SOCA (2009) 3:189–204
DOI 10.1007/s11761-009-0048-5

ORIGINAL RESEARCH PAPER

Equivalences of BPMN processes

Vitus S. W. Lam

Received: 24 March 2009 / Revised: 24 June 2009 / Accepted: 23 July 2009 / Published online: 13 August 2009

Abstract Notwithstanding the business process modelling notation (BPMN) has gained increasing popularity in the context of business process management; a foundation for verifying the equivalences of BPMN processes remains an open research problem. The capacity to prove that two structurally different graphical representations of a business process are behaviourally equivalent using mathematical founded methods is important during the modelling and design of workflows. In this study, various forms of equivalences are formally defined and studied.

Keywords BPMN · Process equivalence · Equivalence classification

1 Introduction

Business process modelling notation (BPMN) [1,2], which is a graphical modelling language, has established itself as the lingua franca for documenting, visualizing, specifying and designing business processes [3]. BPMN developed initially by business process management initiative is currently maintained by the Object Management Group. When compared with BPMN 1.0 [1], major modifications in BPMN 1.1 [2] encompass

(i) the classification of events into catching events and throwing events; and
(ii) the introduction of a new construct signal event for broadcasting a signal to other processes and business process diagrams.

V. S. W. Lam (✉)
Computer Centre, The University of Hong Kong,
Pokfulam Road, Hong Kong, Hong Kong
e-mail: vitus.lam@ieee.org

As BPMN becomes more prevalent in the modelling of business processes, establishing a principled method to determine when two BPMN processes are behaviourally equivalent is crucial in the course of process design. However, little research is done for advancing our understanding on this fundamental issue. This work is concerned with how mathematical techniques are adopted as a solid basis for the equivalence checking of BPMN processes.

The remainder of this paper is structured as follows. Section 2 concentrates on a critical review of the literature. Section 3 is devoted to a summary of the graphical constructs of BPMN. A formalization of BPMN processes is the primary focus of Sect. 4. Section 5 presents a formal description of various forms of equivalences with regard to BPMN processes. The practicality of the proposed equivalences is demonstrated by means of an elaborated example. Section 6 summarizes the major results of our endeavour and outlines directions for further research.

2 Related work

Numerous studies in the literature deal with the simulation, analysis and verification of business processes expressed as BPMN 1.0. Bog et al. [4–7] convert the business process diagrams of BPMN into the π-calculus. The π-calculus representations are then simulated and analysed using PiVizTool [4–7]. In the same spirit, Puhlmann [8] encodes BPMN models in the π-calculus and utilizes Advanced Bisimulation Checker (ABC) [9] for verifying the correctness of the π-calculus specifications. Ou-Yang and Lin [10] use BPEL4WS as intermediate representations when transforming BPMN models into Colored Petri-net XML (CPNXML). The CPNXML representations are verified against various properties using CPN Tools [11]. Raedts et al. [12] define semantic mappings between BPMN models and Petri nets as well as between Petri nets and mCRL2 in order to determine the validity of BPMN models. The work of Dijkman et al. [13, 14] formalizes BPMN models in the form of Petri nets for analyzing with ProM framework. Wong and Gibbons [15] adopt communicating sequential processes (CSP) [16] as the semantic domain for a subset of BPMN graphical constructs. The failure-divergence refinement (FDR) model checker [17] is employed for examining the compatibility between BPMN processes. Although a fairly large body of literature exists on BPMN, there are marked discrepancies between these prior contributions and our work as:

(i) These earlier studies are primarily focused on the simulation, analysis and verification of BPMN rather than BPMN process equivalences. Specifically, the subject of our work is on establishing a taxonomy of equivalences for BPMN processes. Formal description of a theory of equivalences related to BPMN processes is presented.
(ii) The foundational work for the proposed equivalences is based on BPMN 1.1 in lieu of 1.0. BPMN 1.1 is distinct from BPMN 1.0 in several respects:
    (a) the use of catching event and throwing event to differentiate between the two different notions;
    (b) the incorporation of signal event into the BPMN 1.1 specification; and
    (c) the modification of the representations for graphical constructs including multiple instance marker as well as multiple event.

Other closely related studies encompass [18–20]. Gruber [18] and Eder et al. [19] explore the semantically equivalent transformations of workflows. Despite both Gruber et al. [18, 19] and our work are based on semantics-preserving transformations, substantial differences lie in two fundamental aspects:

(i) The emphasis of our work is on the equivalences of BPMN processes, whereas the studies of Gruber et al. [18, 19] deal with the equivalences of structured workflow graphs. Unlike structured workflow graphs, BPMN supports a wider range of notational elements such as start events, end events, intermediate events, transactions, complex decision gateways, message flows, associations, data objects, groups, text annotations, pools and lanes.
(ii) Contrary to Gruber et al., some BPMN specific equivalences are propounded in our work. These equivalences build upon BPMN graphical constructs including none start events, complex decision gateways and conditional sequence flows.

