# Analyzing Web Service Resource Compatibility*

Irina A. Lomazova[1,2] and Ivan V. Romanov[1]

[1] National Research University Higher School of Economics, Moscow, Russia
romanov.ekb@gmail.com
[2] Program Systems Institute of the Russian Academy of Sciences,
Pereslavl-Zalessky, Russia
i_lomazova@mail.ru

**Abstract.** In this work we consider modeling of services with workflow modules, which are a subclass of Petri nets. The service compatibility problem is to answer the question, whether two Web services fit together, i.e. whether the composed system is sound. We study complementarity of service produced/consumed resources, that is a necessary condition for the service compatibility. Resources, which are produced/consumed by a Web service, are described as a multiset language. We define an algebra of multiset languages and present an algorithm for checking the conformance of resources for two given structured workflow modules.

## 1 Introduction

Service-Oriented Computing is an emerging computing paradigm that supports the modular design of (software) systems. Complex systems are designed by composing less complex systems, called services.

A service consists of a control structure describing its behavior and of an interface to communicate asynchronously with other services. An interface is a set of (input and output) channels. In order that two services can interact with each other, an input channel of the one service has to be connected with an output channel of the other service.

The problem of checking services compatibility draws attention of many researchers [11, 12]. A lot of different approaches have developed to verify the compatibility of component Web services. Many of them utilize Petri Nets for this purpose [9, 13]. Other models such as finite state machine are also used [5]. Some of the researches deal with concrete Web service specifications, for example business process execution language for Web services (BPEL) [13].

Checking semantical correctness, e.g. deadlock freedom, for composition of two services is a hard problem. Even when a property is decidable, its complexity makes it almost intractable. So, finding relatively easy to check necessary

* This study was carried out within the National Research University Higher School of Economics' Academic Fund Program in 2012-2013, research grant No. 11-01-0032 and is supported by Russian Fund for Basic Research (projects 11-01-00737, 11-07-00549)

conditions for correctness of services composition may help to find some bugs for low costs and avoid further verification.

In this paper we study services resource conformance, notably whether two services have complementary runs, such that all outputs of one service are consumed by and enough for another service and vice versa.

Services are modeled with workflow modules, also called open workflow nets — WF-nets (see e.g. [1]) with additional input/output places representing input/output channels (cf. [3, 6]). The core WF-net in a workflow module describes service control flow, and resource places represent its interface.

Since we study services compatibility we suppose control workflow nets to be sound WF-nets. The soundness property guarantees proper termination of autonomous workflow processes (not taking modules interactions into account, so that resources can be generated or consumed during a process execution without any restrictions). Moreover, we restrict ourselves to structured WF-moduls — an important subclass of workflow modules with control WF-nets sound by construction.

The paper is organized as follows. Section 2 contains some basic definitions and notions, including formal definitions of workflow nets, workflow modules and composition of workflow modules. In Section 3 a motivating example of two workflow modules, modeling a part of credit allowence system, is given. In Section 4 we define and study a language of quasi regular expressions for describing a workflow module resource interfaces. In Section 5 we present an algorithm for checking resource compatibility of two structured workflow modules. Section 6 contains some conclusions.

## 2 Petri nets and workflow modules. Definitions

Let $S$ be a finite set. A *multiset* $m$ over a set $S$ is a mapping $m : S \to Nat$, where $Nat$ is the set of natural numbers (including zero), i. e. a multiset may contain several copies of the same element.

For two multisets $m, m'$ we write $m \subseteq m'$ iff $\forall s \in S : m(s) \leq m'(s)$ (the inclusion relation). The sum and the union of two multisets $m$ and $m'$ are defined as usual: $\forall s \in S : m + m'(s) = m(s) + m'(s),\ m \cup m'(s) = max(m(s), m'(s))$. By $\mathcal{M}(S)$ we denote the set of all finite multisets over $S$.

Non-negative integer vectors are often used to encode multisets. Actually, the set of all multisets over finite $S$ is a homomorphic image of $Nat^{|S|}$.

