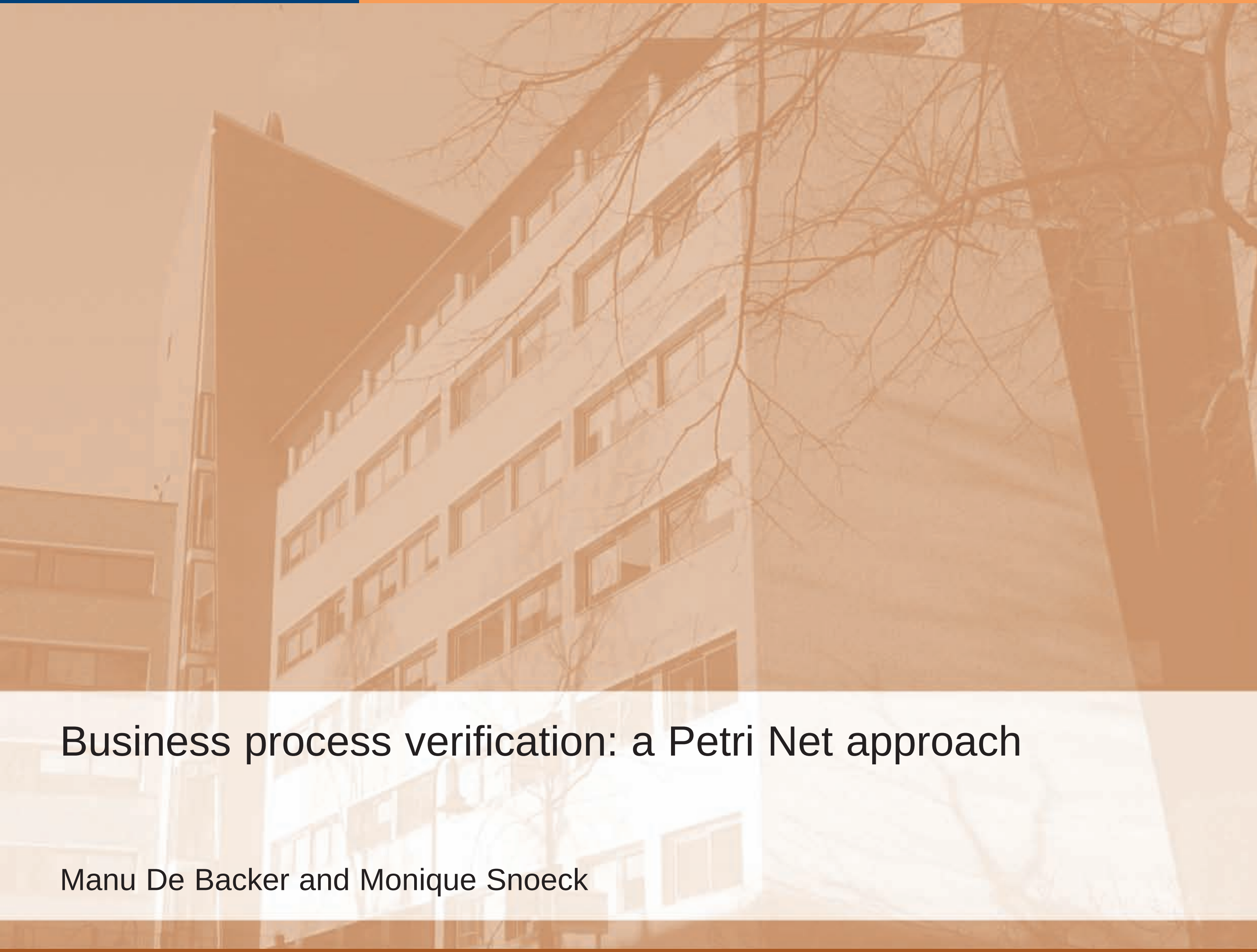
KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Economics and Applied Economics



Business process verification: a Petri Net approach

Manu De Backer and Monique Snoeck

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 0705

Business Process Verification: a Petri Net Approach

Manu De Backer, Monique Snoeck

K.U.Leuven, Dept. of Applied Economic Sciences,

Naamsestraat 69, B-3000 Leuven, Belgium

{Manu.Debacker; Monique.Snoeck}@econ.kuleuven.be

Abstract

In this report, we discuss the use of Petri Net language theory for business process modeling. Essentially, the focus is on the opportunities of the modeling technique for analysis and verification. Semantic compatibility, as opposed to syntactic compatibility, is concerned with the meaningfulness of the distributed business process. We start with a description and motivation of different notions of semantically compatible business processes. Further, these different types of compatibility are formalized by means of Petri Net language theory. Finally, we describe the foundations of an algorithm that enables us to verify the semantic compatibility in an automated way.

Keywords: Petri Net theory; Business Process Modeling; Verification; Semantic Compatibility

1 Introduction

Offering value-added functionality to customers through the use of distributed business processes is one thing, efficiency and correctness of such a business process is another. When multiple business processes are grouped to create a distributed business process and each of the constituting process in itself cooperates or interacts with other processes this system becomes not only quickly unmanageable but also difficult to analyze and to check its correctness. Therefore, this paper deals with some specific problems that can occur when business processes are interconnected and the ways these problems can be identified and solved.