Our earlier attempt [20] exploits a disciplined approach for the classification of different kinds of equivalences for UML activity diagrams. Nevertheless, research concentrating on the equivalences of BPMN processes defined in terms of BPMN 1.1 specification is given little attention. The aim of this work is to address this neglected area by studying the various forms of equivalences along with their properties.

3 An overview of BPMN

In BPMN, a process consists of graphical constructs that are broken down into four categories: flow objects, connecting objects, swimlanes and artifacts. The flow objects and connecting objects are notational elements for specifying the behaviour of a process. The swimlanes are graphical constructs for separating and organizing the graphical elements of a process. The artifacts are dedicated to the specification of extra information which does not associate with both the sequence and message flows of a process [2].

As shown in Figs. 1 and 2, the three types of flow objects are: events, activities and gateways. A start event triggers the commencement of the flow of a process by generating a token. An end event consumes the token and represents the end of the flow of a process. A none start event means that the event type is not specified or indicates the start of a subprocess when the flow is moved from its parent process to it. A message start event, timer start event, conditional start event and signal start event are triggered, respectively, upon receipt of a message, when a particular time date is reached, whenever a condition holds and on receiving a signal. A multiple start event signifying more than one triggers is capable of starting a process.

A none end event represents the event type is not defined or the completion of a subprocess in which the flow is transferred from it to its parent process. A message end event, an error end event, a cancel end event, a compensation end event, a signal end event and a terminate end event symbolize the sending of a message to a participant, the generation of a named error, the cancellation of a transaction, the performing of a compensation, the broadcasting of a signal and the immediate termination of all flows in a process, respectively. A multiple end event indicates more than one results occur when a process ends.

A none intermediate event is used when the event type is not specified. A message intermediate event, a timer intermediate event, an error intermediate event, a cancel intermediate event, a compensation intermediate event, a conditional intermediate event, a link intermediate event and a signal intermediate event signify, respectively, the receipt of a message from a participant, the reaching of a particular time date, the catching of a named error, the receipt of a cancel message, the catching of a compensation event, the holding of

