Runtime verification competition 2014: Specification package

Domenico Bianculli[1], Carlo Ghezzi[2], and Srđan Krstić[2]

[1] SnT Centre - University of Luxembourg, Luxembourg
domenico.bianculli@uni.lu
[2] DEEP-SE group - DEIB - Politecnico di Milano, Italy
{ghezzi,krstic}@elet.polimi.it

Abstract. This is the specification package document for the 1st International Competition of Software for Runtime Verification (CSRV-2014). We first present the SOLOIST specification language and describe the service composition we used to generate our benchmarks. We then describe the composition of the specification package (instrumentation and properties). Five benchmarks are provided as separate files, together with this document.

1 Preliminaries

1.1 SOLOIST at a glance

In this section we provide a summary of the specification language, SOLOIST (Specification Language for Service Compositions Interactions), used by our tool. For the rationale behind the language and a detailed explanation of its semantics see [3]. The syntax of SOLOIST is defined by the following grammar:
$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \mathsf{U}_I \phi \mid \phi \mathsf{S}_I \phi \mid \mathfrak{C}^K_{\bowtie n}(\phi) \mid \mathfrak{U}^{K,h}_{\bowtie n}(\phi) \mid \mathfrak{M}^{K,h}_{\bowtie n}(\phi) \mid \mathfrak{D}^K_{\bowtie n}(\phi,\phi)$
where $p \in \Pi$, with Π being a finite set of atoms; I is a nonempty interval over $\mathbb{N}$; n, K, h range over $\mathbb{N}$; $\bowtie \in \{<, \leq, \geq, >, =\}$. We restrict the arguments ϕ of modalities $\mathfrak{C}, \mathfrak{U}, \mathfrak{M}, \mathfrak{D}$ to atoms in Π.

The U_I and S_I modalities are, respectively, the metric "*Until*" and "*Since*" operators. Additional temporal modalities can be derived using the usual conventions; for example "*Eventually in the Future*" is defined as $\mathsf{F}_I\phi \equiv (\top \mathsf{U}_I \phi)$; "*Always in the future*" (or "*Globally*") is defined as $\mathsf{G}_I\phi \equiv \neg\mathsf{F}_I\neg\phi$; "*Eventually in the Past*" is defined as $\mathsf{P}_I\phi \equiv \top \mathsf{S}_I \phi$; "*Always in the Past*" (or "*Historically*") is defined as $\mathsf{H}_I\phi \equiv \neg\mathsf{P}_I\neg\phi$; "*Next*" is defined as $\mathsf{X}_I\phi \equiv \bot \mathsf{U}_I \phi$ and "*Yesterday*" is defined as $\mathsf{Y}_I\phi \equiv \bot \mathsf{S}_I \phi$, where $\top$ means "true" and $\bot$ means "false" . If interval I is omitted in the temporal formulae, interval $(0, \infty)$ is assumed. The remaining modalities are called *aggregate* modalities. The $\mathfrak{C}^K_{\bowtie n}(\phi)$ modality states a bound (represented by $\bowtie n$) on the number of occurrences of an event ϕ in the previous K time instants. The $\mathfrak{U}^{K,h}_{\bowtie n}(\phi)$ (respectively, $\mathfrak{M}^{K,h}_{\bowtie n}(\phi)$) modality expresses a bound on the average (respectively, maximum) number of occurrences of an event ϕ, aggregated over the set of right-aligned adjacent non-overlapping subintervals within a time window K, as in "the average/maximum number of events per hour in the last ten hours". A subtle difference in the semantics of the $\mathfrak{U}$ and $\mathfrak{M}$ modalities

