.\overline{cancel}.AD)$

Payment Services

$BC \stackrel{\text{def}}{=} cardPay.(\overline{receipt}.\overline{buy}.BC + \overline{invalid}.BC)$

$BCH \stackrel{\text{def}}{=} chequePay.(\overline{receipt}.\overline{buy}.BCH + \overline{invalid}.BCH)$

Transport Services

$FInfo \stackrel{\text{def}}{=} fSearch.\overline{fsearchStarted}.\overline{flights}.(FInfo + book.(\overline{bookOK}.FBUY + \overline{bookKO}.FInfo))$

$FBUY \stackrel{\text{def}}{=} buy.\overline{ticket}.FInfo + FInfo$

$TInfo \stackrel{\text{def}}{=} tSearch.\overline{tsearchStarted}.\overline{trains}.(TInfo + book.(\overline{bookOK}.TBUY + \overline{bookKO}.TInfo))$

$TBUY \stackrel{\text{def}}{=} buy.\overline{ticket}.TInfo + TInfo$

5 CONCLUSIONS AND RELATED WORK

In this paper, given an incomplete system (say p) and a requirement described by the logic formula φ, an automatic procedure is defined to identify a formula ψ such that, for each q satisfying ψ, we have that the parallel composition between p and q satisfies φ. For the sake of clarity, only requirements expressed in the Selective Hennessy-Milner Logic (SHML) are used. The extension to full selective mu-calculus can be easily defined. The procedure can be incorporated in an implementation of a model checker for the Concurrency Workbench of the New Century (CWB-NC) (Cleaveland and Sims, 1996), a tool for the automated analysis of concurrent systems.

In (Andersen, 1995), an automatic method is proposed, sound and complete for the full mu-calculus, able to determine the formula ψ; such method always includes in ψ all the possible behaviors of p, so producing a formula whose complexity depends on the number of states of p. Our aim is, even at the price of non-completeness of the method, to define a more efficient formula ψ for q, that is a formula containing only the corresponding actions of the incomplete communications of p. Indeed, simplicity of the derived property ψ and scalability of the verification process, are necessary for applying the method to both incremental design and system evolution scenarios where p is already in place, realizing some functionality, and one needs to understand the specification of the functionality of the new component that would behave correctly with p. Scalability problems are also tackled, in our work, by using a local model checking methodology combined with the SHML that provides an abstraction technique as shown in (Barbuti *et al.*, 1999).

The formal problem we face in this paper could

be alternatively solved under the well known assume-guarantee theoretical framework. This provides inference rules that permit to deduce the global validity of a formula for a system, by verifying correctness of a given component under a set of assumptions on the environment (a report on this technique is contained in (Furia, 2005)). The most difficult part here is the generation of the assumptions, a job that for long has been left (at least partially) to the developer ((Pasareanu *et al.*, 1999), (Inverardi *et al.*, 2000)). Starting from the paper (Giannakopoulou *et al.*, 2002), a number of works use a learning algorithm for regular languages (Angluin, 1987) to automatically derive the weakest assumptions for the component at hand to satisfy a safety property. This approach requires both the component and the formula be modelled as deterministic finite state machines and model checking is used iteratively (until convergence of the algorithm) to identify states and transitions of the environment. In this respect, our approach, that works for all properties expressible in SHML (both safety and liveness) and includes non-determinism, is more efficient as it is based on local model checking and does not even require to construct the state transition system of the component.

