Service Discovery Using Communication Fingerprints

Olivia Oanea[1], Jan Sürmeli[2], and Karsten Wolf[1]

[1] Universität Rostock
18051 Rostock, Germany
{olivia.oanea,karsten.wolf}@uni-rostock.de
[2] Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin, Germany
suermeli@informatik.hu-berlin.de

Abstract. A request to a service registry must be answered with a service that fits in several regards, including semantic compatibility, non-functional compatibility, and interface compatibility. In the case of stateful services, there is the additional need to check behavioral (i.e. protocol) compatibility. This paper is concerned with the latter aspect. An apparent approach to establishing behavioral compatibility would be to apply the well-known technology of model checking to a composition of the provided service and the requesting service. However, this procedure must potentially be repeated for all provided services in the registry which may unprohibitively slow down the response time of the broker. Hence, we propose to insert a preprocessing step. It consists of computing an abstraction of the behavior for each published service that we call communication fingerprint. Upon request, we use the fingerprint to rule out as many as possible incompatible services thus reducing the number of candidates that need to be model checked for behavioral compatibility. The technique is based on linear programming and is thus extremely efficient. We validate our approach on a large set of services that we cut out of real world business processes.

1 Introduction

In a service oriented architecture, we expect a service *broker* to manage a service *registry*. The broker can be approached by a service *provider* or a service *requester*. Service providers want their service to be published such that it can later on be bound to a service requester. The service broker may extract useful information about the provided service. A service requester approaches the broker for extracting one of the registered services. Besides all kind of functional and non-functional properties that should match the request, it is important that the requesting service R and the service P selected by the broker have compatible behavior, i.e. their interaction should not run into problems such as deadlocks and livelocks. In this article, we propose an approach for supporting a service broker in this regard.

An apparent method for asserting deadlock and livelock freedom would be to model check [4] the composition $R \oplus P$ before shipping the URI of P to R. This is a rather expensive procedure which has a strong negative impact on the response time of the broker. For this purpose, we proposed an alternative check [20] which preprocesses

fragments of the state space to be checked at publish time. However, even this check must in worst case be applied to all compositions $R \oplus P_i$ where $\{P_1, \ldots, P_n\}$ is the set of registered services. Consequently, we need a complementing technique which narrows the set of registered services to be checked with R to a subset as small as possible.

To this end, we propose *communication fingerprints*. A communication fingerprint of service P collects constraints on the number of occurrences of messages in any correct run of a composed system that involves P. The fingerprint of a registered service is computed upon publishing a service. When a requester approaches the registry, its fingerprint is computed as well and matched with the fingerprints of the registered services. We show that fingerprint matching is a necessary condition for correct interaction, so the expensive model checking procedures need only be executed for matching services.

For computing communication fingerprints, we rely on Petri net models of services which can be automatically obtained [10] from specifications in the industrial language WS_BPEL [2]. For these models, we employ a technique called the *state equation*. The state equation provides a linear algebraic relation between two markings (states) m and m' as well as a sequence of transitions that transforms m into m'. Using the state equation, we derive constraints on the number of message occurrences. For matching fingerprints, we rely on a relation between the state equations of components and the state equation of the composed system that has been observed in [22, 16, 13].

The paper is organized as follows. We introduce Petri net models for services. We continue with the formal definition of fingerprints, their calculation, and their application. We present a case study that underpins the performance gain of our approach. Finally, we discuss further use cases for communication fingerprints.

2 Preliminaries

2.1 Mathematical Notations

For our approach, we need some notations for vectors as well as sequences of transitions.

Let A, B, C be sets where A is a set of (natural or integer) numbers. A^B denotes the set of all mappings from $B \to A$. A mapping $f \in A^B$ can be represented as an B-indexed vector over A. Let $f \in A^B$ and $f' \in A^C$, then $f + f' \in A^{B \cup C}$ is defined as follows: $(f + f')(b) \stackrel{\text{def}}{=} f(b)$ if $b \in B \setminus C$, $(f + f')(c) \stackrel{\text{def}}{=} f'(c)$ if $c \in C \setminus B$, and $(f + f')(x) \stackrel{\text{def}}{=} f(x) + f'(x)$, otherwise. The restriction $f_{|B'} \in A^{B'}$ of a mapping $f \in A^B$ to $B' \subseteq B$, is defined as follows: $f_{|B'}(b) \stackrel{\text{def}}{=} f(b)$. Let $f(B') \stackrel{\text{def}}{=} \{f(b) \mid b \in B'\}$. The extension $f_{|C'} \in A^{C'}$ of a mapping $f \in A^C$ to $C' \supseteq C$ is defined as $f_{|C'}(c) \stackrel{\text{def}}{=} f(c)$ if $c \in C$ and $f_{|C'} \stackrel{\text{def}}{=} 0$, otherwise. For $b \in B$, $\boldsymbol{b} \in A^B$ denotes the vector where $\boldsymbol{b}(b) = 1$ and $\boldsymbol{b}(y) = 0$ for $y \neq b$. $\mathbf{0} \in A^B$ denotes the *zero vector*. The *scalar product* of two vectors $x, x \in A^B$ is given by $x \cdot x' \stackrel{\text{def}}{=} \sum_{i \in B} x(i) \cdot x'(i)$.

B^* denotes the set of all sequences over B, including the empty sequence ϵ. The projection $\pi_B(\sigma) \in B^*$ of a sequence $\sigma \in C^*$ to elements of $B \subseteq C$, is defined as follows: $\pi_B(\epsilon) \stackrel{\text{def}}{=} \epsilon$, $\pi_B(t\sigma) = t\pi_B(\sigma)$ if $t \in B$, and $\pi_B(s\sigma) \stackrel{\text{def}}{=} \pi_B(\sigma)$, otherwise. For a sequence, $\sigma \in B^*$ its *count vector* $\Gamma(\sigma) \in \mathbb{N}^B$ assigns to $b \in B$ its number of occurrences in σ.