Semantic compatibility, as opposed to syntactic compatibility, is concerned with the meaningfulness of the distributed business process. Semantic compatibility is concerned with compatibility issues other than implementation details. More specifically, we will assess the potential for creating a successful interaction between two business processes. Later, these results will be shown to be applicable in a broader distributed business process view, i.e. where more than two business processes are integrated. More specifically, we will focus on the following questions:

- Are two business processes semantically compatible?
- When are two business processes semantically compatible?
- If two business processes are semantically incompatible, is it possible to identify the problem area?
- Is it possible to generate a list of the unsupported scenarios?
- How can we automatically verify semantic compatibility?

2 Petri Net Theory

In [1], we have illustrated, by means of a set of business process patterns, how Petri Net language theory can be used for business process modeling. This section repeats the basic definitions of Petri Net theory and introduces the concept of Petri Net language theory. These definitions are used to define the different notions of semantic compatibility.

Definition 1 ***(Petri Net)***
A Petri Net is a triple $N = (P, T, A)$:

- $P = \{p_1, p_2, ..., p_n\}$ *is a finite set of places,* $n \geq 0$,
- $T = \{t_1, t_2, ..., t_m\}$ *is a finite set of transitions,* $m \geq 0$,
- $A \subseteq (P \times T) \cup (T \times P)$ *is the flow relation,*
- $(P \cap T = \emptyset)$*: P and T are disjoint sets.*

Definition 2 ***(Labeled Petri Net)***
A labeled Petri Net PN = (N,τ,μ_0,F) where N = (P,T,A) is a Petri Net, τ:$T \rightarrow \Sigma$ a labeling of T in the alphabet Σ. μ_0 is the initial marking and F is a set of final markings.

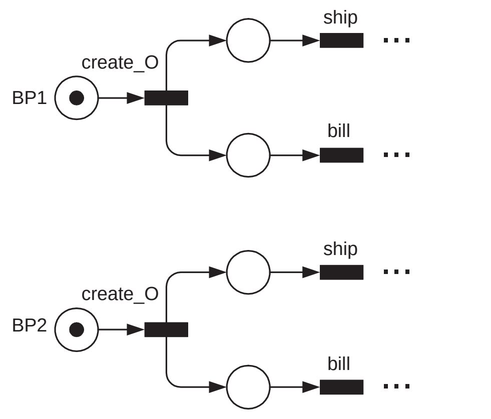
Definition 3 ***(L-type)***
A language L is a L-type *Petri Net language iff there exists a labeled Petri Net PN = (N,τ,μ_0,F) such that L(PN) = $\{\tau(\beta) \in \Sigma^* | \beta \in T^*$ and $\mu_\beta = \delta(\mu_0, \beta)$ and $\mu_\beta \in F\}$. μ_β is the marking reached after firing the sequence β starting from μ_0.*

3 Semantic Compatibility

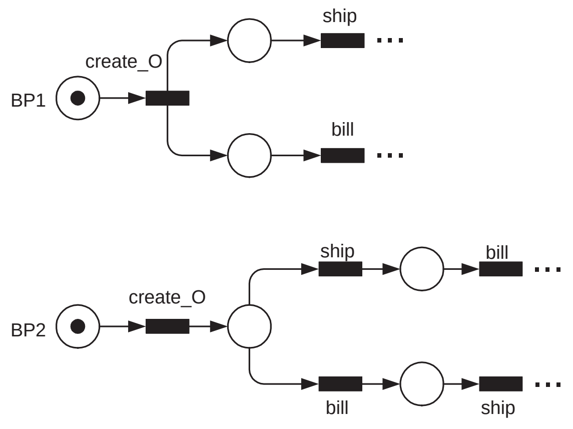
Answering the previously introduced questions on semantic compatibility requires reasoning about the behavioral aspects of the business processes. We have shown (in [1]) that each business process generates a language (L) on the set of business events (the alphabet Σ) in which it participates. More specifically, a language L defines a set of acceptable scenarios of business events over the alphabet. Reasoning about the compatibility of the business processes can then be seen as reasoning about the compatibility of the Petri Net languages. Thus, the Petri Net language generated by the business process can be used to discuss semantic compatibility. In our attempt to find a definition of the semantic compatibility of business processes, it is therefore tempting to say that two business processes are semantically compatible if their Petri Net languages are equivalent. Therefore, the first definition of semantic compatibility requires language equivalence and can be formalized as follows:

Definition 4 ***(Complete Semantic Compatibility*** $\cong$*)*
Let the behavior of two business processes, BP_1 and BP_2, be modeled by two labeled Petri Nets PN_1 and PN_2 respectively. The business processes BP_1 and BP_2 are called complete semantically compatible $BP_1 \cong BP_2$ iff: $L(PN_1)=L(PN_2)$.