In the service-oriented computing area, formal methods have been used to define unambiguous semantics for the languages WS-BPEL and WS-CDL, to describe service compositions and interaction protocols (called *choreography*). An overview of the various formalisms proposed, including process algebras, is contained in (Breugel and Koshkina, 2006). Once a formal model of the system is available, one can check properties such as deadlock-freeness and correctness of conversations with the services (see (Fu *et al.*, 2005) and (Kazhamiakin *et al.*, 2006)). Conversely, given a set of service interfaces and a choreography to be realized, one may ask whether service behaviors may be deduced generating conversations that, at global level, are all admissible by the choreography. In (Fu *et al.*, 2005), sufficient conditions for realizability of a choreography are given, and the service behaviors are deduced through projection of the global conversations (i.e., removing messages that do not involve the specific service). In our work, we are given a partial choreography already ”realized” that needs to be extended with an additional service, so to satisfy a global requirement expressed by a SHML formula. Our method allows us to eventually deduce another formula that is used to discover a class of service implementations all able to complete the realization of the extended choreography. In (Lohmann *et al.*, 2007) the authors propose to attach an operational description to a service *P*, automatically computed, characterizing services whose composition with *P* is deadlock-free or satisfies specific behavioral constraints. Finally, the Open Workflow Nets formalism, a special class of Petri Nets, is used both to describe the processes and the constraints. As we consider all properties that can be expressed in SHML, our approach is more general, and we can check constraints satisfaction by model checking. Finally, the paper (Martinelli and Matteucci, 2007) presents a simplified version of Andersen’s partial model checking algorithm with the aim of applying it to the definition of web service orchestrations: given a parallel composition of processes, all known, the specification of the orchestrator is deduced. They may avoid the formula explosion of the original Andersen’s method as they just need to generate a process, containing only communication actions, to make sure that these happen in the right order. Indeed, differently from Andersen’s and ours, their method only works if the construction of the transition system of the parallel processes is feasible. As a future work, we intend to develop a service discovery tool integrating the approach and analyze its efficiency and usefulness compared to the existing methods.

REFERENCES

Andersen, H. R. (1995). Partial Model Checking (Extended Abstract). In *LICS’95, Proc. 10th Annual IEEE Symposium on Logic in Computer Science, San Diego, California, USA, 26-29 June*. IEEE. 398–407.

Andrews, T. and Curbera, F. and Dholakiam, H. and Goland, Y. and Klein, J. and Leymann, F. and Liu, K. and Roller, D. and Smith, D. and Thatte, S. and Trickovic, I. and Weerawarana, S. Business Process Execution Language for Web Services. (http://www.ibm.com/developerworks/library/specification/ws-bpel/).

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation* 75(2). 87–106.

Bao, L. and Zhang, W. and Zhang, X. (2006). Describing and Verifying Web Service Using CCS. In *PDCAT06, Seventh Int. Conf. on Parallel and Distributed Computing, Applications and Technologies, Washington, DC, USA*. IEEE. 421–426.

Barbuti, R. and De Francesco, N. and Santone, A. and Vaglini, G. (1999). Selective mu-calculus and Formula-Based Abstractions of Transition Systems. *Journal of Computer and System Sciences* 59(3). 537–556.

Bertoli, P. and Hoffmann, J. and Freddy, L. and Pistore, M. (2007). Integrating Discovery and Automated Composition: from Semantic Requirements to Executable Code. In *ICWS 2007, International Conference on*

Web Services, Salt Lake City, Utah, USA, July 9-13 2007. IEEE. 815–822.

Breugel, F. and Koshkina, M. (2006). Models and verification of BPEL. (http://www.cse. yorku.ca/ franck/research/drafts/ tutorial.pdf)

Clarke, E.M. and Grumberg, O. and Peled, D. (2000). *Model Checking*. MIT press.

Cleaveland, R. (1989). Tableau-Based Model Checking in the Propositional Mu-Calculus. *Acta Informatica* 27(8). 725–747.

Cleaveland, R. and Sims, S. (1996). The NCSU Concurrency Workbench. In *CAV'96, Eighth International Conference on Computer-Aided Verification*. Lecture Notes in Computer Science 1102. 394–397.

Di Penta, M. and Bastida, L. and Sillitti, A. and Baresi, L. and Ripa, G. and Melideo, M. and Tilly, M. and Spanoudakis, G. and Maiden, N. and Gorroogoitia Cruz, J. and Hutchinson, J. (2008). SeCSE - Service Centric System Engineering: an overview. In *At your service: Service Engineering in the Information Society Technologies Program*. MIT Press. ISBN: 978-0-262-04253-6.