2.2 Message Exchange

We assume two global disjoint sets of bilateral channels that are used by services to exchange messages with each other: $\mathcal{C}_a$ and $\mathcal{C}_s$. The channels in $\mathcal{C}_a$ are used for *asynchronous* message exchange, i.e. sending and receiving events can be distinguished. We demand that no service both sends and receives messages over the same channel. The channels in $\mathcal{C}_s$ are used for *synchronous* message exchange, i.e. we can neither distinguish sending from receiving events nor initiator from reactor. Thus, those channels are not directed and message exchange can be understood as a handshake. We define $\Sigma^! \stackrel{\text{def}}{=} \{!a | a \in \mathcal{C}_a\}$, $\Sigma^? \stackrel{\text{def}}{=} \{?a | a \in \mathcal{C}_a\}$ and $\Sigma^\# \stackrel{\text{def}}{=} \{\#a | a \in \mathcal{C}_s\}$ as alphabets whose letters are identifiers for sending, receiving and synchronizing events, respectively, and $\Sigma \stackrel{\text{def}}{=} \Sigma^! \cup \Sigma^? \cup \Sigma^\#$ as the alphabet of all communication events. The interface of each service is thus a subset of Σ: It is a collection of all possible externally visible events caused by this service.

2.3 Open Net Syntax

We consider the control flow of a service to be modeled as an *open net*. Open net models of services can automatically be obtained [10] from specifications in the industrial language WS_BPEL [2]. An open net [19] is a Petri net [14] extended by means of asynchronous and synchronous communication: We label communication transitions with elements from Σ. For transitions that do not represent communication activities, we introduce the label τ. The actual message exchange is then modeled in the definition of *composition* below.

Definition 1. *An* open net $N = (P, T, F, M_0, \mathcal{M}_f, \lambda)$ *consists of*

- *two finite and disjoint sets P (of places) and T (of transitions);*
- *a flow relation $F \subseteq (P \times T) \cup (T \times P)$;*
- *an initial marking $M_0 \in \mathbb{N}^P$ and a set of final markings $\mathcal{M}_f \subseteq \mathbb{N}^P$, respectively;*
- *a labeling $\lambda : T \to \Sigma \cup \{\tau\}$ such that, for all $a \in \mathcal{C}_a$, $\{!a, ?a\} \not\subseteq \lambda(T)$.*

$\Sigma(N) \stackrel{\text{def}}{=} \lambda(T) \setminus \{\tau\}$ denotes the interface *of N. If $\Sigma(N) = \emptyset$, then we call N* closed.

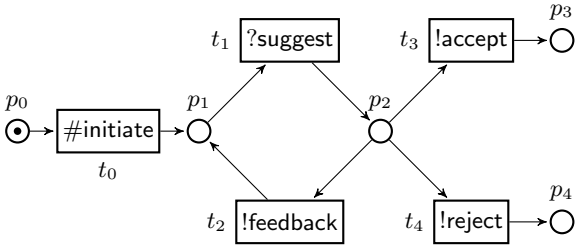

Fig. 1. Open net N_1 with final markings $\boldsymbol{p_3}$ and $\boldsymbol{p_4}$

In graphical representation, places and transitions are depicted as circles and rectangles, respectively. We annotate the transitions with their labels. Figure 1 shows an open net N_1. The modeled service exchanges messages synchronously over channel $\mathsf{initiate} \in \mathcal{C}_s$, sends messages over the channels $\mathsf{accept}, \mathsf{feedback}, \mathsf{reject} \in \mathcal{C}_a$ and receives over $\mathsf{suggest} \in \mathcal{C}_a$. The initial marking of N_1, is the marking $\boldsymbol{p}_0$. The final markings of N are not expressed graphically but can be found in the caption: $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$.

For $n \in P \cup T$, we consider ${}^\bullet n \stackrel{\text{def}}{=} \{n' | (n', n) \in F\}$ and $n^\bullet \stackrel{\text{def}}{=} \{n' | (n, n') \in F\}$. We extend this notion to ${}^\bullet Q \stackrel{\text{def}}{=} \bigcup_{n \in Q} {}^\bullet n$ and $Q^\bullet \stackrel{\text{def}}{=} \bigcup_{n \in Q} n^\bullet$ for $Q \subseteq P \cup T$.

To model the collaboration of services, we compose open nets. We compose only such nets that are syntactically compatible. Syntactical compatibility means that shared asynchronous channels are read by one open net and written by the other. Both services may use additional channels. We call syntactically compatible nets *partners*.

Definition 2 (partners, shared channels). *Two open nets $N = (P, T, F, M_0, \mathcal{M}_f, \lambda)$ and $N' = (P', T', F', M_0', \mathcal{M}_f', \lambda')$ are called* partners *iff $\Sigma(N)_{|\Sigma^!} \cap \Sigma(N')_{|\Sigma^!} = \Sigma(N)_{|\Sigma^?} \cap \Sigma(N')_{|\Sigma^?} = \emptyset$. We define the set of* shared channels *of two partners N, N' as $\mathcal{S}(N, N') \stackrel{def}{=} \{a \mid a \in \mathcal{C}_a, \{!a, ?a\} \subseteq \Sigma(N) \cup \Sigma(N')\} \cup \{c \mid c \in \mathcal{C}_s, \#a \in \Sigma(N) \cap \Sigma(N')\}$.*

Two partners are composed by introducing buffer places for shared asynchronous channels and by fusing transitions that have the same synchronous label. For simplicity, we assume that ingredients of different nets are disjoint, except for the interfaces. Figure 2 shows the composition of two open nets N_2, N_3 with interfaces $\Sigma(N_2) = \{?\mathsf{a}, !\mathsf{b}, \#\mathsf{c}, !\mathsf{d}\}$ and $\Sigma(N_3) = \{!\mathsf{a}, ?\mathsf{b}, \#\mathsf{c}\}$: Buffer places for each asynchronous channel in $\mathcal{S}(N_2, N_3) \cap \mathcal{C}_a = \{\mathsf{a}, \mathsf{b}\}$ are introduced; transitions of the partners with the same synchronous label are fused. The resulting composite has the interface $\Sigma(N_2 \oplus N_3) = \{\#\mathsf{c}\}$.