$$
\begin{array}{ll}
(w,i) \models p & \text{iff } p \in \sigma_i \\
(w,i) \models \neg\phi & \text{iff } (w,i) \not\models \phi \\
(w,i) \models \phi \wedge \psi & \text{iff } (w,i) \models \phi \wedge (w,i) \models \psi \\
(w,i) \models \phi \mathsf{S}_I \psi & \text{iff for some } j < i, \tau_i - \tau_j \in I, (w,j) \models \psi \text{ and for all } k, j < k < i, (w,k) \models \phi \\
(w,i) \models \phi \mathsf{U}_I \psi & \text{iff for some } j > i, \tau_j - \tau_i \in I, (w,j) \models \psi \text{ and for all } k, i < k < j, (w,k) \models \phi \\
(w,i) \models \mathfrak{C}^K_{\bowtie n}(\phi) & \text{iff } c(\tau_i - K, \tau_i, \phi) \bowtie n \text{ and } \tau_i \geq K \\
(w,i) \models \mathfrak{U}^{K,h}_{\bowtie n}(\phi) & \text{iff } \frac{c(\tau_i - \lfloor \frac{K}{h} \rfloor h, \tau_i, \phi)}{\lfloor \frac{K}{h} \rfloor} \bowtie n \text{ and } \tau_i \geq K \\
(w,i) \models \mathfrak{M}^{K,h}_{\bowtie n}(\phi) & \text{iff } \max \left\{ \bigcup_{m=0}^{\lfloor \frac{K}{h} \rfloor} \{c(lb(m), rb(m), \phi)\} \right\} \bowtie n \text{ and } \tau_i \geq K \\
(w,i) \models \mathfrak{D}^K_{\bowtie n}(\phi, \psi) & \text{iff } \frac{\sum_{(s,t) \in d(\phi,\psi,\tau_i,K)} (\tau_t - \tau_s)}{|d(\phi,\psi,\tau_i,K)|} \bowtie n \text{ and } \tau_i \geq K
\end{array}
$$

where $c(\tau_a, \tau_b, \phi) = |\{s \mid \tau_a < \tau_s \leq \tau_b \text{ and } (w,s) \models \phi\}|$, $lb(m) = \max\{\tau_i - K, \tau_i - (m+1)h\}$, $rb(m) = \tau_i - mh$, and $d(\phi,\psi,\tau_i,K) = \{(s,t) \mid \tau_i - K < \tau_s \leq \tau_i \text{ and } (w,s) \models \phi, t = \min\{u \mid \tau_s < \tau_u \leq \tau_i, (w,u) \models \psi\}\}$

Fig. 1: Formal semantics of SOLOIST

is that $\mathfrak{M}$ considers events in the (possibly empty) tail interval, i.e., the leftmost observation subinterval whose length is less than h, while the $\mathfrak{U}$ modality ignores them. The $\mathfrak{D}^K_{\bowtie n}(\phi,\psi)$ modality expresses a bound on the average time elapsed between a pair of specific adjacent events ϕ and ψ occurring in the previous K time instants.

The formal semantics of SOLOIST is defined on timed ω-words [1] over $2^{\Pi} \times \mathbb{N}$. A timed sequence $\tau = \tau_0\tau_1\ldots$ is an infinite sequence of values $\tau_i \in \mathbb{N}$ with $\tau_i > 0$ satisfying $\tau_i < \tau_{i+1}$, for all $i \geq 0$, i.e., the sequence increases strictly monotonically. A timed ω-word over alphabet 2^{Π} is a pair (σ,τ) where $\sigma = \sigma_0\sigma_1\ldots$ is an infinite word over 2^{Π} and τ is a timed sequence. A timed language over 2^{Π} is a set of timed words over the same alphabet. Notice that there is a distinction between the integer position i in the timed ω-word and the corresponding timestamp τ_i. Figure 1 defines the satisfiability relation $(w,i) \models \phi$ for every timed ω-word w, every position $i \geq 0$ and for every SOLOIST formula ϕ.

LTL Mapping The mapping of SOLOIST into linear temporal logic (LTL) has been introduced in [3]. Here we provide only a brief overview.

First we need to bridge the gap between the semantics of SOLOIST based on timed ω-words, where the temporal information is denoted by a natural time-stamp, and the one used for LTL, where the temporal information is implicitly defined by the integer position in an ω-word. The two temporal models can be transformed into each other. We are interested in pinpointing in an LTL ω-word the positions that correspond to time-stamps in a SOLOIST timed ω-word where events occurred. To do so, we add to the set Π a special propositional symbol e, which is true in each position corresponding to a "valid" time-stamp in the timed ω-word.