Fu, X. and Bultan, T. and Su J. (2005). Synchronizability of Conversations among Web Services. *IEEE Trans. Software Eng.*, 31(12). 1042–1055.

Furia, C.A. (2005). A compositional world: a survey of recent works on compositionality in formal methods. *Technical Report 2005.22, Dipartimento di Elettronica e Informazione, Politecnico di Milano*.

Giannakopoulou, D. and Pasareanu, C. and Barringer, H. (2002). Assumption Generation for Software Component Verification. In *ASE 2002, 17th International Conference on Automated Software Engineering, 23-27 September 2002, Edinburgh, Scotland, UK*. 3–12.

Inverardi, P. and Yankelevich, D. and Wolf, A. L. (2000). Static Checking of Systems Behaviors Using Derived Component Assumptions. *ACM Transactions on Software Engineering and Methodology* 9(3). 239–272.

Kazhamiakin, R. and Pistore, M. and Santuari, L. (2006). Analysis of communication models in web service compositions. In *WWW'06, 15th international conference on World Wide Web*. 267–276.

Lohmann, N. and Massuthe, P. and Wolf, K. (2007). Behavioral Constraints for Services. In *BPM 2007, 5th International Conference on Business Process Management, Brisbane, Australia, Sept. 24-28*. Lecture Notes in Computer Science 4714. Springer. 271–287.

Martinelli, F. and Matteucci, I. (2007). Synthesis of Web Services Orchestrators in a Timed Setting. In *WS-FM 2007, 4th International Workshop on Web Services and Formal Methods, Sept. 28-29*. Lecture Notes in Computer Science 4937. Springer. 124–138.

Milner, R. (1989). *Communication and Concurrency*. Prentice-Hall.

Pasareanu, C. and Dwyer, M. and Huth, M. (1999). Assume-guarantee model checking of software: A comparative case study. In *6th SPIN Workshop*. Lecture Notes in Computer Science 1680. 168–183.

Stirling, C. (1991). An Introduction to Modal and Temporal Logics for CCS. In *UK/Japan workshop on Concurrency : theory, language, and architecture, Oxford, UK*. Springer-Verlag. 2–20.

W3C Working Group (2005). Web Services Choreography Description Language Version 1.0. (http://www.w3.org/TR/ws-cdl-10/).

W3C Working Group (2007). Web Services Description Language (WSDL) Version 2.0. (http://www.w3.org/TR/wsdl20-primer/).

APPENDIX

Operational semantics of CCS.

The *operational semantics* (see for the standard version (Milner, 1989)) is given by the relation $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{P}$, where $\mathcal{A}$ is the set $\{a, \overline{a}, \tau_a, b, \overline{b}, \tau_b, ...\}$; this relation is the least one defined by the rules in Table 4 (we omit the symmetric rule of **Sum** and **Par**).

Since we consider abstract WS-BPEL process, where all actions are communications, we modify the standard CCS semantic rule for the operator " $|$ " so that the produced action be τ_l, different for each visible action l. These new visible actions can be used in the global formulae. Each relabelling function f has the property that $f(\tau_l) = \tau_l$ for each visible action l.

Satisfaction of a Selective mu-calculus Logic Formula.

The precise definition of the satisfaction of the closed formula φ by the process p is given in Table 5 where the transition relation $\Longrightarrow_I$, parametric with respect to $I \subseteq \mathcal{A}$, is used. By $p \stackrel{\alpha}{\Longrightarrow}_I q$ we express the fact that it is possible to pass from p to q by performing a (possibly empty) sequence of actions not belonging to I (i.e., non-interesting actions belonging to $\mathcal{A} - I$) and then the action α in I. Note that $\Longrightarrow_{\mathcal{A}} = \longrightarrow$.