Basically, this definition of complete semantic compatibility is correct. Language equivalence implies that the two Petri Nets (business processes) define the same sequence constraints on a set of business events. In this case, the business processes support the same scenarios, and they are therefore able to cooperate in a meaningful way. Clearly, this definition suggests that language equivalence is a sufficient but not a required condition for semantic compatibility. In order to verify complete semantic compatibility we need to discuss the notion of language equivalence. It is interesting to note and clearly illustrated in the next example that morphologic equivalence is not a necessary condition for language equivalence, i.e. two Petri Nets which are not morphologic equivalent can still generate the same language.
Example:



(a) Morphologic and language equivalence.



(b) Not morphologic equivalent but language equivalent.

Figure 1: Difference between language and morphologic equivalence.

Although this definition is definitely usable in this context, we will show that there are in fact two problems that require a weakening of the definition:

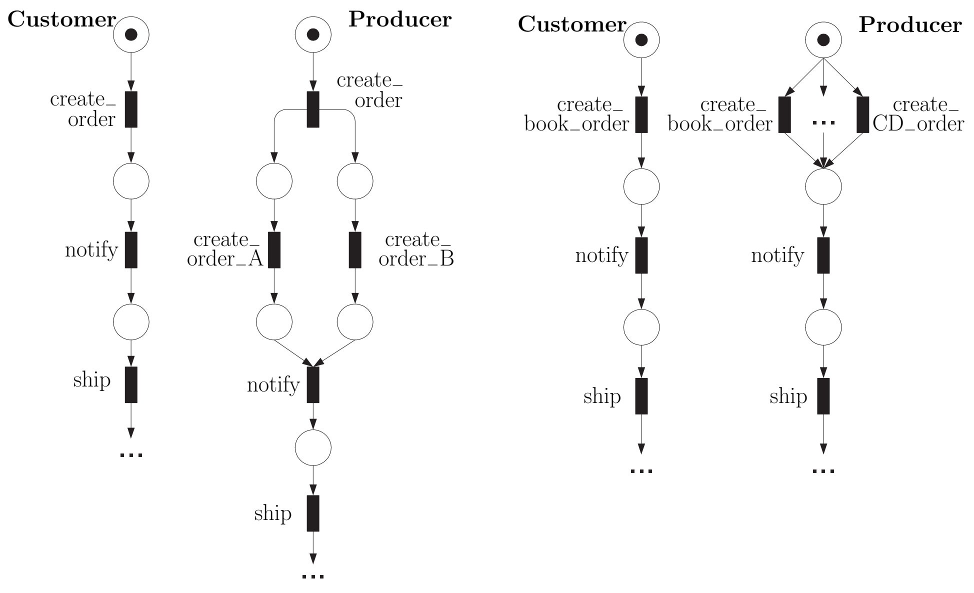
- Different alphabets immediately imply semantic incompatibility;

- Specific characteristics of the interaction require different notions of semantic compatibility.

In the next sections, we will introduce other notions of semantic compatibility to solve these problems.

3.1 Strong Semantic Compatibility

The first identified problem deals with differences in the alphabet. The above definition of semantic compatibility (Definition 4) states that two Petri Nets can never be semantically compatible if their alphabets (Σ) differ. Indeed, if two Petri Nets define a language over a different alphabet, their languages can never be equivalent, thus not compatible. We will show, however, that different alphabets do not automatically imply semantic incompatibility. At least two scenarios can be discussed where Definition 4 is too strict, see Figure 2.



(a) Internal Business Events (b) Redundant Business Events

Figure 2: Reasons for strong semantic compatibility.

First of all, Figure 2(a), shows the existence of internal business events. These business events are irrelevant for the cooperation of the two business processes and thus redundant for the verification. In this case, if we use Definition 4, we conclude that the two business processes are semantically incompatible. Secondly, Figure 2(b) shows a situation where the Producer's business process offers additional functionality which is not necessary for the Customer's business process. The Customer is only interested in ordering books, but the Producer also offers the possibility to order CDs, DVDs, etc. This does however not imply that the business processes are incompatible. Consequently, this problem requires a weaker notion of semantic compatibility that takes into account the differences of the alphabets.

Therefore, we define Strong Semantic Compatibility (SSC) as follows:

Definition 5 ***(Strong Semantic Compatibility*** $\lhd\!\rhd$*)*
Let the behavior of two business processes, BP_1 *and* BP_2*, be modeled by* PN_1 *and* PN_2 *respectively. The business processes* BP_1 *and* BP_2 *are called strong semantically compatible* $BP_1 \lhd\!\rhd BP_2$ *iff:* $L(PN_1|\Sigma_c)=L(PN_2|\Sigma_c)$*. With* Σ_c *being the common alphabet of the two processes:* $\Sigma_c = \Sigma_1 \cap \Sigma_2$*.* $L(PN_\alpha|\Sigma_c)$ *is the language that the business process* α *generates over the common alphabet* Σ_c*.*

Example: If we consider the business processes, BP_{Cus} *and* BP_{Pro} as modeled in Figure 2(a), we know that:

$\Sigma_{Cus} = \{create\_order, notify, ship\}$;

$\Sigma_{Pro} = \{create\_order, create\_order\_A, create\_order\_B, notify, ship\}$.