Let $P$ and $T$ be disjoint sets of *places* and *transitions* and let $F : (P \times T) \cup (T \times P) \to Nat$. Then $N = (P, T, F)$ is a *Petri net*. A *marking* in a Petri net is a function $M : P \to Nat$, mapping each place to some natural number (possibly zero). Thus a marking may be considered as a multiset over the set of places. Pictorially, $P$-elements are represented by circles, $T$-elements by boxes, and the flow relation $F$ by directed arcs. Places may carry tokens represented by filled circles. A current marking $M$ is designated by putting $M(p)$ tokens into each place $p \in P$. Tokens residing in a place are often interpreted as resources of some type consumed or produced by a transition firing.

For a transition $t \in T$ an arc $(x, t)$ is called an *input arc*, and an arc $(t, x)$ — an *output arc*; the *preset* ${}^\bullet t$ and the *postset* $t^\bullet$ are defined as the multisets over $P$ such that ${}^\bullet t(p) = F(p, t)$ and $t^\bullet(p) = F(t, p)$ for each $p \in P$.

A transition $t \in T$ is *enabled* in a marking $M$ iff $\forall p \in P\ M(p) \geq F(p, t)$. An enabled transition $t$ may *fire* yielding a new marking $M' =_{\text{def}} M - {}^\bullet t + t^\bullet$, i. e. $M'(p) = M(p) - F(p, t) + F(t, p)$ for each $p \in P$ (denoted $M \xrightarrow{t} M'$, or just $M \to M'$). We say that $M'$ is reachable from $M$ iff there is a sequence of firings $M = M_1 \to M_2 \to \cdots \to M_n = M'$. For a Petri net $N$ by $\mathcal{R}(N, m)$ we denote the set of all markings reachable in $M$ from the marking $m$, by $\mathcal{R}(N, m)$ — the set of all markings reachable in $M$ from its initial marking.

Workflow nets (WF-nets) is a special subclass of Petri nets designed for modeling workflow processes [1].

A workflow net has one initial and one final place, and every place or transition in it is on a directed path from the initial to the final place.

**Definition 1.** *A Petri net $N$ is called a* workflow net (WF-net) *iff*

1. *There is one source place $i \in P$ and one sink place $f \in P$ s. t. ${}^\bullet i = f^\bullet = \emptyset$;*
2. *Every node from $P \cup T$ is on a path from $i$ to $f$.*
3. *The initial marking in $N$ contains the only token in its source place.*

By abuse of notation we denote by $i$ both the source place and the initial marking in a WF-net. Similarly, we use $f$ to denote the final marking in a WF-net $N$, defined as a marking containing the only token in the sink place $f$.

An important correctness property for WF-nets is *soundness*. Classical WF-nets are called sound if one can reach the final marking from any marking reachable from the initial marking [1]. The intuition behind this notion is that no matter what happens, there is always a way to complete the execution and reach the final state. This soundness property is sometimes also called *proper termination* and corresponds to classical soundness in [2].

**Definition 2.** *A WF net $N$ with a source place $i$ and a sink place $f$ is called* sound *iff*

1. *For every marking $m$ reachable from the initial marking $i$, there exists a firing sequence leading from $m$ to the final marking $f$:*
   $\forall m \in \mathcal{R}(N) : (i \xrightarrow{*} m) \Rightarrow (m \xrightarrow{*} f)$;
2. *The marking $f$ is the only marking reachable from $i$ with at least one token in place $f$:*
   $\forall m \in \mathcal{R}(N) : m \geq f \Rightarrow m = f$;
3. *There are no dead transitions in $N$:*
   $\forall t \in T \exists m, m' : i \xrightarrow{*} m \xrightarrow{t} m'$.

For an arbitrary WF net soundness is decidable, but it is EXPSPACE-hard [4].

Modeling workflow consists of modeling case management with the help of sequential routing, parallelism, iteration, and conditional routing. To express

it explicitly building blocks such as the AND-split, AND-join, OR-split and OR-join can be used. The AND-split and the AND-join are used for parallel routing. The OR-split and the OR-join are used for conditional routing. All these constructs can be easily expressed in Petri net formalism.