We denote the translation of SOLOIST formulae to LTL with ρ. It is straightforward for the "standard" propositional operators ($\neg$ and $\wedge$). For temporal operators it is a two-step process. First we want to include proposition e in order to correctly model the timed ω-word. As shown in [5], MTL is simply LTL with an exponentially succinct encoding. Below we show how metric operators can be written in LTL, where $0 < l \leq$

$u < \infty, 1 < d < \infty, 0 \leq i < \infty, 0 \leq f \leq \infty$: [3]

$$\rho(\mathsf{X}_{=i}(\phi)) \equiv \underbrace{\mathsf{X}\mathsf{X}\cdots\mathsf{X}}_{i \text{ times}}(e \wedge \phi) \tag{$\mathscr{T}_X$}$$

$$\rho(\mathsf{G}_I(\phi)) \equiv \bigwedge_{i \in I} \rho(X_{=i}(\phi)) \tag{$\mathscr{T}_G$}$$

$$\rho(\phi \mathsf{U}_{\emptyset} \psi) \equiv \bot \tag{$\mathscr{T}_{U1}$}$$

$$\rho(\phi \mathsf{U}_{=i} \psi) \equiv \rho(\mathsf{G}_{(0,i)}(\phi)) \wedge \rho(\mathsf{X}_{=i}(\psi)) \tag{$\mathscr{T}_{U2}$}$$

$$\rho(\phi \mathsf{U}_{(0,d)} \psi) \equiv \mathsf{X}(\rho(\psi) \vee (\rho(\phi) \wedge \rho(\phi \mathsf{U}_{(0,d-1)} \psi))) \tag{$\mathscr{T}_{U3}$}$$

$$\rho(\phi \mathsf{U}_{(l,u)} \psi) \equiv \rho(\mathsf{G}_{(0,l]}(\phi)) \wedge \rho(\mathsf{X}_{=l}(\phi \mathsf{U}_{(0,u-l)} \psi)) \tag{$\mathscr{T}_{U4}$}$$

$$\rho(\phi \mathsf{U}_{(i,\infty)} \psi) \equiv \rho(\mathsf{G}_{(0,i]}(\phi)) \wedge \rho(\mathsf{X}_{=i}(\phi \mathsf{U} \psi)) \tag{$\mathscr{T}_{U5}$}$$

The translation of the $\mathfrak{C}$ modality considers a formula of the form $\mathfrak{C}^K_{>n}(\phi)$ as the base case. This formula is translated as a disjunction of formulae denoting all possible cases where ϕ holds $n+1$ times within the time window K.

Example. Consider the formula $\mathfrak{C}^3_{>1}(\phi)$. Within a time window of length 3, there are three possible combinations, in terms of positions on the timeline, for representing 2 (i.e., the bound n in the formula incremented by 1) occurrences of the event ϕ: $\mathsf{Y}^2(\phi) \wedge \mathsf{Y}^1(\phi)$, $\mathsf{Y}^2(\phi) \wedge \phi$, and $\mathsf{Y}^1(\phi) \wedge \phi$; [4] these formulae are combined in a disjunction in the final translation.

More general, the translation of $\mathfrak{C}^K_{>n}(\phi)$ is the following:

$$\rho(\mathfrak{C}^K_{>n}(\phi)) = \bigvee_{0 \leq i_1 < \ldots < i_{n+1} < K} \left(\mathsf{Y}^{i_1}(e \wedge \phi) \wedge \ldots \wedge \mathsf{Y}^{i_{n+1}}(e \wedge \phi)\right)$$

The translation for other values of the $\bowtie$ operator is defined by reducing the formula to an equivalent instance of the base case: $\mathfrak{C}^K_{\leq n} \equiv \neg\mathfrak{C}^K_{>n}$; $\mathfrak{C}^K_{\geq n} \equiv \mathfrak{C}^K_{>n-1}$; $\mathfrak{C}^K_{<n} \equiv \neg\mathfrak{C}^K_{>n-1}$; $\mathfrak{C}^K_{=n} \equiv \mathfrak{C}^K_{>n-1} \wedge \neg\mathfrak{C}^K_{>n}$.