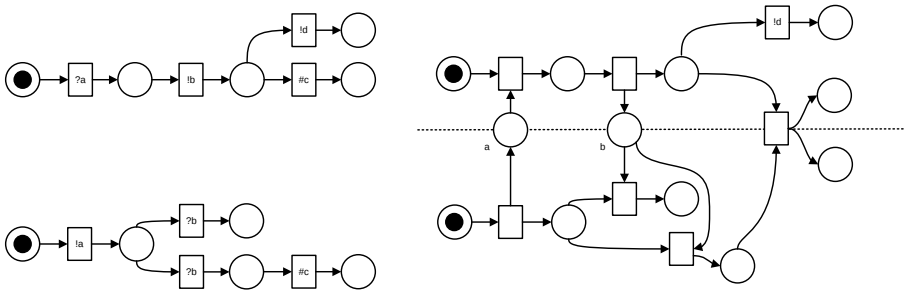

Fig. 2. Two open nets N_2, N_3 and their composition $N_2 \oplus N_3$

The formal definition of composition is a bit tedious, due to the presence of both synchronous and asynchronous communication. Formally, a channel name is used for the corresponding buffer place; the fusion of two transitions t, t' results in a new transition $[t, t']$.

Definition 3 (composition). *Let N, N' be partners. The* composition *of N and N' is the open net $N \oplus N' \stackrel{def}{=} (P^{N\oplus N'}, T^{N\oplus N'}, F^{N\oplus N'}, M_0^{N\oplus N'}, \mathcal{M}_f^{N\oplus N'}, \lambda^{N\oplus N'})$, where*

- $P^{N\oplus N'} \stackrel{def}{=} P \cup P' \cup P^a$, *where* $P^a = \mathcal{S}(N, N') \cap \mathcal{C}_a$ *(P^a represents the buffers for pending asynchronous channels);*
- $T^{N\oplus N'} \stackrel{def}{=} ((T \cup T') \setminus \{t \mid \lambda(t) = \#x \text{ with } x \in \mathcal{S}(N, N')\}) \cup \{[t, t'] | t \in T, t' \in T', \lambda(t) = \lambda'(t')\}$ *(a pair $[t, t']$ represents the joint occurrence of t and t');*
- $F^{N\oplus N'} \stackrel{def}{=} ((F \cup F') \cap ((P^{N\oplus N'} \cup T^{N\oplus N'}) \times (P^{N\oplus N'} \cup T^{N\oplus N'})) \cup F^a \cup F^s$, *where*
  - $F^a \stackrel{def}{=} \{(t, a) \in (T \cup T') \times P^a | (\lambda + \lambda')(t) = !a\} \cup \{(a, t) \in P^a \times (T \cup T'), (\lambda + \lambda')(t) = ?a\}$;
  - $F^s = \bigcup\{((\bullet t \cup \bullet t') \times \{[t, t']\}) \cup (\{[t, t']\} \times (t^\bullet \cup t'^\bullet)) \mid [t, t'] \in (T \times T') \cap T^{N\oplus N'}\}$;
- $M_0^{N\oplus N'} \stackrel{def}{=} M_0 + M_0' + \mathbf{0}$ *is the initial marking and* $\mathcal{M}_f^{N\oplus N'} \stackrel{def}{=} \{M + M' + \mathbf{0} | M \in \mathcal{M}_f, M' \in \mathcal{M}_f'\}$ *is the set of final markings;*
- $\lambda^{N\oplus N'}(t) =$
  - τ *if* $t \in T^{N\oplus N'}$ *and* $(\bullet t \cup t^\bullet) \cap P^a \neq \emptyset$,
  - τ *if* $t \in (T \times T') \cap T^{N\oplus N'}$,
  - $(\lambda + \lambda')(t)$, *otherwise.*

The composition of two open nets may or may not be closed. The intuition of a composite comprising a not-empty interface is that of a composite component that is to be composed with other components.

2.4 Open Net Semantics

We define the behavior of an open net as that of the underlying Petri net: A transition $t \in T$ is *enabled* in marking M if $M(p) \geq 1$ for all places $p \in \bullet t$. An enabled transition may fire yielding a marking m' where $M'(p) = M(p) - 1$ for all places $p \in \bullet t \setminus t^\bullet$, $M'(p) = M(p) + 1$ for all places $p \in t^\bullet \setminus \bullet t$ and $M'(p) = M(p)$ for all other places, which is denoted by a *step* $M \xrightarrow{t} M'$. This notion can be extended to finite sequences of steps $M_0 \xrightarrow{t_0} M_1 \xrightarrow{t_1} \ldots \xrightarrow{t_n} M_{n+1}$, denoted as $M_0 \xrightarrow{t_0 t_1 \ldots t_n} M_{n+1}$. We call $t_0 t_1 \ldots t_n$ a *firing sequence* of N, finishing in M_{n+1}. An example firing sequence of N_1 depicted in Fig. 1 is $t_0 t_1 t_2 t_1 t_3$, which results in marking $\boldsymbol{p_3}$.

Definition 4 (M-behavior, language of an open net). *Let N be an open net and M be a marking of N. We call* $\mathrm{beh}(N, M) \stackrel{def}{=} \{\sigma \mid M_0 \xrightarrow{\sigma} M\}$ *the* M-behavior *of N. We define the* language *of N as* $\mathcal{L}(N) \stackrel{def}{=} \bigcup_{M \in \mathcal{M}_f} \mathrm{beh}(N, M)$.

For example, the language of N_1 depicted in Fig. 1 includes the firing sequence $t_0 t_1 t_2 t_1 t_3$ but not the firing sequence $t_0 t_1 t_2 t_1$, because the latter results in $\boldsymbol{p_2} \notin \mathcal{M}_f$.

Let N and N' be two partners and $N \oplus N'$ their composition. Then every transition sequence of the composite can be projected on transition sequences of the components, yielding a transition sequence of the component. The formal definition is similar to the definition of classical projection on sequences, only differing in the handling of fused transitions.

Definition 5 (projection of transition sequences). *We define the* projection *of transition sequence σ of $N \oplus N'$ to transitions of N as $\sigma_{[N]} \stackrel{def}{=} \lfloor \sigma \rfloor$ where*

- $\lfloor \epsilon \rfloor = \epsilon$,
- *for all $t \in T$ and $\sigma' \in (T^{N \oplus N'})^*$, $\lfloor t\sigma' \rfloor = t \lfloor \sigma' \rfloor$,*
- *for all $t \in T'$ and $\sigma' \in (T^{N \oplus N'})^*$, $\lfloor t\sigma' \rfloor = \lfloor \sigma' \rfloor$, and*
- *for all $[t, t'] \in T \times T'$ and $\sigma' \in (T^{N \oplus N'})^*$, $\lfloor [t, t']\sigma' \rfloor = t \lfloor \sigma' \rfloor$.*

We observe that the projection of a firing sequence in the composition is a firing sequence of the component it is projected to. This enables us to analyze an open net in isolation and draw first conclusions on its behavior with a future partner.