Then, the common alphabet $\Sigma_C = \Sigma_{Cus} \cap \Sigma_{Pro}$ is :

$\Sigma_c = \{create\_order, notify, ship\}$.

Consequently,

$L(Cus|\Sigma_c) = \{create\_order.notify.ship\}$;

$L(Pro|\Sigma_c) = \{create\_order.notify.ship\}$;

and,

$L(Cus|\Sigma_c) = L(Pro|\Sigma_c)$.

Thus,

$BP_{Cus} \lhd\!\rhd BP_{Pro}$.

3.2 one-way Strong Semantic Compatibility

The second problem we are addressing here stems from the fact that equality of languages is not absolutely required for a meaningful collaboration between partners. More specifically, sometimes, it is sufficient to have one or a few scenarios supported to allow collaboration.

Before we elaborate on this problems we would like to introduce the *initiator-cooperator* relation (IC-relation). The idea behind this relationship, stems from the fact that during the cooperation of two business processes each process fulfills a specific role in the cooperation. A similar relation, but with other semantics, was used in the Business Transaction Protocol [2], e.g. the Superior-Inferior relationship. The IC-relation ascribes to one of the partners the initiator role and to the other the cooperator role. The *initiator* who starts the interaction of the processes and the *cooperator* who cooperates in the process. More specifically, we could say that the initiator uses functionality of another process which is called the cooperator. It is important to see that a single (local) business process can play multiple roles in a distributed business process.

Definition 6 ***(initiator-cooperator relationship*** $\dashrightarrow$***)***
(BP_1,BP_2) $\in \dashrightarrow \Rightarrow$ *BP_1 is the initiator of the cooperation and BP_2 is the cooperator in the interaction.*
(BP_1,BP_2) $\in \dashrightarrow$ *and (BP_2,BP_1)* $\in \dashrightarrow \Rightarrow$ *BP_1 and BP_2 are called* ***peer*** *processes.*

A graphical representation of the initiator-cooperator relationship could be defined in the following way: the initiator is placed above the cooperator and the two are connected by a (dashed) arrow. Peer processes are placed next to each other and are connected by a double sided (dashed) arrow. All the initiator-cooperator relations of a distributed business process can be modeled in an initiator-cooperator graph, see Figure 3. The initiator-cooperator graph should be interpreted in the following way: in the Customer-Producer cooperation, the Customer acts as initiator while the Producer is the cooperator. A more business related interpretation is that the Producer process aids the Customer process in reaching a specific goal. However, in the Producer-Supplier_A relation, the Producer plays the role of initiator and the Supplier_A is the dependent.

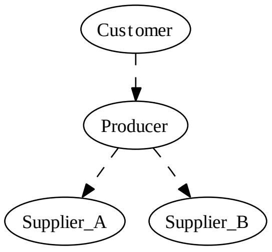


Figure 3: The initiator-cooperator graph of the Customer-Producer Example.

The consequences of this initiator-cooperator relationship on the definition of semantic compatibility are fundamental. Another notion of semantic compatibility should be introduced and defined. We will call two business processes *one-way strong semantically compatible* if the scenarios of the initiator process are entirely supported by the process of the cooperator. If all scenarios of the initiator are supported then a meaningful interaction between the two partners is possible. An example can demonstrate the fundamental character of one-way strong semantic compatibility more efficiently.

Example: This example demonstrates the notion of one-way strong semantic compatibility by means the business process modeled in Figure 5. From the initiator-cooperator graph, cf. Figure 3, we can conclude that the Producer acts as the initiator of the cooperation while Supplier_A is the cooperator of the process. In this example it is clear that all scenarios of the Producer process are supported by Supplier_A's process. Supplier_A however, does support some additional scenarios. We will call two processes one-way strong semantic compatible, if all the scenarios of the initiator process are supported by the cooperator process but the cooperator process supports additional scenarios. Considering this problem we can define another notion of semantic compatibility as follows:

Definition 7 ***(one-way Strong Semantic Compatibility*** $\triangleright$*)*
Let the behavior of two business processes, BP_1 and BP_2, be modeled by PN_1 and PN_2 respectively. Given that (BP_1,BP_2) $\in \dashrightarrow$ then the business processes BP_1 and BP_2 are called one-way strong semantically compatible $BP_1 \triangleright BP_2$ iff: $\forall \alpha \in L(PN_1) : \alpha|\Sigma_c \in L(PN_2|\Sigma_c)$. In Petri Net language theory one-way strong semantic compatibility can be verified through: $L(PN_1|\Sigma_c) \subset L(PN_2|\Sigma_c)$.

3.3 Weak Semantic Compatibility

Additionally, we would like to introduce the weakest notion of semantic compatibility, e.g. weak semantic compatibility. Two business processes are weak semantically compatible if there is at least one common scenario. Weak semantic compatibility can be defined as follows:

Definition 8 *(**Weak Semantic Compatibility** $\bowtie$)*
Let the behavior of two business processes, BP_1 and BP_2, be modeled by PN_1 and PN_2 respectively. The business processes BP_1 and BP_2 are called weak semantically compatible $BP_1 \bowtie BP_2$ iff: $\exists \alpha \in L(PN_1|\Sigma_c) : \alpha \in L(PN_2|\Sigma_c)$.