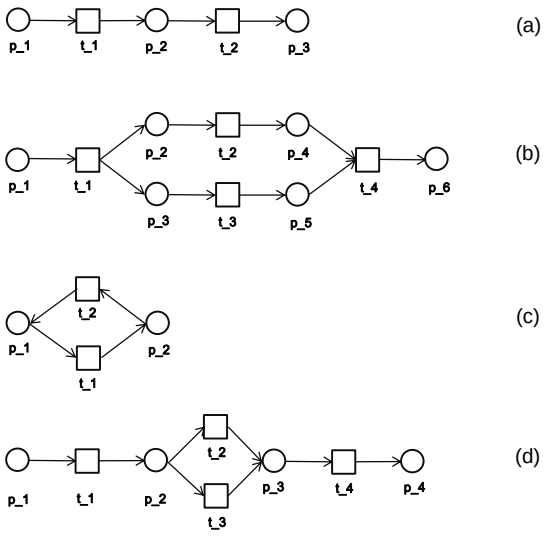


**Fig. 1.** Routing operations: (a) sequential routing, (b) parallel routing, (c) iteration, (d) conditional routing

To guarantee, that we get 'good' workflows, we are to balance AND/OR-splits and AND/OR-joins. Clearly, two parallel flows initiated by an AND-split, should not be joined by an OR-join. Two alternative flows created via an OR-split, should not be synchronized by an AND-join. When we follow these rules we obtain structured WF-nets (see [1] for more details). Fig. 1 shows fore routing operations used in structured WF-nets. Thus, to apply sequential operation to two WF-nets $N_1$ and $N_2$ is to substitute $N_1$ for $t_1$, and $N_2$ for $t_2$ in the net, shown in Fig. 1 (a), by substituting the source place of $N_1$ for $p_1$, merge the sink place of $N_1$ and the source place of $N_2$ (for $p_2$), and substitute the sink place of $N_2$ for $p_3$. To apply parallel operation to WF-nets $N_1$ and $N_2$ is to substitute $N_1$ for $t_2$, and $N_2$ for $t_3$ in the net, shown in Fig. 1 (b), while transitions $t_1, t_4$ are additional routing transitions here. The iteration and conditional operations are defined in the similar way.

**Definition 3.** *An* atomic WF-net *is a WF-net, consisting of one source place $i$, one sink place $f$ and one transition, for which $i$ is the only input place, and $f$ is the only output place.*

*A WF-net is called a* structured WF-net *iff it can be obtained from atomic WF-nets by successive application of routing operations, shown in Fig. 1.*

It was shown in [1], that structured WF-nets are sound by construction.

Following P. Martens [11], S. Stahl [12], and others to model services we use workflow modules a — special subclass of Petri nets.

**Definition 4.** *A Petri net $M = (P, T, F)$ is called a workflow module (WF-module) iff*

1. *The set $P$ of places is a disjoint union of three sets: internal places $P^N$, input places $P^I$, and output places $P^O$.*
2. *The flow relation is divided into internal flow $F^N \subseteq (P^N \times T) \cup (T \times P^N)$ and communication flow $F^C \subseteq (P^I \times T) \cup (T \times P^O)$.*
3. *The net $PM = (P^N, T, F^N)$ is a WF-net.*
4. *No transition is connected both to an input place and an output place.*

Within a WF-module $M$, the workflow net $PM$ is called the *internal process* of $M$ and the tuple $\mathcal{I}(M) = (P^I, P^O)$ is called its *interface*. Places belonging to the interface $\mathcal{I}(M)$ are called *ports*. We suppose that each port in $\mathcal{I}(M)$ has a unique name. A workflow module is called *structured* iff its internal process is a structured WF-net.