Lemma 1. *Let N, N' be partners. Then, $\mathcal{L}(N \oplus N')_{[N]} \subseteq \mathcal{L}(N)$.*

Counting the interaction events that occur along a firing sequence of an open net, we obtain a channel usage vector. We can translate the language of an open net to the channel usage vectors. The result is an abstraction of the interaction behavior of the open net.

Definition 6 (channel usage). *Let N be an open net and $\sigma \in T^*$. We define the* channel usage *of σ with respect to N as $\Psi(N, \sigma) \in \mathbb{N}^{\mathcal{C}_a \cup \mathcal{C}_s}$ where for all $c \in \mathcal{C}_a \cup \mathcal{C}_s$, $\Psi(N, \sigma)(c) \stackrel{def}{=} \sum_{t \in T, \lambda(t) \in \{!c, ?c, \#c\}} \Gamma(\sigma)(t)$. Likewise, we define the* channel usage of *N as $\Psi(N) \stackrel{def}{=} \bigcup_{\sigma \in \mathcal{L}(N)} \Psi(N, \sigma)$.*

The before mentioned firing sequence $\sigma = t_0 t_1 t_2 t_1 t_3$ of open net N_1 in Fig. 1 can be abstracted to its channel usage, resulting in vector $\Psi(N, \sigma) = v$ with $v(\mathsf{initiate}) = v(\mathsf{feedback}) = v(\mathsf{accept}) = 1$, $v(\mathsf{suggest}) = 2$, and $v(\mathsf{suggest}) = 0$.

In a final marking of a composed open net, each message buffer place is empty by definition. Thus, every message that is sent asynchronously is also received before reaching a final state. Thus, for all firing sequences in the composed open net that finish in a final marking, the two partners agree on the channel usage.

Lemma 2. *Let N, N' be partners. Then, for any $s \in \mathcal{S}(N, N')$ and $\sigma \in \mathcal{L}(N \oplus N')$, $\Psi(N, \sigma_{[N]})(s) = \Psi(N', \sigma_{[N']})(s)$.*

Although syntactically compatible, not every two partners are semantically compatible. There exist different notions for semantical compatibility, one of them is weak termination: From any reachable marking in the composition, a final marking is still reachable.

Definition 7 (weakly terminating). *An open net N is weakly terminating if, for every firing sequence σ of N, there exists some σ' such that $\sigma\sigma' \in \mathcal{L}(N)$.*

Definition 8 (compatibility w.r.t. weak termination). *Let N, N' be partners. N and N' are* compatible w.r.t. weak termination*, if either*

1. *$N \oplus N'$ is closed and and weakly terminating, or*
2. *$N \oplus N'$ is not closed and there exists a partner N'' of $N \oplus N'$, such that $(N \oplus N') \oplus N''$ is closed and weakly terminating.*

The nets N_2 and N_3 in Figure 2 are not compatible with respect to weak termination.

3 Communication Fingerprints

A communication fingerprint is a finitely representable over-approximation for the set of channel usage vectors which can be realized in a component. It abstracts from the order in which messages are sent or received and from any internal control flow. Syntactically, a communication fingerprint is a boolean combination of constraints. The simplest constraints are lower and upper bounds for the number of occurrences of a message in any sequence of $\mathcal{L}(N)$. For N_1 in Fig 1, $\mathsf{suggest} \leq 5$. However, these constraints do not suffice in the case of services with cycles in their control flow as the lower bound tends to be 0 and the upper bound tends to be infinity. For this reason, we also introduce constraints that compare the *difference* between message occurrences. If one message is sent and another one is received on a cycle, the difference between the number of occurrences is finite even if the number of occurrences of each message in isolation is not bounded. For N_1, we can think of $\mathsf{suggest} - \mathsf{feedback} = 1$. Pushing this idea to the limits, a constraint can finally compare any linear combination of message occurrences with a number. A linear combination of message occurrence counts can be represented as a vector $x \in \mathbb{A}^{\mathcal{C}_a \cup \mathcal{C}_s}$ where $\mathbb{A}$ is a suitable set of numbers. In the following, let $\mathbb{A} \in \{\mathbb{Z}, \mathbb{Q}\}$. As an example, $\mathsf{suggest} - \mathsf{feedback}$ can be written as the vector v with $v(\mathsf{suggest}) = 1$, $v(\mathsf{feedback}) = -1$ and $v(c) = 0$ for $c \in \{\mathsf{initiate}, \mathsf{accept}, \mathsf{reject}\}$. Such a vector, together with an upper bound, forms a *constraint*. While the notion as a vector with one upper bound eases up the formalism, we often write constraints as described above.

Definition 9 (constraint, formula). *A* constraint *is a pair* $(v, k) \in \mathbb{A}^{\mathcal{C}_a \cup \mathcal{C}_s} \times \mathbb{A}$. *A* formula *is any boolean combination of constraints and the literals* true *and* false.

Note that our formalism covers constraints of the form $a = 1$ or $a > 3$. These constraints can be represented by the former, by using the classical transformation rules for inequalities: $a = 1 \Leftrightarrow (a \leq 1) \wedge (1 \leq a)$, $1 < a \Leftrightarrow -a < -1$.

A formula φ is evaluated in a channel usage vector x. If it evaluates to true, we say that x models φ, denoted as $x \models \varphi$.

