Timed Transition Discovery from Web Service Conversation Logs

Didier Devaurs[1], Kreshnik Musaraj[2], Fabien De Marchi[2], and Mohand-Saïd Hacid[2]

[1] University of Windsor, School of Computer Science, Windsor, Ontario, Canada
ddevaurs@uwindsor.ca

[2] Université Claude Bernard Lyon 1, LIRIS, UMR CNRS 5205, Villeurbanne, France
{kreshnik.musaraj, fabien.demarchi, mohand-said.hacid}@liris.cnrs.fr

Abstract. Despite their importance, Web service business protocols are not always published with service interfaces, which hinders automatic management. A solution is to extract them from past executions. One of the raised issues is the discovery of temporal constraints called *timed transitions*, which are not explicitly recorded. In this paper we present our approach for discovering such transitions. We define a class of patterns called *proper timeouts* which are equivalent to timed transitions, and present a polynomial algorithm for extracting these patterns.[3]

Keywords: Web service, business protocol, knowledge extraction, temporal constraint

1 Introduction

A very important ambition associated with Web services relates to loosely-coupled integration, which is already partially carried out by the fact that services use widespread standards. A good flexibility is possible only if users know how to interact with a service. This requires to associate with services elaborate descriptions (such as WSDL) enabling a good understanding of their execution semantics. However descriptions like WSDL are not sufficient for a sophisticated and automatic use of services because they provide only static properties [1]. This is what motivated authors in [1] to define a higher level model, the so-called *business protocol*, which specifies the conversations supported by a service, i.e. all valid sequences of message exchanges. It is formalized by a deterministic finite-state machine, where states represent the various service phases; transitions are triggered when the service sends or receives messages. A *timed business protocol* [2] is an enhanced version of the basic model allowing for the definition of *timed transitions*, which are not related to the emission of explicit messages but to temporal constraints (validity period, expiration date, etc); they are triggered automatically after a time interval is elapsed or after some date is reached.

[3] This work is partially funded by the ANR project Service Mozaïc (2007–2009, JCJC06_134393) and by the EU Framework 7 STREP project COMPAS (215175, FP7-ICT-2007-1).

Business protocols offer automatic reasoning mechanisms with many applications, such as correctness verification, compatibility testing, etc. However they are not often specified in real life services. Potential reasons include lack of time during implementation or uncontrolled evolution. A solution is then to infer this protocol from the *conversation logs* of a service. Direct applications are re-engineering issues, such as implementation correctness checking or service evolution. Once automated this extraction process could be applied in service discovery architectures [3] for automatic service composition or replacement.

Discovering service protocols includes many technical challenges: cleaning logs from "noise", identifying the different conversations, defining assessable models, developing refining tools for an interactive extraction, etc. The first contribution to this problem has been proposed in [4], but relates only to *untimed* business protocols. With the importance of temporal aspects in real life services it becomes crucial to extend this work to timed business protocols, which contain both explicit and timed transitions.

This paper presents our approach for extracting timed transitions from conversation logs.[4] We define a class of patterns called *proper timeouts* which reveal the presence of timed transitions in the protocol. We propose a characterization of the set of proper timeouts satisfied by the logs, which leads to a polynomial extraction algorithm. This work is an extension of [4], and both take part in *ServiceMosaic* international project (http://servicemosaic.isima.fr) which aims at developing a platform for modeling, analysing and managing Web services [6].

2 Associating Patterns with Timed Transitions

We define an **episode** as a sequence of two message names. Given an episode $\alpha = \langle m, m' \rangle$, an **occurrence** of α is a sequence of two consecutive occurrences of m and m' in the logs. The occurrence duration of an episode occurrence is the difference between the message timestamps. The minimal (respect. maximal) occurrence duration of an episode is the smallest (respect. greatest) occurrence duration of all its occurrences. The **occurrence duration interval** (ODI) of an episode is the interval which includes all its occurrence durations. The minimal (respect. maximal) occurrence duration of a set of episodes is the minimum (respect. maximum) of all the minimal (respect. maximal) occurrence durations of these episodes. The occurrence duration interval (ODI) of a set of episodes is the interval which includes all the occurrence durations of these episodes. For each message m, we denote by P_m the set of episodes whose first message is m.

Given two sets of episodes A and B, we say that A **precedes** B (denoted by $A \prec B$) if $ODI(A)$ is before $ODI(B)$.[5] We say that A and B are not comparable (denoted by $A \parallel B$) if $A \not\prec B$ and $B \not\prec A$. Given $A, B \subset P_m$, we show that: if there exists a timed transition between the state from which the transitions corresponding to the elements of A are going out, and the one from which the

[4] Technical results are presented in an extended version of this paper [5].

[5] $\prec$ is a strict order relation on sets of episodes.

transitions corresponding to the elements of B are going out, then $A \prec B$.

We define a **proper timeout** as a triplet $PT(m, A, B)$, where m is a message and $A, B \subset P_m$. We say that logs L satisfy the proper timeout $PT(m, A, B)$, which is denoted by $L \vDash PT(m, A, B)$, if:

$$\begin{cases} A \prec B \\ \forall\, \alpha \in P_m \setminus (A \cup B),\ \{\alpha\} \nparallel A \cup B \\ \forall\, Z \in \{A, B\},\ \forall\, X, Y \subset Z\ (X, Y \neq \phi),\ (X \cup Y = Z) \Rightarrow (X \not\prec Y)\ . \end{cases} \tag{1}$$

Given a message m and $A, B \subset P_m$, we show that: if there exists a timed transition in the protocol, between two states s_1 and s_2 such that the sets of transitions going out of s_1 and s_2 respectively are in bijection with A and B, then there exist $A' \subseteq A$ and $B' \subseteq B$ such that $L \vDash PT(m, A', B')$. Since each timed transition involves the satisfaction of a proper timeout, we can find all of them. However we can discover more proper timeouts than there are timed transitions, if some messages always take longer to be sent or received than messages associated with other transitions of the same state. Thus we will say that: a satisfied proper timeout reveals the presence of a *potential* timed transition.