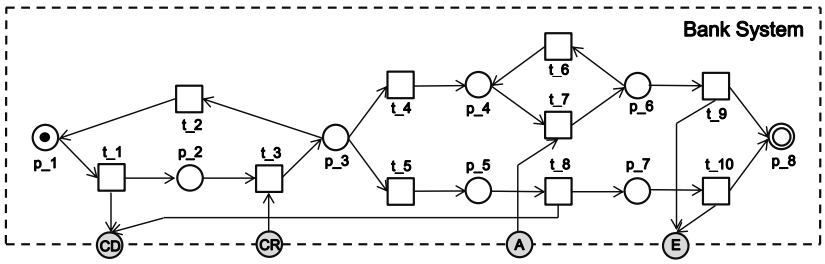


**Fig. 2.** WF-module $WFM_1$ for a bank service

Fig. 2 shows an example of a WF-module $WFM_1$. Here $p_1$ is a source place, $p_8$ — a sink place. Places $p_1, \ldots, p_8$ are internal places. The interface of $M_1$ consists of input places $CR$ and $A$, and output places $CD$ and $E$. The communication flow of $WFM_1$ includes arcs $(t_1, CD)$, $(CR, t_3)$, $(A, t_7)$, and $(t_9, E)$. All other arcs belong to the internal process of $M_1$. Note, that $WFM_1$ is a structured WF-module.

**Definition 5.** *A WF-module is called* sound *iff its internal process is sound.*

A composition of WF-modules, modeling Web services interaction, is defined as follows.

**Definition 6.** *Let $M_1, M_2$ be two WF-modules. A* composition $M_1 \odot M_2$ *of $M_1$ and $M_2$ is a net $N$ obtained from $M_1$ and $M_2$ by merging input ports of $M_1$ with output ports of $M_2$ with the same names, and similarly output ports of $M_1$ with input ports of $M_2$, i.e. $p_1 \in \mathcal{I}(M_1)$ and $p_2 \in \mathcal{I}(M_2)$ are merged iff they have the same name and one of them is an input port, while the other is an output port.*

An example of two WF-modules composition is shown in the next section.

A composition $M_1 \odot M_2$ of two WF-modules $M_1$ and $M_2$ can be easily transformed into a WF-module as follows. Let $i_1, i_2$ be source places in $M_1$ and $M_2$ correspondingly, and $f_1, f_2$ be their output places. Add to $M_1 \odot M_2$ a new source place $i$, a new sink place $f$, and two new transitions $t_i, t_f$, s.t. ${}^\bullet t_i = \{i\}$, $t_i^\bullet = \{i_1, i_2\}$, and $t_f^\bullet = \{f\}$, ${}^\bullet t_i = \{f_1, f_2\}$.

We say that a composition $M_1 \odot M_2$ is *sound* iff its corresponding WF-module is sound.

## 3 Motivating example

As a motivating example we consider a model of credit allowance. The model describes two services, represented by two WF-modules.

The WF-module $WFM_1$ in Fig. 2 models a service of a bank credit system, that allows to estimate whether it is reasonable to loan money to this or that person. In this model work startes from the position "p_1", where one token is placed in the initial state. Then service sends client information to the credit bureau ("t_1") and is waiting for the response ("p_2"). When bank system takes the credit rating and is analyzed it ("t_3"), decision makes in position "p_3", service can goes to the "p_1" through "t_2" for the repetition of request , or can goes to the operation of the payment. There are two ways for that ("t_5" and "t_6"). First, pay all accounts, which were obtained from the credit bureau ("t_7"). Second, send information about the client, when he will pay the loan. At the end of the each possible way bank system sends E to report about the competition of conversation("t_9", "t_10") and stops in position "p_8".

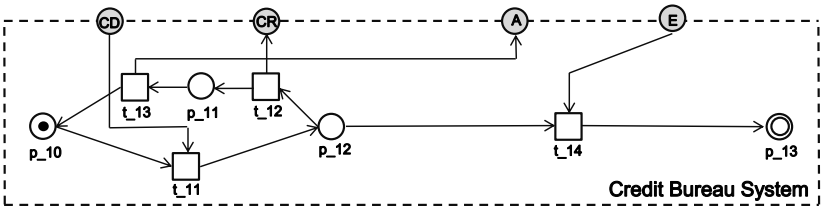


**Fig. 3.** WF-module $WFM_2$ for a credit bureau

Fig. 3 shows a WF-module, modeling a credit bureau. It is a company that collects information from various sources and provides consumer credit information on individual consumers for a variety of uses. Here there are four internal

places and four interface places. “p_10” is starting position, where service is waiting for the client information from the external bank system. When it receives necessary data from “CD”, transition “t_11” fires, client credit rating is calculated, and service goes to the position “p_12”. If position “CR” is enabled, that means that bank is requesting for the credit rating. In that case credit bureau sends this information (“CR”) to the partner service(“t_12”), after that sends an account (“A”) to the bank (“t_13”) and goes to the place “p_10” again. Otherwise, if credit bureau receives end request (“E”), it goes to the final position “p_13” and stops.