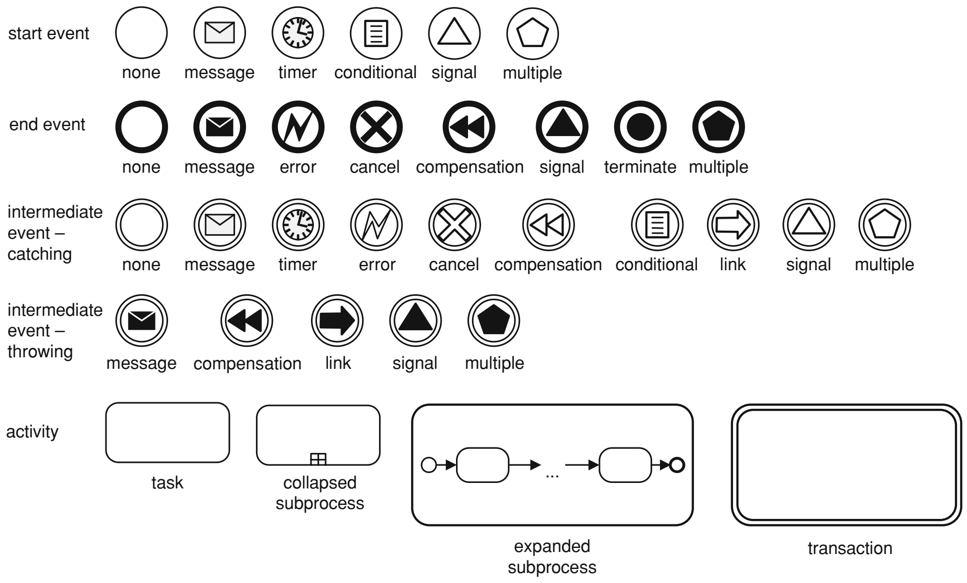


Fig. 1 BPMN notational elements

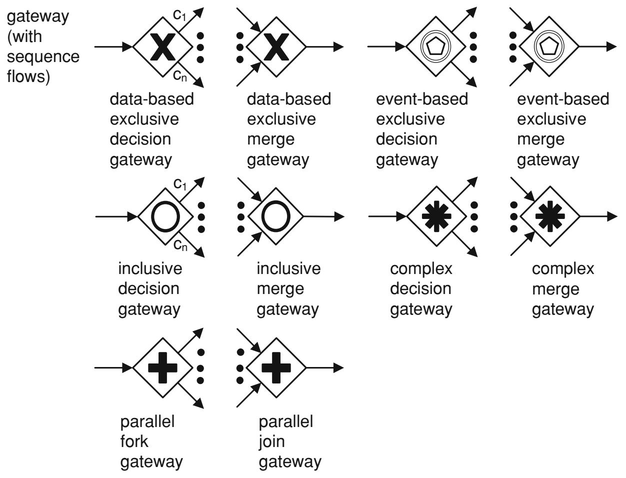

gateway (with sequence flows)
c1
cn
data-based exclusive decision gateway
data-based exclusive merge gateway
event-based exclusive decision gateway
event-based exclusive merge gateway
inclusive decision gateway
inclusive merge gateway
complex decision gateway
complex merge gateway
parallel fork gateway
parallel join gateway

Fig. 2 BPMN notational elements (continued)

a condition, the catching from a source link and the receipt of a signal when the intermediate events are used for catching event triggers. A multiple intermediate event, which is used for catching an event trigger, represents more than one triggers can resume a process.

A message intermediate event, a compensation intermediate event, a link intermediate event and a signal intermediate event indicate the sending of a message to a participant, the throwing of a compensation event, the throwing to a target link and the sending of a signal if the intermediate events are used for throwing event triggers. A multiple intermediate event, which is used for throwing an event trigger, symbolizes more than one triggers are thrown.

There are two types of activities: tasks and subprocesses. A task is an activity that is not decomposable and does not contain any other activities. In contrast, a subprocess is an activity that is decomposable. It has a lower level of detail and consists of a set of other activities. A subprocess is either represented as a collapsed view or an expanded view. In a collapsed subprocess, all the details are hidden and only a plus sign is shown. On the contrary, all the fine details are visible in an expanded subprocess. A transaction is a special kind of subprocess in which all its activities are regarded as an atomic unit that is either complete or cancel. All the activities within a transaction are reverted, compensation activities are executed and a cancellation handler is performed upon receipt of a cancellation event.

A data-based exclusive decision gateway is composed of one incoming sequence flow and two or more mutually exclusive outgoing sequence flows. Each outgoing sequence flow is associated with a corresponding conditional expression evaluated merely at runtime. In accordance to which conditional expression evaluates to true, a token is sent on one of the mutually exclusive outgoing sequence flows whenever a token is received from the incoming sequence flow. A data-based exclusive merge gateway comprises two or more mutually exclusive incoming sequence flows and one outgoing sequence flow. Upon receipt of a token along one of the incoming sequence flows, a token is offered to the outgoing sequence flow.

An event-based exclusive decision gateway connects the outgoing sequence flows to events. A token is sent on one of the mutually exclusive sequence flows according to the first received event. An event-based exclusive merge gateway, like a data-based exclusive merge gateway, emits a token along the outgoing sequence flow on receiving a token from one of the incoming sequence flows.

In contrast to a data-based exclusive decision gateway, an inclusive decision gateway sends a token on all outgoing sequence flows which conditional expressions evaluate

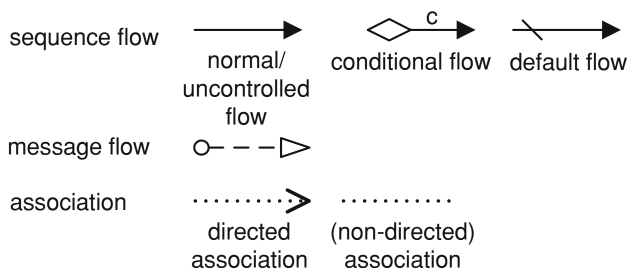


Fig. 3 BPMN notational elements (continued)

to true. Every outgoing sequence flow is completely independent of any other outgoing sequence flows in lieu of mutually exclusive. An inclusive merge gateway waits until all the tokens generated by a corresponding inclusive decision gateway are received before the traverse of a token along the outgoing sequence flow.

A complex decision gateway, which is based on an expression, determines which one of the possible sets of outgoing sequence flows is selected. A complex merge gateway makes use of an expression for deciding which one of the possible combinations of incoming sequence flows is needed in order to continue the process flow.