The translation of the $\mathfrak{U}$ and $\mathfrak{M}$ modalities is defined in terms of the $\mathfrak{C}$ modality. A formula like $\mathfrak{U}^{K,h}_{\bowtie n}(\phi)$ is equivalent to $\mathfrak{C}^{\lfloor \frac{K}{h} \rfloor \cdot h}_{\bowtie n \cdot \lfloor \frac{K}{h} \rfloor}(\phi)$. For the $\mathfrak{M}$ modality, we have two base cases: a formula like $\mathfrak{M}^{K,h}_{<n}(\phi)$ is equivalent to $\left(\bigwedge_{m=0}^{\lfloor \frac{K}{h} \rfloor - 1} \mathsf{Y}^{m \cdot h}(\rho(\mathfrak{C}^h_{<n}\phi))\right) \wedge \left(\mathsf{Y}^{\lfloor \frac{K}{h} \rfloor \cdot h}(\rho(\mathfrak{C}^{(K \bmod h)}_{<n}\phi))\right)$ and $\mathfrak{M}^{K,h}_{>n}(\phi)$ is equivalent to $\left(\bigvee_{m=0}^{\lfloor \frac{K}{h} \rfloor - 1} \mathsf{Y}^{m \cdot h}(\rho(\mathfrak{C}^h_{>n}\phi))\right) \vee \left(\mathsf{Y}^{\lfloor \frac{K}{h} \rfloor \cdot h}(\rho(\mathfrak{C}^{(K \bmod h)}_{>n}\phi))\right)$. The translation for other values of the $\bowtie$ operator is defined similarly as the one for $\mathfrak{C}$ modality.

The translation of a $\mathfrak{D}^K_{\bowtie n}(\phi, \psi)$ formula is defined as a disjunction of formulae denoting all possible occurrences of instances of the pair of events (ϕ, ψ), satisfying the bound related to the average distance. Within a time window K, the maximum number of possible instances of events pairs is $\lfloor \frac{K}{2} \rfloor$. For each of these possible numbers of

[3] Notice that interval "=i" can be written as $[i,i]$, or $(i-1, i+1)$ since intervals are over $\mathbb{N}$. Also, we omit the encoding of the *since* modality, which is derivable similarly.

[4] We use the LTL shorthand notation: $\mathsf{Y}^K(\phi) \equiv \underbrace{\mathsf{Y}\mathsf{Y}\cdots\mathsf{Y}(\phi)}_{K \text{ times}}$.

instances, we define a disjunction of formulae characterizing all their possible combinations in terms of position on the timeline, such that bound on the average distance is satisfied. We also need to explicitly state that events pairs do not occur in all other time instants. *Example.* Consider, the formula $\mathfrak{D}^4_{\leq 1}\{(\phi,\psi)\}$. One possible combination of occurrences of a pair of events (ϕ,ψ) is characterized by the formula $\mathsf{Y}^3(\phi)\wedge\mathsf{Y}^2(\psi)$; the absence of other occurrences in the other time instants is denoted by the formula $\neg((\mathsf{Y}^2(\phi)\wedge\mathsf{Y}^1(\psi))\vee(\mathsf{Y}^2(\phi)\wedge\psi)\vee(\mathsf{Y}^1(\phi)\wedge\psi))$. A similar template can be followed to characterize other combinations of events on the timeline. For the $\mathfrak{D}^K_{\bowtie n}$ modality, $\rho(\mathfrak{D}^K_{\bowtie n}(\phi,\psi))$ is defined as follows:

$$\bigvee_{0<h\leq\lfloor\frac{K}{2}\rfloor}\left(\bigvee_{\substack{0\leq i_1<j_1<\ldots i_h<j_h<K\\ \text{and}\\ \left(\sum_{m=1}^{h}\frac{j_m-i_m}{h}\right)\bowtie n}}\left(\begin{array}{c}\mathsf{Y}^{i_1}(e\wedge\phi)\wedge\mathsf{Y}^{j_1}(e\wedge\psi)\wedge\\ \ldots\\ \wedge\mathsf{Y}^{i_h}(e\wedge\phi)\wedge\mathsf{Y}^{j_h}(e\wedge\psi)\wedge\\ \neg\left(\bigvee_{\substack{0\leq s<t<K\\ s\notin\{i_1,\ldots,i_h\}\\ t\notin\{j_1,\ldots,j_h\}}}\left(\mathsf{Y}^s(e\wedge\phi)\wedge\mathsf{Y}^t(e\wedge\psi)\right)\right)\end{array}\right)\right)$$

1.2 Service composition example

In this section we describe the service composition example used to generate our benchmarks. Consider the service composition depicted in Fig. 2 using the (visually intuitive) notation for BPEL introduced in [2].