Definition 10 ($\models$). *Let φ be a formula.*

1. *($\mathrm{true}, \mathrm{false}$) $x \models \mathrm{true}$ and $x \not\models \mathrm{false}$.*
2. *(atomic formulae) Let $\varphi = (v, k)$. If $\sum_{c \in \mathcal{C}_a \cup \mathcal{C}_s} v(c) \cdot x(c) \leq k$, then $x \models \varphi$, else $x \not\models \varphi$.*
3. *(conjunction) Let $\varphi = \varphi_1 \wedge \varphi_2$. If $x \models \varphi_1$ and $x \models \varphi_2$, then $x \models \varphi$, else $x \not\models \varphi$.*
4. *(disjunction) Let $\varphi = \varphi_1 \vee \varphi_2$. If $x \models \varphi_1$ or $x \models \varphi_2$, then $x \models \varphi$, else $x \not\models \varphi$.*
5. *(negation) $x \models \neg\varphi$ if and only if $x \not\models \varphi$.*

As an example, a channel usage vector $\boldsymbol{a}$ models $(\boldsymbol{a}, 5)$. Based on the model relation, we define the set of all models of a formula and feasibility of a formula.

Definition 11 ($\Psi(\varphi)$, feasible, infeasible). *Let φ be a formula. We define the* set of models *of φ as $\Psi(\varphi) \stackrel{def}{=} \{x \mid x \models \varphi\}$. We call φ* feasible *if $\Psi(\varphi) \neq \emptyset$ and* infeasible*, otherwise.*

We observe that the boolean operators $\wedge, \vee, \neg$ correspond to the set operations intersection, union and complement.

Lemma 3. *Let φ, φ' be formulae.*

1. $\Psi(\varphi \wedge \varphi') = \Psi(\varphi) \cap \Psi(\varphi')$,
2. $\Psi(\varphi \vee \varphi') = \Psi(\varphi) \cup \Psi(\varphi')$, *and*
3. $\Psi(\neg\varphi) = \mathbb{A}^{\mathcal{C}_a \cup \mathcal{C}_s} \setminus \Psi(\varphi)$.

A communication fingerprint of an open net N is a formula that overapproximates the set of channel usage vectors *realizable* in N.

Definition 12 (Communication fingerprint). *Let N be an open net and φ be a formula. φ is a communication fingerprint (or fingerprint for short) of N, if and only if $\Psi(\varphi) \supseteq \Psi(N)$.*

The formulae $\mathsf{initiate} = 1 \wedge \mathsf{suggest} - \mathsf{feedback} = 1 \wedge \mathsf{accept} + \mathsf{reject} = 1$ and $\mathsf{initiate} \leq 1 \wedge \mathsf{suggest} - \mathsf{feedback} = 1 \wedge (\mathsf{reject} = 1 \vee \mathsf{accept} = 1)$ are both example fingerprints of N_1 in Fig. 1.

Given the fingerprints of two partners, we can semi-decide compatibility of the partners by checking the conjunction of the fingerprints for feasibility.

Theorem 1. *Let N_1, N_2 be partners. Let φ_1, φ_2 fingerprints of N_1, N_2, respectively. If N_1, N_2 are compatible (with respect to weak termination), $\varphi_1 \wedge \varphi_2$ is feasible.*

Proof. Let N_1 and N_2 be compatible with respect to weak termination. Then at least one final marking must be reachable in the composition, say $M_0 \xrightarrow{\sigma} M_f$. The two projections σ_1 and σ_2 to N_1 and N_2, respectively, must agree on each other, i.e. $x^* = \Psi(\sigma_1, N_1) = \Psi(\sigma_2, N_2)$. By Def. 12, $x^* \models \varphi_1$ and $x^* \models \varphi_2$. Consequently, $\varphi_1 \wedge \varphi_2$ is feasible.

Consider the fingerprint $\varphi_R = \mathsf{suggest} - \mathsf{feedback} = 0 \wedge \mathsf{accept} + \mathsf{reject} = 1$ of a partner of open net N_1 from Fig. 1. A corresponding open net could expect feedback on any suggestion it does. Let φ be the before mentioned fingerprint $\mathsf{initiate} = 1 \wedge \mathsf{suggest} - \mathsf{feedback} = 1 \wedge \mathsf{accept} + \mathsf{reject} = 1$ of N_1. Inspecting the conjunction $\varphi \wedge \varphi_R$, we find that it is infeasible: No channel usage vector can model both $\mathsf{suggest} - \mathsf{feedback} = 0$ and $\mathsf{suggest} - \mathsf{feedback} = 1$. Therefore, the two services are incompatible.

In Sect. 4, we propose an efficient way to obtain a communication fingerprint for an open net N. Then, in Sect. 5, we shall discuss how to reduce feasibility of $\varphi_1 \wedge \varphi_2$ to the feasibility of a set of systems of linear inequalities.

4 Computation of fingerprints

In this section we show how fingerprints can be computed with standard Petri technique called the *state equation* [8]. The state equation is a system of linear equalities based on the incidence matrix of the Petri net, its initial marking and a target marking. The solution set of the state equation of N for the target marking M is syntactically a set of transition occurrence vectors and semantically an over-approximation for $\mathrm{beh}(N, M)$. Before we elaborate on its use for computing fingerprints, we formally introduce the incidence matrix and the state equation.

Definition 13 (incidence matrix). *Let N be an open net. We define the* incidence matrix *of N as $I(N) \in \mathbb{Z}^{P \times T}$ with $I(N)(p,t) = 1$ if $t \in {}^\bullet p$ and $t \notin p^\bullet$, $I(N)(p,t) = -1$ if $t \notin {}^\bullet p$ and $t \in p^\bullet$, and $I(N)(p,t) = 0$, otherwise.*

Based on the incidence matrix, we can introduce the state equation.

Proposition 1 (Petri net state equation). *Let M_1 and M_2 be a markings of N and σ a transition sequence σ with count vector $\Gamma(\sigma)$. If $M_1 \xrightarrow{\sigma} M_2$ then*

$$M_1 + I(N) \cdot \Gamma(\sigma) = M_2.$$

If no particular sequence σ is given, the vector $\Gamma(\sigma)$ can be replaced by unknowns. This transforms the state equation into a linear system of equations. The solutions are count vectors among which are all count vectors of firing sequences that transform M_1 into M_2. Replacing M_1 with the initial marking and M_2 with any final marking thus yields an overapproximation of all count vectors of sequences in $\mathcal{L}(N)$ which can be easily translated into channel usage vectors.