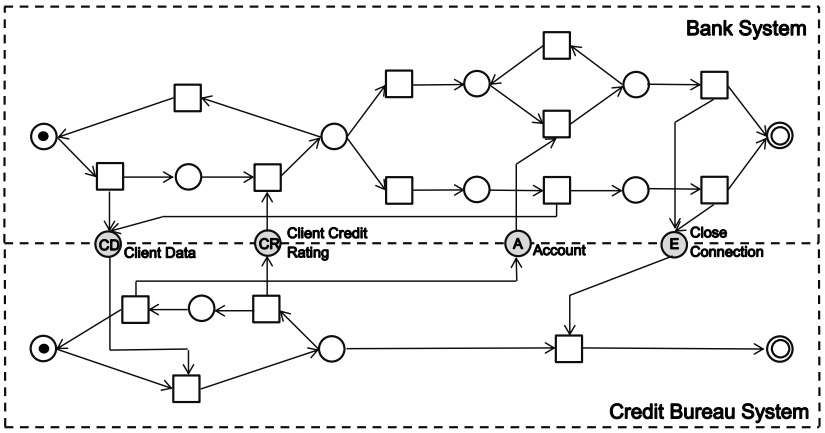


**Fig. 4.** Composition of services

Interaction of these two services is modeled by the composition of the corresponding WF-modules, depicted in Fig. 4. Services communicate with each other in the following way: the bank sends all information about the person to the credit bureau, this information is analyzed, and basing on all data credit bureau provides a credit rating of a definite client to the bank. This sequence of actions may be repeated several times.

While each of WF-modules $WFM_1$ and $WFM_2$ is sound, their composition $WFM_1 \odot M\, WFM_2$ is not sound. These services are resource incompatible, i.e. for these services there is no executions without pending inputs and/or outputs. In the next sections we show, how to find out such incompatibilities.

## 4 Workflow module resources and multiset languages

Let $M$ be a structured WF-module with two disjoint sets $P^I$ of input places and $P^O$ output places. Consider then some run $\delta$ for $M$ — a sequence of states and transition firings, starting from the initial state and coming to the final state of $M$. For each run $\delta$ the pair of *input and output resources* $R(\delta) = (R^I(\delta), R^O(\delta))$,

where $R^I(\delta) \in \mathcal{M}(P^I)$ and $R^O(\delta) \in \mathcal{M}(P^O)$, are defined as multisets of inputs/outputs consumed/produced in the course of $\delta$ execution. By $\rho(M)$ we denote the set of all pairs of resources for all runs in $M$. More formally:

**Definition 7.** *Let $M$ be a structured WF-module with input ports $P^I$ and output ports $P^O$. Then for $R^I \in \mathcal{M}(P^I)$ and $R^O \in \mathcal{M}(P^O)$ we define $(R^I, R^O) \in \rho(M)$ iff $f + R^O \in \mathcal{R}(M, i + R^I)$.*

It is easy to note, that a pair of multisets over non-intersecting sets of places can be considered as just one multiset. Thus for a workflow module a resource is a multiset language [7, 8].

For describing resources of structured workflow modules we define a special *language of quasi-regular expressions.*

**Definition 8.** *Let Atom be a finite set of atoms (letters). The* language of quasi-regular expressions *over Atom is defined by induction as follows:*

1. $\epsilon \in \mathcal{L}$;
2. $a \in \mathcal{L}$, *if* $a \in Atom$;
3. $e_1 \circ e_2 \in \mathcal{L}$, *if* $e_1, e_2 \in \mathcal{L}$;
4. $e_1 \oplus e_2 \in \mathcal{L}$, *if* $e_1, e_2 \in \mathcal{L}$;
5. $e^* \in \mathcal{L}$, *if* $e \in \mathcal{L}$.