Definition 5.1. *Let p be a CCS process and $I \subseteq \mathcal{A}$, for each $\alpha \in I$, $p \stackrel{\alpha}{\Longrightarrow}_I q$ iff $p \stackrel{\gamma\alpha}{\longrightarrow} q$, for some $\gamma \in (\mathcal{A} - I)^*$.*

From now on, the following abbreviations will be used:

$$\begin{aligned} [\alpha_1, \ldots, \alpha_n]_R\varphi &= [\{\alpha_1, \ldots, \alpha_n\}]_R\varphi \\ [-]_R\varphi &= [\mathcal{A}]_R\varphi \\ [-K]_R\varphi &= [\mathcal{A} - K]_R\varphi \end{aligned}$$

Example 5.1. *Some examples of selective mu-calculus formulae are given.*
$\varphi_1 = \langle b \rangle_{\{c\}}\, \mathtt{tt}$*: "it is possible to perform b without performing c before";*
$\varphi_2 = \nu Z.\, [a]_\emptyset (Z \wedge [a]_{\{c\}}\, \mathtt{ff})$*: "it always holds that two successive occurrence of a are not possible if not interleaved by an occurrence of c".*

Table 4: Operational semantics of CCS.

$$\textbf{Act}\ \frac{}{\alpha.p \xrightarrow{\alpha} p} \qquad \textbf{Sum}\ \frac{p \xrightarrow{\alpha} p'}{p+q \xrightarrow{\alpha} p'} \qquad \textbf{Con}\ \frac{p \xrightarrow{\alpha} p'}{x \xrightarrow{\alpha} p'}\ x \stackrel{\text{def}}{=} p$$

$$\textbf{Par}\ \frac{p \xrightarrow{\alpha} p'}{p|q \xrightarrow{\alpha} p'|q} \qquad \textbf{Com}\ \frac{p \xrightarrow{l} p',\ q \xrightarrow{\overline{l}} q'}{p|q \xrightarrow{\tau_l} p'|q'} \qquad \textbf{Rel}\ \frac{p \xrightarrow{\alpha} p'}{p[f] \xrightarrow{f(\alpha)} p'[f]}$$

$$\textbf{Res}\ \frac{p \xrightarrow{\alpha} p'}{p\backslash L \xrightarrow{\alpha} p'\backslash L}\quad \alpha, \overline{\alpha} \notin L$$

Table 5: Satisfaction of a closed formula by a process.

$p \not\models \mathtt{ff}$		$p \models \mathtt{tt}$
$p \models \varphi \wedge \psi$	iff	$p \models \varphi$ and $p \models \psi$
$p \models \varphi \vee \psi$	iff	$p \models \varphi$ or $p \models \psi$
$p \models [K]_R \varphi$	iff	$\forall p'\ \forall \alpha \in K\ p \stackrel{\alpha}{\Longrightarrow}_{K \cup R} p'$ implies $p' \models \varphi$
$p \models \langle K \rangle_R \varphi$	iff	$\exists p'\ \exists \alpha \in K . p \stackrel{\alpha}{\Longrightarrow}_{K \cup R} p'$ and $p' \models \varphi$
$p \models \nu Z.\varphi$	iff	$p \models \nu Z^n.\varphi$ for all n
$p \models \mu Z.\varphi$	iff	$p \models \mu Z^n.\varphi$ for some n

where, for each n, $\nu Z^n.\varphi$ and $\mu Z^n.\varphi$ are defined as:

$\nu Z^0.\varphi = \mathtt{tt}$ $\qquad$ $\mu Z^0.\varphi = \mathtt{ff}$

$\nu Z^{n+1}.\varphi = \varphi[\nu Z^n.\varphi/Z]$ $\qquad$ $\mu Z^{n+1}.\varphi = \varphi[\mu Z^n.\varphi/Z]$

and $\varphi[\psi/Z]$ indicates the substitution of ψ for each free occurrence of Z in φ.