The process *ATMFrontEnd* starts when the *receive* activity `connect` processes a message from the *Teller machine*. This starts a customer session: the process requests a ticket from the *Ticket issuer* service and sends a confirmation to the teller machine that the connection is established (*connect* reply activity). A loop is started to service the different customers using the teller machine. If a user attempts at logging in by inserting a card, the *pick* activity will receive a *logOn* message. If a user cancels the session and retrieves the card, a *disconnect* message is received. After the user input the login data, the process verifies whether the customer holds a valid account at the bank, by invoking the `checkAccess` operation of the *Account system* service. If the latter identifies the customer, a loop (denoted as *main* loop in Fig. 2) is started to manage the customer's requests. The *pick* activity (shown in Fig. 3), contained in the body of the loop may receive four kinds of possible requests: three of them (`getBalance`, `deposit`, `withdraw`) are forwarded to the corresponding operations of the *Account system* service; the `logOff` request terminates the loop, closing the current customer session. the `getBalance` request retrieves the current balance of the account; the `deposit` and `withdraw` requests are used for depositing/withdrawing money.

2 Instrumentation

We create the benchmarks accompanying this submission using the Process Log Generator (PLG) tool [4] on a model of the service composition described in Sect. 1.2. This model was defined by specifying the workflow structure, the duration of each synchronous *invoke* activity, the branching probabilities, and the error rates. Other activities