*Semantics of $\mathcal{L}$ maps a quasi-regular expression $e$ to a multiset language $\mu(e)$, i.e. a set of multisets over Atom according to the following rules:*

1. $\mu(\epsilon) = \emptyset$;
2. $\mu(a) = [a]$ *for* $a \in Atom$;
3. $\mu(e_1 \circ e_2) = \{m_1 + m_2 | m_1 \in \mu(e_1), m_2 \in \mu(e_2)\}$;
4. $\mu(e_1 \oplus e_2) = \mu(e_1) \cup \mu(e_2)$;
5. $\mu(e^*) = \mu(e^0) \cup \mu(e^1) \cup \mu(e^2) \cdots \cup \mu(e^n) \ldots$, *where* $e^0 = \epsilon$, $e^n = e \circ e^{n-1}$ *for* $n \geq 1$.

Now we say, that two quasi-regular expressions $e_1$ and $e_2$ are *equivalent* (denoted $e_1 = e_2$ by abuse of notation) iff $\mu(e_1) = \mu(e_2)$.

**Proposition 1.** *For all quasi-regular expressions $e, e_1, e_2, e_3$ the following equations are valid:*

1. $e \circ \epsilon = e$,
2. $e \oplus \epsilon = e$,
3. $\epsilon^* = \epsilon$,
4. $e_1 \circ e_2 = e_2 \circ e_1$,
5. $e_1 \oplus e_2 = e_2 \oplus e_1$,
6. $(e_1 \circ e_2) \circ e_3 = e_1 \circ (e_2 \circ e_3)$,
7. $(e_1 \oplus e_2) \oplus e_3 = e_1 \oplus (e_2 \oplus e_3)$,
8. $e \circ (e_1 \oplus e_2) = (e \circ e_1) \oplus (e \circ e_2)$,
9. $e \oplus e = e$,

*10.* $(e^*)^* = e^*$,
*11.* $(e_1 \oplus e_2)^* = e_1^* \circ e_2^*$,
*12.* $(e_1 \circ e_2^*)^* = e_1^* \circ e_2^*$.

These equations define the *algebra of quasi-regular expressions.*

**Definition 9.** *We say that a quasi-regular expression $e$ is in the standard form, iff $e = e_1 \oplus \cdots \oplus e_n$, where $n \geq 1$, $e_1, \ldots, e_n$ do not contain $\oplus$ and nested $^*$.*

**Proposition 2.** *Every quasi-regular expression can be transformed to an equivalent quasi-regular expression in the standard form by applying equations (1–12).*

*Proof (Sketch).* To take $\oplus$-operation outside we use equations 8 and 11. Equations 10, 12 allow taking $\circ$-operation outside in subexpressions without $\oplus$. Nested $^*$ are eliminated with the help of equation 10.

Now we show, that for a structured WF-module $M$ a quasi-regular expression $e(M)$, describing interface resources $\rho(M)$, can be effectively constructed.

Recall, that structured workflow nets are constructed from atomic transitions by sequential application of four control structures: sequential routing, conditional routing, parallel routing and iteration.

**Algorithm 1.** *(Constructing a quasi-regular expression representing interface resources of a WF-modul).*
For a structured workflow module $M$ a quasi-regular expression $e(M)$ can be constructed by induction on the structure of internal process $N$ of $M$:

- for an atomic net $N$ — a transition with input resource places $p_1, \ldots, p_k$ and output resource places $q_1, \ldots, q_n$, define $e(N) = ?p_1 \circ \cdots \circ ?p_k \circ !q_1 \circ \cdots \circ !q_n$;
- for $N$ being sequential composition of $N_1$ and $N_2$ define $e(N) = e(N_1) \circ e(N_2)$;
- for $N$ being parallel composition of $N_1$ and $N_2$ define $e(N) = e(N_1) \circ e(N_2)$;
- for $N$ being selective composition of $N_1$ and $N_2$ define $e(N) = e(N_1) \oplus e(N_2)$;
- for $N$ being an iteration of $N_1$ and $N_2$ define $e(N) = e(N_1) \circ (e(N_2) \circ e(N_1))^*$.