Definition 14 (Overapproximated behavior). *With $\mathrm{beh}^*(N,M)$, denote the set of all sequences that fit the state equation, i.e. $\mathrm{beh}^*(N,M) = \{\sigma \in T^* \mid M_0 + I(N) \cdot \Gamma(\sigma) = M\}$. Let $\mathrm{beh}^*(N) = \bigcup_{M_f \in \mathcal{M}_f} \mathrm{beh}^*(N, M_f)$.*

In our context, the state equation thus translates into:

Theorem 2. $\mathrm{beh}^*(N) \supseteq \mathcal{L}(N)$.

We are now ready to compute a communication fingerprint for N. We proceed as follows. First, we assume that the left hand side of a constraint (a formal sum of message counts) and one of the final markings is given. We use the state equation to compute an upper bound for the evaluation of the formal sum. i.e. an as small as possible constant right hand side for the constraint. Then we aggregate the results for all final markings of N and for a given set of formal sums. In this approach, the set V of formal sums to be used is a parameter. Later in this section, we discuss possible options for choosing this set. Our approach results in the following fingerprint.

Definition 15 (Computed fingerprint). *Let N be an open net. Let $V \subseteq \mathbb{A}^{\mathcal{C}_a \cup \mathcal{C}_s}$. Let $\mathcal{M}_f^* = \{M \in \mathcal{M}_f \mid \mathrm{beh}^*(M) \neq \emptyset\}$. Then the computed fingerprint of N w.r.t. V is*

$$\bigvee_{M \in \mathcal{M}_f^*} \bigwedge_{v \in V} (v, k_{M,v})$$

where $k_{M,v}$ is the solution of the following linear program: Minimize $k = v \cdot y$ in $M_0 + I(N) \cdot x = M$ and, for all $c \in \mathcal{C}_a \cup \mathcal{C}_s$, $y(c) = \sum_{t:\lambda(t) \in \{!c, ?c, \#c\}} x(t)$.

Theorem 3. *The computed fingerprint is a valid fingerprint of N, regardless of V.*

Proof. The equation $M_0 + I(N) \cdot x = M$ represents the state equation for final marking M, so its solution space covers all transition count vectors of sequences in $\mathrm{beh}(N,M)$. Thus, the constraint $(v, k_{M,v})$ is valid under the assumption that M is the only final marking. The conjunction of constraints is valid under the same assumption as all single constraints are valid. The disjunction is finally valid for N as each sequence in $\mathcal{L}(N)$ must reach one of the final markings in N. □

As linear programming is a standard technique, and the remaining steps are simple aggregations, we skip the presentation of pseudo-code for the procedure. We implemented the proposed procedure in a tool called *Linda* [9] which is freely available. Run times concerning Linda will be presented in Sect. 6. Meanwhile, we discuss several strategies for choosing the set V for formal sum that are used in the procedure sketched above.

4.1 Semantically motivated constraints

A semantic constraint bases on a formal sum which expresses a relation between message counts known to the user. Semantical constraints may be useful in an interactive computation of a fingerprint. One possible semantical constraint is *mutual exclusion* of a number of events. As an example, a service might send either a reject or an accept message. We could use the formal sum $\mathsf{reject} + \mathsf{accept}$. An upper bound of 1 would indicate that the two events are mutually exclusive. Additionally, there might exist pairs of messages that occur *equally often*. For example, a login is followed by a log off at some point. Or each request is either granted or denied. Corresponding formal sums would be $\mathsf{login} - \mathsf{logoff}$ and $\mathsf{request} - (\mathsf{granted} + \mathsf{denied})$. An upper bound of 0 then indicates that the events occur equally often. Maybe a service is not as restrictive and accepts that a logoff message is not sent, but it expects not to receive more logoff messages than login messages. In this case, an upper bound greater than 0 is a useful indicator.

4.2 Geometrically motivated constraints

There are certain formal sums that represent particular geometrical shapes. For instance, if only single message counts are compared, the solutions correspond to hyper-cubes. If only differences are compared, the space of solutions corresponds to bounded differences, and so on up to the most expressive represented by convex polyhedra. For several such classes of constraints, there exist alternative representations and efficient procedures for checking feasibility as well as for computing conjunction and disjunction on that alternative representation. Thus, it makes sense to favor formal sums that belong to such specific classes. A survey on known geometrically motivated classes of constraints can be found in [15, 3].

4.3 Constraints with finite bounds

Constraints in a fingerprint are typically useful when their upper bound is finite. Otherwise, all sequences are realizable and constraints cannot not discriminate any open net. In this subsection we give a sufficient condition for finiteness of the upper bound which uses the Petri net concept of t-invariants.

Formally, a t-invariant is a solution of $I(N) \cdot y = \mathbf{0}$. Intuitively, a t-invariant is a transition occurrence vector, such that firing the transitions accordingly from marking M yields the marking M again (i.e. the state equation has the form $M + I(N) \cdot \Gamma(\sigma) = M$). A t-invariant which is realizable in a firing sequence represents the count vector of a cycle in the state space which, in turn, can be executed arbitrarily often. Cycles hence

cause infinite upper bounds to any constraint where the t-invariant itself does not yield the value 0.

For exploiting this observation, consider a t-invariant x and a solution y of the state equation with some target marking M. Then $x + k \cdot y$ is a solution as well, for any $k \in \mathbb{A}$. This yields infinite bounds for several terms. Let $\boldsymbol{t}$ be a t-invariant and a transition t with label $!a$. Our approach results in an infinite upper bound for formal sum a, even if t fires only finitely often or never at all.

Lemma 4. *Let φ be a formula, $v \in \mathbb{A}^{\mathcal{C}_a \cup \mathcal{C}_s}$. Then, $\Psi(\varphi) = \Psi(\varphi \vee (v, \infty))$.*

Next we state a sufficient condition for a formal sum having a finite bound based on the set of all minimal t-invariants (or a super set). A t-invariant x is called *minimal* if it cannot be written as a positive linear combination of t-invariants. In a first step, we translate the t-invariants into channel usage vectors. Then, we build a system of linear equations based on these vectors so that the solution set is a set of terms for which our approach results in finite bounds.