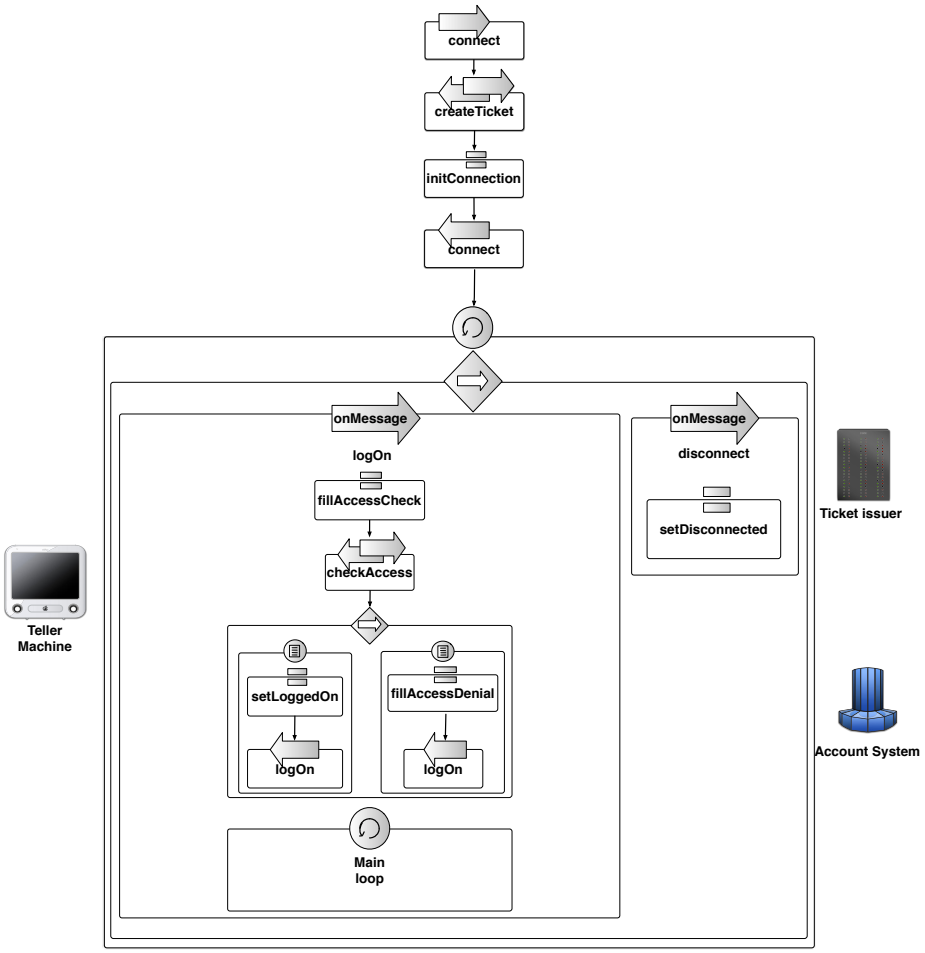

Fig. 2: *ATMFrontEnd* business process

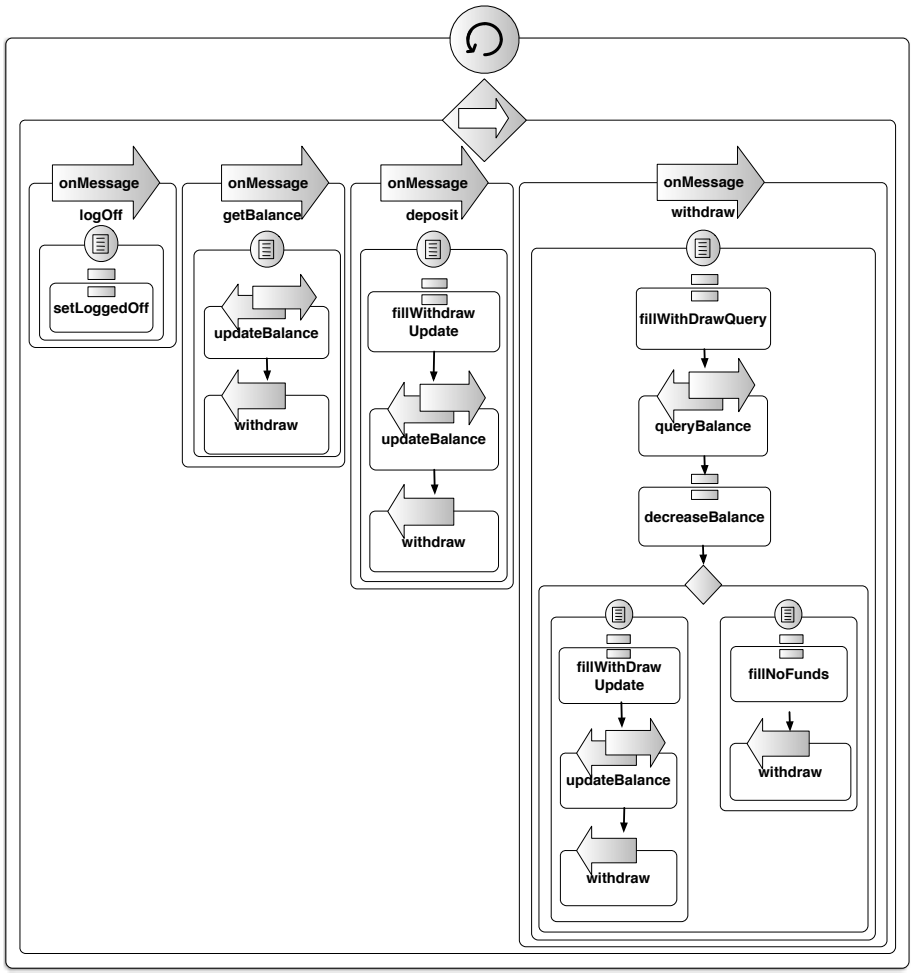

Fig. 3: Main loop of the *ATMFrontEnd* business process

(e.g., *receive*) were given 0 as duration; branching was used to create loops and simulate the behavior of the *pick* activity. The PLG tool is able to synthesize logs of process invocations that conforms to the described service composition.