A parallel fork gateway receives a token along the incoming sequence flow, splits the token apart and sends them on all outgoing sequence flows for modelling concurrent flows. A parallel join gateway blocks until a token is received from each incoming sequence flow, merges the token together and emits it along the outgoing sequence flow.

The three kinds of connecting objects are: sequence flows, message flows and associations (Fig. 3). A sequence flow links up a source flow object and a target flow object. A token is generated by the source flow object at the end of its execution. Then the token traverses the sequence flow and arrives at the target flow object eventually. A normal flow is a flow that commences at a start event, passes through a number of gateways and terminates at an end event. Unlike a normal flow, an uncontrolled flow is a flow that does not pass over any gateways. A conditional flow, which is a sequence flow, is associated with a conditional expression. The mini-diamond marker is only shown when the source flow object is an activity instead of a gateway. A default flow is fired whenever all the conditional expressions of the other outgoing conditional flows evaluate to false.

A message flow captures the message exchanged between two participants that take part in an interaction. The source and target objects of a message flow are limited to the following combinations [21]: (i) two different pools; (ii) a pool and a flow object of another pool; and (iii) the flow objects of two different pools.

In BPMN, there are three pre-defined artifacts: data objects, groups and text annotations (Fig. 4). A data object

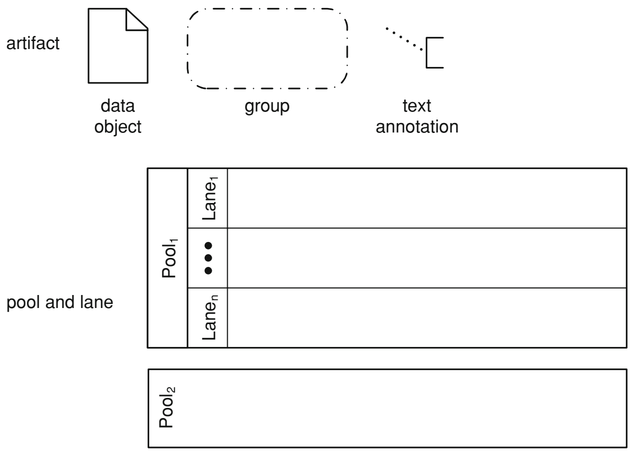


Fig. 4 BPMN notational elements (continued)

defines what the input or output of an activity is. A group is a graphical construct for grouping the notational elements of a BPMN diagram as a category. A text annotation gives extra information to the reader in the form of text description.

An association connects an artifact with a flow object (Fig. 3). A directed association indicates whether a data object is considered as the input or output of an activity. An association, which is non-directional, is utilized for attaching a text annotation with a flow object.

A pool stands for a participant. It serves as the container of a process. Every pool comprises at minimal one lane. Each lane has a distinct name as delineated in Fig. 4. On the condition that a pool contains just one lane, the lane name is identical to the pool name.

4 Formal model of BPMN processes

The formal rigour, which is essential for studying a variety of equivalences of BPMN processes, is absent from the BPMN 1.1 specification. This section seeks to provide a mathematical model for BPMN processes which lays the foundational work for categorizing equivalences of BPMN processes. The proposed model considerably extends our previous one introduced in [21] in two perspectives. First, the BPMN 1.1 specification, which is the latest version, is adopted as the basis for developing the proposed formal model in lieu of the BPMN 1.0 specification. Second, the whole set of notational elements of flow objects, the sequence flows, message flows and directed associations of connecting objects as well as the data objects of artifacts are covered in the current model instead of confining to a subset of the graphical constructs. We do not consider non-directed associations, groups and text annotations as they are notational elements which are not concerned with the behavioural aspect of BPMN processes.

Definition 1 (*Start-event tuple*) A start-event tuple is a 6-tuple $\Omega_{\mathrm{SE}} = \left(F_{\mathrm{SE}}^{\mathrm{None}}, F_{\mathrm{SE}}^{\mathrm{Msg}}, F_{\mathrm{SE}}^{\mathrm{Timer}}, F_{\mathrm{SE}}^{\mathrm{Cond}}, F_{\mathrm{SE}}^{\mathrm{Sign}}, F_{\mathrm{SE}}^{\mathrm{Multi}}\right)$ where

- $F_{\mathrm{SE}}^{\mathrm{None}}$ is a set of none start events for catching the event triggers;
- $F_{\mathrm{SE}}^{\mathrm{Msg}}$ is a set of message start events for catching the event triggers;
- $F_{\mathrm{SE}}^{\mathrm{Timer}}$ is a set of timer start events for catching the event triggers;
- $F_{\mathrm{SE}}^{\mathrm{Cond}}$ is a set of conditional start events for catching the event triggers;
- $F_{\mathrm{SE}}^{\mathrm{Sign}}$ is a set of signal start events for catching the event triggers; and
- $F_{\mathrm{SE}}^{\mathrm{Multi}}$ is a set of multiple start events for catching the event triggers.