Lemma 5. *Let N be an open net and M be a marking of N. Let $A \subseteq \mathbb{A}^T$ be a set of minimal t-invariants of N. Let $X \supseteq A$ and $Y = \Psi(X)$. Let $m \in \mathbb{A}^{Y \times (\mathcal{C}_a \cup \mathcal{C}_s)}$, so that $m(y, c) = y(c)$. Then, for each $v \in \mathbb{A}^{\mathcal{C}_a \cup \mathcal{C}_s}$ with $m \cdot v = \mathbf{0}$ holds: $\min(\{v^\top \cdot \Psi(N, x) \mid x \in \mathrm{beh}^*(N, M)\})$ is finite.*

Note that the above condition for finite bounds is not a necessary one; however we can use it to create generic formal sums which lead to finite bounds by construction.

5 Matching of fingerprints

We call two fingerprints *matching*, if their conjunction is feasible. If fingerprints of two services do not match, they are incompatible; else, we can not give a conclusive answer to the question of compatibility. In this section, we discuss how matching of two fingerprints given in DNF can be decided efficiently. In the following, we only assume fingerprints in DNF that contain at least one conjunctive clause; any other case is trivial. The main idea is that feasibility of a conjunction of atomic formulae is equivalent to feasibility to a corresponding system of linear inequalities.

Definition 16 (system of linear inequalities associated to formulae). *Let F be a set of atomic formulae and $\varphi = \bigwedge_{f \in F} f$. Then, the asociated system of linear inequalities is denoted with $\mathrm{sli}(\varphi) \stackrel{def}{=} (A, b)$ where*

- *$A \in \mathbb{Z}^{F \times \mathcal{C}_a \cup \mathcal{C}_s}$ such that $A((v, k), c) = v(c)$, and*
- *$b \in \mathbb{Z}^F$ such that $b((v, k)) = k$.*

Please note, that this definition also works for an empty set of atomic formulae, yielding $\varphi = \mathrm{true}$ and the empty system of linear inequalities $\mathrm{sli}(\mathrm{true})$.

Lemma 6. *Let F be a set of atomic formulae and $\varphi = \bigwedge_{f \in F} f$. Then, φ is feasible if and only if $\mathrm{sli}(\varphi)$ is feasible.*

In case a conjunction of two formulae in DNF is feasible, there exists a conjunctive clause of each formula so that their conjunction is feasible. Such a conjunctive formula is by construction a conjunction of atomic formulae.

Lemma 7. *Let $\varphi = \varphi_1 \vee \ldots \vee \varphi_m$ and $\varphi' = \varphi'_1 \vee \ldots \vee \varphi'_n$. Then, $\varphi \wedge \varphi'$ is feasible if and only if there exist $i, j \in \mathbb{N}$ with $1 \leq i \leq m, 1 \leq j \leq n$ and $\mathrm{sli}(\varphi_i \wedge \varphi'_j)$ is feasible.*

To decide matching, we have to check $m \cdot n$ systems of linear inequalities in the worst case. Following our approach, a fingerprint of an open net is a formula with less or equal conjunctive clauses than final markings. Thus, $m \leq |\mathcal{M}_f|$ and $n \leq |\mathcal{M}'_f|$. In our experience, services have a very small number of final markings. We thus think that matching of two fingerprints can work as an efficient quick check that results in either incompatible or inconclusive.

Fingerprint matching can be done with a tool called *Yasmina* [21]. Run times concerning Yasmina are presented in Sect. 6.

6 Case study

In the introduction, we discussed the following scenario: Given one requester service R and a service registry, the task is to find a service in the registry that is compatible with the requester. We assume that the registry contains an open net model and a fingerprint for each available service. To find a compatible service, we check each candidate C for compatibility with R until we find a compatible partner for R: First, we decide matching of the fingerprints. In the case that this check yields inconclusive, we decide compatibility with model checking.

For validating this approach, we had to build up a large number of services. As a sufficiently large set of actual services was not available to us, we generated "close to real" services as follows. We started with a large set of *real* industrial business processes available to us. They have been modeled using the IBM WebSphere Business Modeler, then anonymized so they could be made available to us and finally translated into Petri net models. Anonymization had been done by replacing semantical annotations to the activities by generic strings. The used set of business processes has been analyzed in [6].

Each process has then been decomposed into two asynchronously communicating parts which then service as services. Decomposition follows the idea of [22, 12], however, we decomposed only into two components instead of as many as possible components using an extension of *Diane* [5] tailored for the specific libraries. For many processes, there exist several possibilities for decomposing them, so we obtain a rather big set of services. For several processes, the obtained services are actually infinite state systems. In this case, we added artificial capacities to the unbounded places. Two services that have been obtained from a weakly terminating business process model are not necessarily compatible. In addition, several original models have not been compatible in the first place. The set of available processes is organized into libraries. Each library contains models from similar business fields. For this reason, we experimented with separate sample sets, one for each library.

For all model checking tasks, we used *LoLA* [11]. The tools for fingerprint computation and matching were *Linda* [9] and *Yasmina* [21]. The first table shows some numbers

concerning the processes in the used libraries with their characteristics: For each library, we list the number of processes and the resulting composites. The fourth column shows how many of those composites were actually weakly terminating, while the fifth displays for how many composites the fingerprint matching returned inconclusive. The number of weakly terminating composites is rather low (as explained above). This ratio seems, however, to be realistic in a service registry as a diverse registry should contain rather many incompatible services to a given requester.

Table 1. Testbed statistics and analysis results

Library	# Processes	# Composites	# Weakly terminating	# Matching inconclusive
A	127	2412	252	672
B1	43	2066	20	597
B2	49	592	25	209
B3	105	3460	210	1165

In the second table, we compare the run times of a pure state based approach with the run times of the proposed fingerprint based approach. To this end, we checked all pairs of partner services that stem from the same library. The reported times are the overall times for executing all these checks within a library: The second column lists total amount of time for model checking the composites. The third column states the run time of the fingerprint computation. The fourth column displays the time needed for fingerprint matching. For all inconclusive results, we used model checking, resulting in run times as given in the fifth column. Finally, the sixth column displays the total amount for the fingerprint based compatibility check, which does not include the run time for fingerprint computation.