3.4 Summary of Semantic Compatibility

Name	**Symbol**	**PN-language**
Complete Semantic Compatibility (CSC)	$\cong$	$L(PN_1) = L(PN_2)$
Strong Semantic Compatibility (SSC)	$\lhd\!\rhd$	$L(PN_1\|\Sigma_c) = L(PN_2\|\Sigma_c)$
one-way SSC (owSSC)	$\rhd$	$L(PN_1\|\Sigma_c) \subset L(PN_2\|\Sigma_c)$
Weak Semantic Compatibility (WSC)	$\bowtie$	$L(PN_1\|\Sigma_c) \cap L(PN_2\|\Sigma_c) \neq \emptyset$

Table 1: Overview of the different semantic compatibility definitions.

Table 1 gives an overview of the different definitions of semantic compatibility that are used in this dissertation. The third column shows how we can verify these semantic compatibilities by means of Petri Net languages, which we will discuss in the following sections. The definitions of semantic compatibility are logically related, i.e. complete semantic compatibility automatically implies strong semantic compatibility etc. We can identify the following relationships:

$$CSC \Longrightarrow SSC \Longrightarrow owSSC \Longrightarrow WSC \tag{1}$$

4 Semantic Compatibility in an Algorithm

In this section, we will show that semantic compatibility between two business processes can be verified by means of Petri Net language theory. Checking semantic compatibility is a complex process consisting of different important steps. In this section we try to summarize on the results that have been presented in previous sections by discussing it from a different angle.

The complete analysis process is depicted in Figure 4. The analysis process starts with modeling the business processes as deterministic Petri Net languages. Next, the processes are checked for alphabet differences. If the alphabets are equivalent the processes are checked for complete semantic compatibility, otherwise language projection is performed and the analysis continues by checking strong semantic compatibility. Either way if the result is false for the complete or strong semantic compatibility the complement of the process with the least complexity is computed and the IC-graph is used to compute the one-way strong semantic compatibility. Finally, weak semantic compatibility is checked. In some cases bisimilarity based reduction (BBR) can be an interesting strategy to lower the complexity of the compared Petri Net languages. BBR is a straightforward technique of reducing the complexity by comparing two Petri Net languages and based on this comparison it is sometimes interesting (in case they are equivalent) to remove some ending parts of the languages. Essentially, this means that we remove equivalent parts of the Petri Nets (if there are any) to lower the future computation times of the algorithms. For each of the steps, i.e. compatibility checks, an analysis report is generated with a summary of the detected problems. Some of the issues added in the report are the difference in the alphabet, the set of unsupported scenarios, and the computation times of the algorithms.

The analysis technique is implemented in a prototype environment called the Business Process Analyzer. In future work we will assess the techniques and the implementation on realistically sized problems. Additionally, some optimizations, such as BBR, should be implemented to further improve the performance of the tool.

In the next section we show how these different types of semantic compatibility can be applied to a realistic example.