We show that: for practical purposes, proper timeouts are the best possible representations of timed transitions. That justifies the relevance of the development of a timed transition discovery method based on the research of the proper timeouts satisfied by the logs.

3 Extracting the Proper Timeouts

The complexity of a basic "generate and test" method for extracting proper timeouts is exponential. Instead, we propose a nice characterization of the set of satisfied proper timeouts, which leads to a polynomial algorithm. This characterization, formalized by Theorem 1, states that: the proper timeouts satisfied by the logs and related to message m are exactly given by the pairs of consecutive elements of the partition of P_m satisfying (2). Thus, partitioning all sets P_m gives us all the proper timeouts satisfied by the logs.

Theorem 1. *Consider a mesage m, $i_m \in \mathbb{N}^*$, and $\{P_m^{(1)}, P_m^{(2)}, \ldots, P_m^{(i_m)}\}$ a partition of P_m. The following assertions are equivalent:*

$$\begin{cases} P_m^{(1)} \prec P_m^{(2)} \prec \ldots \prec P_m^{(i_m)} \\ \forall\, 1 \leq i \leq i_m,\ \forall\, X, Y \subset P_m^{(i)}\ (X, Y \neq \phi),\ (X \cup Y = P_m^{(i)}) \Rightarrow (X \not\prec Y)\ . \end{cases} \tag{2}$$

$$\begin{cases} \forall\, 1 \leq i < i_m,\ L \vDash PT(m, P_m^{(i)}, P_m^{(i+1)}) \\ \forall\, A, B \subset P_m,\ L \vDash PT(m, A, B) \Rightarrow \exists\, 1 \leq i < i_m,\ A = P_m^{(i)},\ B = P_m^{(i+1)}\ . \end{cases} \tag{3}$$

We propose a polynomial algorithm, called *partition*P_m, for constructing this partition in an incremental way. The input of algorithm *partition*P_m comprises

a message m, the set P_m, and the ODIs of all episodes in P_m. The output is the partition Π of P_m satisfying (2). Π is constructed by inserting one by one the elements of P_m in such a way that (2) is satisfied at each step. In order to describe the general step of the algorithm, let us consider that Π is already partly constructed. Let α be an episode of P_m not yet considered. A single pass is made over the partition to determine (i) whether the ODIs of some elements of Π overlap $ODI(\alpha)$, and (ii) between which sets of Π α is situated according to $\prec$. If there is no overlap, a new set containing α is created and inserted into the partition in compliance with $\prec$. If the overlap takes place with only one element of Π, α is simply inserted in this set. If the overlap occurs between α and several parts of Π, they are necessarily consecutive according to $\prec$; as such they are merged and α is inserted into the resulting set. As for each episode $\alpha \in P_m$ only one pass is made over the partition, the complexity is $\mathcal{O}(|P_m|^2)$.

The global method for extracting all the proper timeouts satisfied by the logs is divided in two steps. The first one is a preprocessing of the data, performed in order to obtain the set of messages, the set of episodes, and the ODIs of all episodes. A single pass is made over the logs, during which the occurrence duration of each sequence of two consecutive messages is calculated. The second step consists in constructing all sets P_m, and running algorithm $partitionP_m$ for each of them. The logs' size being far greater than the number of episodes, the first step is the most costly in term of running time. Thus the complexity of the global algorithm is $\mathcal{O}(|L|)$.

We have implemented our discovery process to test its scalability. In order to easily have a big amount of data, we have also implemented a log generator which creates conversation logs from a given business protocol by mimicking the behaviour of a service. Results of our experiments confirm the complexity results we have established formally. The final test will be to run our algorithm on real-life data in further experiments.

References

1. Benatallah, B., Casati, F., Toumani, F.: Representing, analysing and managing web service protocols. Data & Knowledge Engineering **58**(3) (2006) 327–357
2. Ponge, J., Benatallah, B., Casati, F., Toumani, F.: Fine-grained compatibility and replaceability analysis of timed web service protocols. In: ER '07. (2007) 599–614
3. Denaro, G., Pezzé, M., Tosi, D., Schilling, D.: Towards self-adaptive service-oriented architectures. In: TAV-WEB '06, Portland, Maine, USA, ACM (2006) 10–16
4. Motahari Nezhad, H.R., Saint-Paul, R., Benatallah, B., Casati, F.: Protocol discovery from imperfect service interaction logs. In: ICDE '07. (2007) 1405–1409
5. Devaurs, D., Musaraj, K., De Marchi, F., Hacid, M.S.: Timed transition discovery from web service conversation logs (extended version). Technical Report RR-LIRIS-2008-007, LIRIS UMR 5205 CNRS/Université Claude Bernard Lyon 1, Villeurbanne, France (2008) http://liris.cnrs.fr/publis/?id=3369.
6. Benatallah, B., Casati, F., Toumani, F., Ponge, J., Motahari Nezhad, H.R.: Service mosaic: A model-driven framework for web services life-cycle management. IEEE Internet Computing **10**(4) (2006) 55–63