Each event in a trace corresponds to one of the activities in Fig. 2 and 3. All events follow a naming convention: *invoke* activities are prefixed by "inv", *receive* activities by "recv", *assign* activities by "assn" and *reply* activities by "rep". Each event has two fields: its *timestamp* (the time elapsed since the beginning of the process execution) and *activity type*, denoting the type of the activity (*invoke*, *receive*, *assign* or *reply*). *invoke* activities have an additional field *event type* that distinguishes between the start (value: *start*) and the end (value: *complete*) of the invocation. For example:

```
<event>
<name>invcheckaccess</name>
        <field>
                <name>Timestamp</name>
                <value>935</value>
        </field>
        <field>
                <name>EventType</name>
                <value>complete</value>
        </field>
        <field>
                <name>ActivityType</name>
                <value>Invocation</value>
        </field>
</event>
```

corresponds to the completion of the `checkAccess` *invoke* activity, occurred 935 time units since the beginning of the process.

3 Properties

To annotate a BPEL process with SOLOIST, we denote the execution of each activity with a predicate symbol. Synchronous *invoke* activities are actually modeled with two predicates, corresponding to the start and the end of the invocation; these are denoted with the "_start" and "_complete" suffixes, respectively. Time units are expressed in seconds.

3.1 Formal and informal description

QP1: WithdrawalLimit
The number of withdrawal operations performed within 10 minutes before customer logs off is less than or equal to the allowed limit (assumed to be 3). This property is expressed as:
$\mathsf{G}(\mathit{recvlogoff} \rightarrow \mathfrak{C}^{600}_{\leq 3}(\mathit{repwithdraw}))$.

QP2: CheckAccessAverageResponseTime
The average response time of operation `checkAccess` provided by the *Account system* service is always less than 5 seconds within any 15 minute time window. This property is expressed as:
$\mathsf{X}^6(\mathsf{G}(\mathfrak{D}^{900}_{\leq 5}\{(\mathit{invcheckaccess\_start}, \mathit{invcheckaccess\_complete})\}))$.

QP3: CheckAuthControl
Every `logon` reply activity must be preceded by `invcheckAccess` operation and no `logon` activity may occur in between. This property is expressed as:
$\mathsf{G}(\mathit{replogon} \rightarrow (\neg \mathit{replogon})\mathsf{S}\ \mathit{invcheckaccess\_complete})$

QP4: AbsoluteResponseTime
Each invocation of `checkAccess` activity will receive a matching response within 10 time units. This property is expressed as:
$\mathsf{G}(\mathit{invcheckaccess\_start} \rightarrow (\neg \mathit{invcheckaccess\_start})\mathsf{U}_{(0,10)}\mathit{invcheckaccess\_complete})$

3.2 Verdicts

Table 1 contains the verdicts (shown in columns) for all the properties QP1–QP4 described above, with respect to each benchmark B1–B5 (shown in rows).

Table 1: Specification verdicts with respect to the benchmarks

	QP1	QP2	QP3	QP4
B1	holds	holds	holds	holds
B2	holds	violated	holds	violated
B3	holds	violated	holds	violated
B4	violated	holds	holds	holds
B5	violated	holds	holds	holds

Acknowledgments. This work has been partially supported by the National Research Fund, Luxembourg (FNR/P10/03).

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoreotical Computer Science 126(2), 183–235 (Apr 1994)
2. Baresi, L., Bianculli, D., Ghezzi, C., Guinea, S., Spoletini, P.: Validation of web service compositions. IET Softw. 1(6), 219–232 (2007)
3. Bianculli, D., Ghezzi, C., San Pietro, P.: The tale of SOLOIST: a specification language for service compositions interactions. In: Proc. of FACS'12. LNCS, vol. 7684, pp. 55–72. Springer (2013)
4. Burattin, A., Sperduti, A.: Plg: A framework for the generation of business process models and their execution logs. In: Business Process Management Workshops. LNBIP, vol. 66, pp. 214–219. Springer (2011)
5. Furia, C.A., Spoletini, P.: Tomorrow and all our yesterdays: Mtl satisfiability over the integers. In: Theoretical Aspects of Computing - ICTAC. pp. 126–140. Springer (2008)