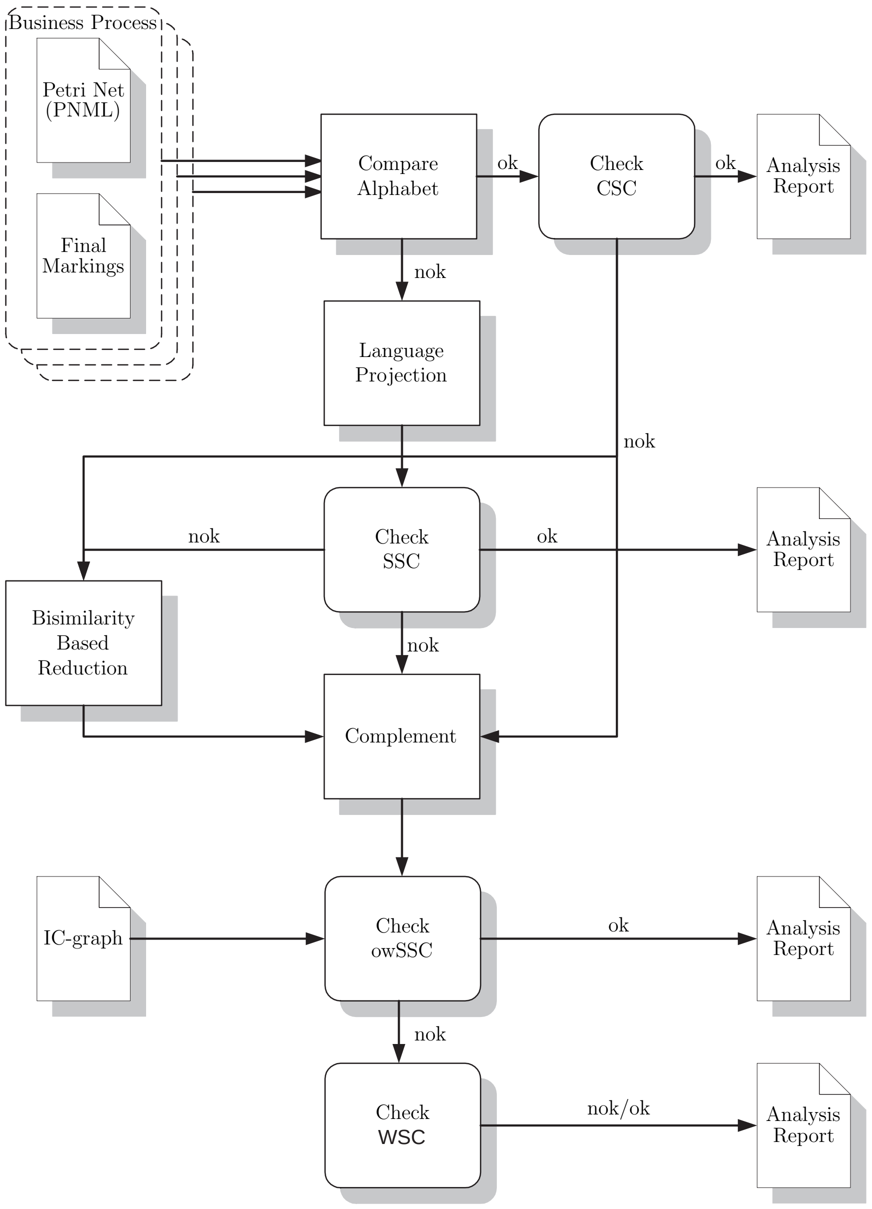


Figure 4: The analysis process.

5 Case-Study

In this section, we use a realistic business process model to show the concept of semantic compatibility. The proposed technique can be applied in one of the following cases: the business process is initially modeled using the business events and Petri Net or the business process is modeled using a business process modeling language (BPMN, BPEL). Ideally, business process modeling starts with an abstract representation of the business process and in the following phases details are added and finally, an implementation ready version is achieved.

In this example we assume that the initial process was modeled by means of BPMN (see Figure 5). This model cannot be used immediately for verification. Therefore, we need to translate the model to the proposed technique by identifying the business events and by transforming the control flow to Petri Net semantics. The approach to achieve this is not discussed in detail, but the basic idea is that whenever communication between two process is modeled, the business event represents the communication without explicitly modeling the message exchanges. Figure 6 shows the business event and Petri Net based representation of the business process. Next, the initiator-cooperator graph of this distributed business process is given in Figure 7.