**Proposition 3.** *Let $M$ be a structured WF-module, and let $e(M)$ be a quasi-regular expression, obtained for $M$ according to the Algorithm 1. Then $e(M) = \rho(M)$.*

The proof of this proposition is straightforward by induction on the structure of the internal process of a given WF-module.

## 5 Resources compatibility

We say, that two workflow modules are resource compatible, if they may execute runs with producing/consuming mutually complimentary resources. If the composition of two workflow modules is sound, then they are resource compatible.

However, it could be that for resource compatible modules their composition is not sound, since resources may be outputted in a wrong order. So, resource compatibility is a necessary, but not sufficient condition for the correctness of services composition.

**Definition 10.** *Let $M_1, M_2$ be two WF-modules with source places $i_1, i_2$ and sink places $f_1, f_2$ correspondingly. We say that $M_1$ and $M_2$ are resource compatible iff $f_1 + f_2 \in \mathcal{R}(i_1 + i_2)$ in $M_1 \odot M_2$.*

Immediately from the definition we get that if $M_1 \odot M_2$ is sound, then $M_1$ and $M_2$ are resource compatible.

We show that resource compatibility is decidable for structured WF-modules.

**Algorithm 2.** *(checking resource compatibility of two structured WF-modules).* Let $M_1, M_2$ be structured WF-modules.

**Step 1.** Construct quasi-regular expressions $e(M_1), e(M_2)$.
**Step 2.** Reduce $e(M_1), e(M_2)$ to standard forms $e_s(M_1), e_s(M_2)$ correspondingly.
**Step 3.** Construct a complementary expression $e_s^{-1}(M_1)$ by changing ? for ! and vice versa in $e_s(M_1)$.
**Step 4.** Check whether $\mu(e_s^{-1}(M_1)) \cap \mu(e_s(M_2))$ is not empty. If $\mu(e_s^{-1}(M_1)) \cap \mu(e_s(M_2))$ is not empty, output "YES", otherwise output "NO".

**Proposition 4.** *Let $M_1, M_2$ be two structured WF-modules. Modules $M_1$ and $M_2$ are resource compatible iff the output of the Agorithm 2 for $M_1, M_2$ is "YES".*

All steps of the Algorithm 2 except for Step 4 are already described. Checking, whether the intersection of multiset languages $\mu(e_s^{-1}(N_1))$ and $\mu(e_s(N_2))$ is empty, can be reduced to solving a set of linear equations.

We illustrate this algorithm by applying it to our motivation example from Section 3.

First we build quasi-regular expressions for bank and credit bureau services:
$e(WFM_1) = (!CD \circ ?CR)^* \circ ((?A^* \circ !E) \oplus (!CD \circ !E))$;
$e(WFM_2) = ?CD \circ (!CR \circ !A \circ ?CD)^* \circ ?E$.

Then we are to reduce these expressions to regular forms. Note, that $e(WFM_2)$ is already in a regular form, and hence $e(WFM_2) = e_s(WFM_2)$. To reduce $e(WFM_1)$ it is sufficient to apply distributivity law (equation 8):
$e_s(WFM_1) = ((!CD \circ ?CR)^* \circ ?A^* \circ !E) \oplus ((!CD \circ ?CR)^* \circ !CD \circ !E)$.
Then $e_s^{-1}(WFM_1) = ((?CD \circ !CR)^* \circ !A^* \circ ?E) \oplus ((?CD \circ !CR)^* \circ ?CD \circ ?E)$.

Now $r \in e_s{-}1(WFM_1)$ iff $r = n_1 \cdot ?CD + n_1 \cdot !CR + n_2 \cdot !A + ?E$, or $r = (n_1 + 1) \cdot ?CD + n_1 \cdot !CR + ?E$ for some natural $n_1, n_2 \geq 0$.

Similarly, $r \in e_s(WFM_2)$ iff $r = (n+1) \cdot ?CD + n \cdot !CR + n \cdot !A + ?E$ for some natural $n \geq 0$.