Table 2. State-based approach vs. fingerprint approach

Library	**State-based**	**Fingerprint**			
	Total	Computation	Matching	Model checking	Total
A	>48h	2m49s	30s	28h	≈28h
B1	18m3s	2m57s	29s	6m9s	6m38s
B2	30m43s	53s	12s	16s	28s
B3	>36h	11m6s	1m29s	2h	≈2h

We see that for about two thirds of the individual problems, the fingerprint check tells that these services are incompatible. These are the problem instances for which it is not necessary to perform a subsequent model checking. For the remaining services, model checking must be applied in any case. Hence, the speed-up can be seen in comparing the overall time of model checking *all instances* with the overall time of all fingerprint matchings plus the overall time for those model checking runs where the fingerprint check

was inconclusive. The runtime of the fingerprint matching alone does not contribute significantly to the overall run time and the fingerprint approach requires only about one third of the state-based approach.

7 Other Applications

Service discovery is not the only possible application of communication fingerprints. In this section, we sketch other possible application areas.

Organizing a registry To reduce complexity in the find scenario, we can substitute the linear approach to a binary tree traversal: Each node is annotated with a formula. Each leaf of the tree is a fingerprint of a service in the repository, the other nodes are disjunctions. Thus, the formula of each node is the fingerprint of a set of services and the tree can be traversed from the root to find a candidate.

Checking substitutability Two services are substitutable if all compatible partners of the old service are compatible to the new one as well. This implies that the fingerprints of the original and the substitute have to match.

Checking adaptability When adapting two services, a mediator is introduced to realize proper interaction. The mediator can be build by creating an *engine* from a set of semantical message transformation rules. Then, we compose the two services and the engine and synthesize a partner, such that the four services are compatible. We can use a fingerprint quick check to decide if the semantic rules are sufficient to adapt the services.

8 Conclusion

In this paper we have considered service communication fingerprints as an approach for pre-selecting appropriate interaction partners with respect to weak termination. We used the state equation of the underlying Petri net to derive constraints over the set of synchronous and asynchronous message event occurrences. Communication fingerprints are considerably small in comparison to the state space of a service. We considered a simple and efficient procedure for obtaining a suitable (not necessarily optimal) communication fingerprint. Matching fingerprints amounts to solving linear programming problems. Our experiments show that the fingerprint approach can significantly speed up service discovery.

Our approach is complementary to testing observed behavior against model behavior using frequency profiles [1] and keeping repositories of behavioral profiles [18, 17]. Both approaches apply to monolithic workflow and are restricted to transition occurrences. Our approach is different from compositional analysis of invariants of functional nets [22]: We analyze communication patterns which are inherently related to communication.

For future work, we shall consider the application of fingerprints in the synthesis of livelock-free partners. Further, we shall experiment how service communication fingerprint registries created to store subclasses of potentially compatible partners contributes to speeding up operations on behavioral registry [7].

Acknowledgements Olivia Oanea is supported by German Research Foundation (DFG) under grant WO 1466/11-1

References

1. W. M. P. van der Aalst. Matching observed behavior and modeled behavior: an approach based on Petri nets and integer programming. *Decis. Support Syst.*, 42(3):1843–1859, 2006.
2. A. Alves et all. Web Services Business Process Execution Language Version 2.0. Technical Report CS-02-08, OASIS, 2007.
3. R. Clarisó and J. Cortadella. The octahedron abstract domain. *Sci. Comput. Program.*, 64(1):115–139, 2007.
4. E.M. Clarke, D. Peled, and O. Grumberg. *Mode Checking*. MIT Presss, 1999.
5. Diane. `http://service-technology.org/diane`.
6. D. Fahland, C. Favre, B. Jobstmann, J. Koehler, N. Lohmann, H. Völzer, and K. Wolf. Instantaneous soundness checking of industrial business process models. In *BPM 2009*, volume 5701 of *LNCS*. Springer-Verlag, 2009.
7. K. Kaschner and K. Wolf. Set algebra for service behavior: Applications and constructions. In *BPM 2009*, volume 5701 of *LNCS*, pages 193–210. Springer-Verlag, 2009.
8. K. Lautenbach. *Liveness in Petri Nets.* St. Augustin: Gesellschaft fr Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-75-02.1, 1975.
9. Linda. `http://service-technology.org/linda`.
10. N. Lohmann. A feature-complete Petri net semantics for WS-BPEL 2.0. In *Web Services and Formal Methods, Forth International Workshop, WS-FM 2007, Brisbane, Australia, September 28-29, 2007, Proceedings*, volume 4937 of *LNCS*, pages 77–91. Springer-Verlag, 2008.
11. Lola. `http://service-technology.org/lola`.
12. S. Mennicke, O. Oanea, and K. Wolf. Decomposition into open nets. In *AWPN 2009*, volume 501 of *CEUR Workshop Proceedings*, pages 29–34. CEUR-WS.org, 2009.
13. O. Oanea and K. Wolf. An efficient necessary condition for compatibility. In *ZEUS*, volume 438 of *CEUR Workshop Proceedings*, pages 81–87. CEUR-WS.org, 2009.
14. W. Reisig. *Petri nets. An Introduction*. Springer, 1985.
15. A. Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience series in discrete mathematics. John Wiley & Sons, 1986.
16. Jan Sürmeli. Profiling services with static analysis. In *AWPN 2009 Proceedings*, volume 501 of *CEUR Workshop Proceedings*, pages 35–40. CEUR-WS.org, 2009.
17. M. Weidlich, A. Polyvyanyy, J. Mendling, and M. Weske. Efficient computation of causal behavioural profiles using structural decomposition. In *PETRI NETS 2010*, volume 6128 of *LNCS*. Springer-Verlag, 2010.
18. M. Weidlich, M. Weske, and J. Mendling. Change propagation in process models using behavioural profiles. In *SCC '09*, pages 33–40. IEEE, Sept. 2009.
19. K. Wolf. Does my service have partners? *LNCS ToPNoC*, 5460(II):152–171, March 2009. Special Issue on Concurrency in Process-Aware Information Systems.
20. K. Wolf, C. Stahl, J. Ott, and R. Danitz. Verifying livelock freedom in an SOA scenario. In *ACSD 2009*, pages 168–177. IEEE, 2009.
21. Yasmina. `http://service-technology.org/yasmina`.
22. D. A. Zaitsev. Compositional analysis of Petri nets. *Cybernetics and Systems Analysis, Volume 42, 1, 2006*, pages 126–136, 2006.