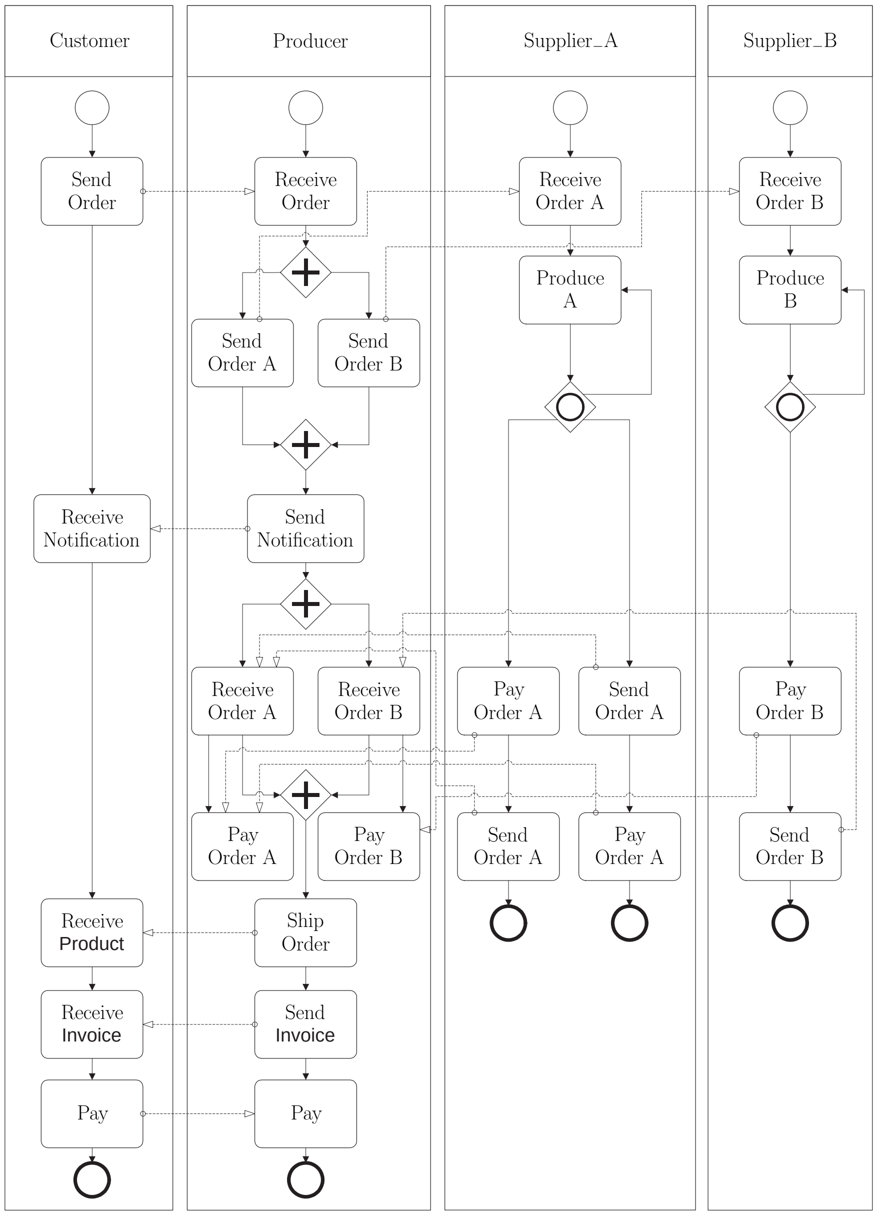


Figure 5: The Business Process Diagram.

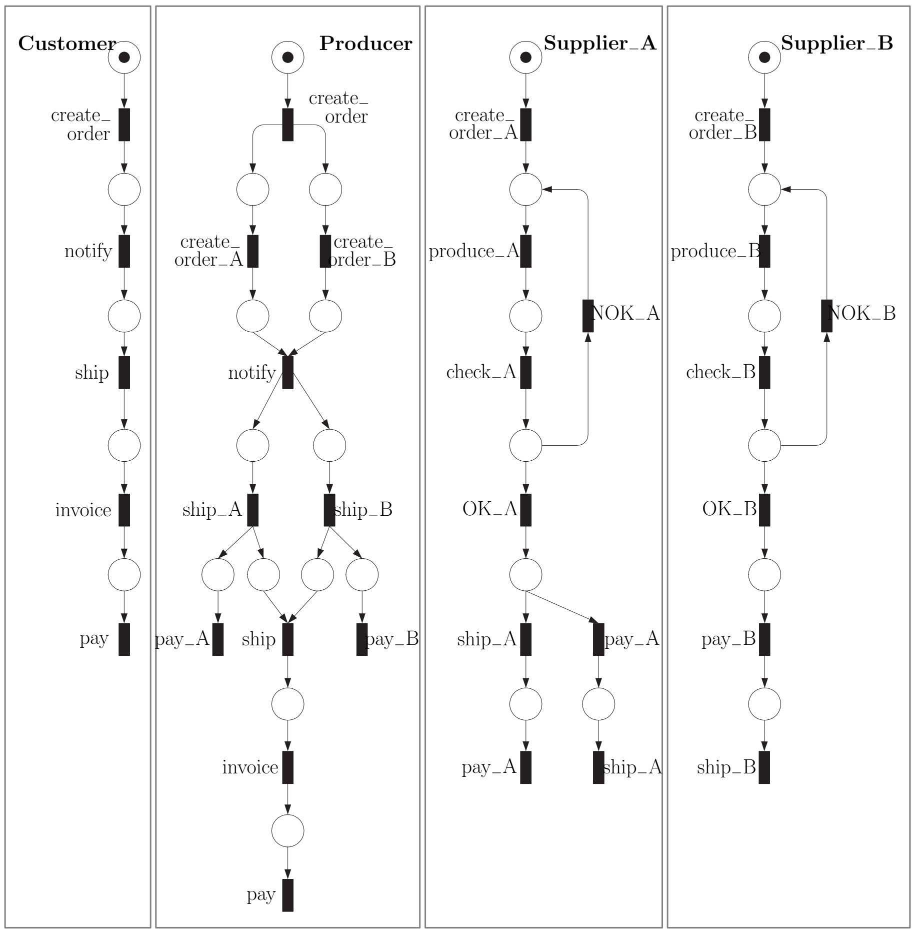


Figure 6: Business Event based business process modeling.

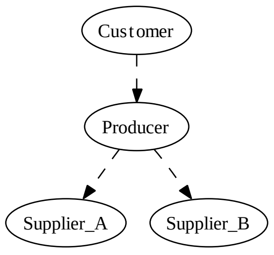


Figure 7: The initiator-cooperator graph of the Customer-Producer Example.

In order to complete this business process specification we need to define each of the Petri Net languages as follows:

- $L(Customer)$ with $\mu_0 = (1, 0, 0, 0, 0)$ and F=$(0, 0, 0, 0, 0)$;
- $L(Producer)$ with $\mu_0 = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ and F=$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$;
- $L(Supplier\_A)$ with $\mu_0 = (1, 0, 0, 0, 0, 0, 0)$ and F=$(0, 0, 0, 0, 0, 0, 0)$;
- $L(Supplier\_B)$ with $\mu_0 = (1, 0, 0, 0, 0, 0)$ and F=$(0, 0, 0, 0, 0, 0)$;

For each of the relations defined in the initiator-cooperator graph we need to verify semantic compatibility. The verification process was discussed in Figure 4 (see pg. 11). The process starts by checking the alphabets, in case of differences complete semantic compatibility is not applicable. Next, language projection is performed to check strong semantic compatibility. If the processes are not strong semantically compatible the complement of the cooperator process is computed and one-way strong semantic compatibility is checked. Finally, the processes can be weak semantically compatible. If none of the previous types of semantic compatibility is correct, the processes are completely incompatible, i.e. there is not one scenario that is supported by the two business processes.