The multiset $\mu(e_s^{-1}(WFM_1)) \cap \mu(e_s(WFM_2))$ is not empty iff at least one of the following sets of simultaneous equations is consistent.

*Set 1 of equations* (corresponds to the first summand in $e_s{-}1(WFM_1)$:
$n_1 = n + 1$ (factors of $?CD$),

$n_1 = n$ (factors of $!CR$),
$n_2 = n$ (factors of $!A$),
$1 = 1$ (factors of $?E$).

*Set 2 of equations* (corresponds to the second summand in $e_s - 1(WFM_1)$:
$n_1 = n + 1$ (factors of $?CD$),
$n_1 = n$ (factors of $!CR$),
$0 = n$ (factors of $!A$),
$1 = 1$ (factors of $?E$).

Both these sets of equations are obviously inconsistent. So, we can immediately conclude, that the services of our bank example in Section 3 are not comparable.

Note, that Algorithm 2 has a polynomial on the size of WF-module time complexity.

## 6 Conclusion

In this paper we have presented a new approach for detecting incompatibility of Web services by analysis of resource compatibility of their structured workflow models. The resource compatibility is a necessary condition for soundness of composition of workflow models.

We introduce a language of quasi-regular expressions for describing multiset languages, and use this language for representing interface resources of structured workflow modules. Checking resource compatibility is then reduced to checking emptiness of intersection of multiset languages, represented by quasi-regular expressions. Thus checking resource compatibility can be solved in polynomial time.

Though resource compatibility is not sufficient for correct Web service interaction, the proposed procedure may be helpful for efficient detecting in consistences on the early stages of analysis of Web services.

## References


1. W. van der Aalst and K. van Hee. *Workflow Management: Models, Methods and Systems* MIT Press,2002.
2. W. M. P. van der Aalst, K. M. van Hee, A. H. M. ter Hofstede, N. Sidorova, et al. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing.* Vol. 23, Nr. 3, pages 333-363. 2011.
3. V. A. Bashkin and I. A. Lomazova. Resource equivalence in workflow nets. In *Proc. CS&P'2006 Workshop. Volume 1*, pages 80–91. Humboldt-Universität zu Berlin, 2006.
4. Cheng A., Esparza J., and Palsberg J.: Complexity results for 1-safe nets. In LNCS 761, 326-337. 1993.
5. X. Gong. Formal Analysis of Services Compatibility In *Computer Software and Applications Conference, 2009*, volume 2, pages 243–248. Seattle, WA, 2005.

6. K. van Hee, A. Serebrenik, N. Sidorova and M. Voorhoeve. Soundness of resource-constrained workflow nets. In *Proc. 26th Int. Conf. Application and Theory of Petri Nets, 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 250–267. Springer, 2005.
7. M. Kudlek, C. .Martìn Vide, Gh. Păun. Towards FMT (Formal Mucroset Theory). In *Multiset Processing*, volume 2235 of *Lecture Notes in Computer Science*, pages 123–133. Springer, 2001.
8. M. Kudlek, V. Mitrana. Closure properties of multiset languages. *Fundamenta Informaticae*, Vol. 49, Nr. 1-3, pages 191–203. 2002.
9. X. Li. A Petri Net Approach to Analyzing Behavioral Compatibility and Similarity of Web Services *Systems, Man and Cybernetics, Part A: Systems and Humans*, Vol. 41, pages 510–521, 2011.
10. I. A. Lomazova. Interacting workflow nets for workflow process re-engineering. *Fundamenta Informaticae*, Vol. 101, Nr. 1-2, pages 59–70, 2010.
11. A. Martens. Analyzing Web service based on business processes. In *Proc. Int. Conf. on Fundamental Approaches to Software Engineering, 2005*, volume 3442 of *Lecture Notes in Computer Science*, pages 19–33. Springer, 2005.
12. C. Stahl. Service substitution - a behavioral approach based on Petri nets. *PhD Thesis*, Eindhoven: Technische Universiteit Eindhoven, 2009.
13. P. Xiong. A Petri Net Approach to Analysis and Composition of Web Services *Systems, Man and Cybernetics, Part A: Systems and Humans*, Vol. 40, pages 376–387, 2010.