Customer-Producer Clearly, the Customer and Producer business processes are by no means complete semantically compatible (see Definition 4). From the analysis process (discussed in Figure 4) we know that the next phase is language projection in order to eliminate irrelevant activities.

$$\Sigma_{Cus} = \{cr\_order, notify, ship, invoice, pay\};$$

$$\Sigma_{Pro} = \{cr\_order, cr\_order\_A, cr\_order\_B, notify, ship\_A, ship\_B, pay\_A, pay\_B, ship, invoice, pay\}.$$

Then, the common alphabet $\Sigma_C = \Sigma_{Cus} \cap \Sigma_{Pro}$ is :

$$\Sigma_c = \{create\_order, notify, ship, invoice, pay\}.$$

Consequently,

$$L(Cus|\Sigma_c) = \{cr\_order.notify.ship.invoice.pay\};$$

$$L(Pro|\Sigma_c) = \{cr\_order.notify.ship.invoice.pay\};$$

and,

$$L(Cus|\Sigma_c) = L(Pro|\Sigma_c).$$

Thus,

$$BP_{Cus} \lhd\rhd BP_{Pro}.$$

The customer and producer business process are strong semantically compatible, and every scenario is supported by the two process.

Producer-Supplier_A For the Producer-Supplier_A we need to apply the same procedure. Of course, they are not complete semantically compatible. Next, language projection is performed.

$$\Sigma_{Pro} = \{cr\_order, cr\_order\_A, cr\_order\_B, notify, ship\_A,$$
$$ship\_B, pay\_A, pay\_B, ship, invoice, pay\}.$$
$$\Sigma_{SupA} = \{cr\_order\_A, produce\_A, check\_A, OK\_A, NOK\_A,$$
$$ship\_A, pay\_A\}.$$

Then, the common alphabet $\Sigma_C = \Sigma_{Pro} \cap \Sigma_{SupA}$ is :

$$\Sigma_c = \{cr\_order\_A, ship\_A, pay\_A\}.$$

Consequently,

$$L(Pro|\Sigma_c) = \{cr\_order\_A.ship\_A.pay\_A\};$$
$$L(SupA|\Sigma_c) = \{cr\_order\_A.(ship\_A.pay\_A + pay\_A.ship\_A)\};$$

and,

$$L(Pro|\Sigma_c) \neq L(SupA|\Sigma_c) \ but \ L(Pro|\Sigma_c) \subset L(SupA|\Sigma_c)$$

Thus,

$$BP_{Pro} \rhd BP_{SupA}.$$

The producer and the supplier_A process are one-way semantically compatible, i.e. all the scenarios of the initiator process (Producer) are supported by the cooperator (Supplier_A) process.

Producer-Supplier_B

$\Sigma_{Pro} = \{cr\_order, cr\_order\_A, cr\_order\_B, notify, ship\_A,$

$ship\_B, pay\_A, pay\_B, ship, invoice, pay\}.$

$\Sigma_{SupB} = \{cr\_order\_B, produce\_B, check\_B, OK\_B, NOK\_B,$

$ship\_B, pay\_B\}.$

Then, the common alphabet $\Sigma_C = \Sigma_{Pro} \cap \Sigma_{SupB}$ is :

$\Sigma_c = \{cr\_order\_B, ship\_B, pay\_B\}.$

Consequently,

$L(Pro|\Sigma_c) = \{cr\_order\_B.ship\_B.pay\_B\};$

$L(SupB|\Sigma_c) = \{cr\_order\_B.pay\_B.ship\_B\};$

and,

$L(Pro|\Sigma_c) \neq L(SupB|\Sigma_c)$ *and* $L(Pro|\Sigma_c) \cap L(SupB|\Sigma_c) = \varnothing$

Thus,

BP_{Pro} *is by no means compatible with* BP_{SupB}.

The producer business process is not compatible with the process of supplier_B, i.e. this inconsistency needs to be solved before the implementation of the business process.

6 Conclusion

In this report, the focus was on the compatibility of business processes, more specifically, we introduced the notion of semantically compatible business processes. Semantic compatibility was described as a criterion to assess the meaningfulness of the interaction of distributed business processes. Next, we have motivated that a single definition of semantic compatibility is out of the question, and that there are multiple problems that require other notions of semantic compatibility. Therefore, we defined, in this report, four different notions of semantic compatible business processes, e.g. complete semantic compatibility, strong semantic compatibility, one-way strong semantic compatibility and weak semantic compatibility. Additionally, we have defined these notions by means of Petri Net language theory. Further, we discussed the basis of a verification technique which enables us to check semantic compatibility in an automated way.

References

[1] M. De Backer and M. Snoeck. Deterministic petri net languages as business process specification language. Dtew research report 0577, K.U.Leuven, 2005.

[2] A. Ceponkus, S. Dalal, T. Fletcher, P. Furniss, A. Green, and B. Pope. Business transaction protocol, v1.0. Technical report, OASIS, June 2002.

[3] P. Jančar. Undecidability of bisimilarity for petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.