A start-event tuple contains six kinds of start events comprising none, message, timer, conditional, signal and multiple.

Definition 2 (*Intermediate-event tuple*) An intermediate-event tuple is a 15-tuple $\Omega_{\mathrm{IE}} = \left(F_{\mathrm{IE}}^{\mathrm{None}}, F_{\mathrm{IE}}^{\mathrm{Msg}}, F_{\mathrm{IE}}^{\overline{\mathrm{Msg}}}, F_{\mathrm{IE}}^{\mathrm{Timer}}, F_{\mathrm{IE}}^{\mathrm{Err}}, F_{\mathrm{IE}}^{\mathrm{Cncl}}, F_{\mathrm{IE}}^{\mathrm{Cmpen}}, F_{\mathrm{IE}}^{\overline{\mathrm{Cmpen}}}, F_{\mathrm{IE}}^{\mathrm{Cond}}, F_{\mathrm{IE}}^{\mathrm{Link}}, F_{\mathrm{IE}}^{\overline{\mathrm{Link}}}, F_{\mathrm{IE}}^{\mathrm{Sign}}, F_{\mathrm{IE}}^{\overline{\mathrm{Sign}}}, F_{\mathrm{IE}}^{\mathrm{Multi}}, F_{\mathrm{IE}}^{\overline{\mathrm{Multi}}}\right)$ where

- $F_{\mathrm{IE}}^{\mathrm{None}}$ is a set of none intermediate events for catching the event triggers;
- $F_{\mathrm{IE}}^{\mathrm{Msg}}$ is a set of message intermediate events for catching the event triggers;
- $F_{\mathrm{IE}}^{\overline{\mathrm{Msg}}}$ is a set of message intermediate events for throwing the event triggers;
- $F_{\mathrm{IE}}^{\mathrm{Timer}}$ is a set of timer intermediate events for catching the event triggers;
- $F_{\mathrm{IE}}^{\mathrm{Err}}$ is a set of error intermediate events for catching the event triggers;
- $F_{\mathrm{IE}}^{\mathrm{Cncl}}$ is a set of cancel intermediate events for catching the event triggers;
- $F_{\mathrm{IE}}^{\mathrm{Cmpen}}$ is a set of compensation intermediate events for catching the event triggers;
- $F_{\mathrm{IE}}^{\overline{\mathrm{Cmpen}}}$ is a set of compensation intermediate events for throwing the event triggers;
- $F_{\mathrm{IE}}^{\mathrm{Cond}}$ is a set of conditional intermediate events for catching the event triggers;
- $F_{\mathrm{IE}}^{\mathrm{Link}}$ is a set of link intermediate events for catching the event triggers;
- $F_{\mathrm{IE}}^{\overline{\mathrm{Link}}}$ is a set of link intermediate events for throwing the event triggers;
- $F_{\mathrm{IE}}^{\mathrm{Sign}}$ is a set of signal intermediate events for catching the event triggers;
- $F_{\mathrm{IE}}^{\overline{\mathrm{Sign}}}$ is a set of signal intermediate events for throwing the event triggers;
- $F_{\mathrm{IE}}^{\mathrm{Multi}}$ is a set of multiple intermediate events for catching the event triggers; and
- $F_{\mathrm{IE}}^{\overline{\mathrm{Multi}}}$ is a set of multiple intermediate events for throwing the event triggers.

In BPMN, there are ten kinds of intermediate events for catching the event triggers and five types of intermediate events for throwing the event triggers. The notion is captured formally in form of an intermediate-event tuple.

Definition 3 (*End-event tuple*) A end-event tuple is a 8-tuple $\Omega_{\mathrm{EE}} = \left(F_{\mathrm{EE}}^{\overline{\mathrm{None}}}, F_{\mathrm{EE}}^{\overline{\mathrm{Msg}}}, F_{\mathrm{EE}}^{\overline{\mathrm{Err}}}, F_{\mathrm{EE}}^{\overline{\mathrm{Cncl}}}, F_{\mathrm{EE}}^{\overline{\mathrm{Cmpen}}}, F_{\mathrm{EE}}^{\overline{\mathrm{Sign}}}, F_{\mathrm{EE}}^{\overline{\mathrm{Term}}}, F_{\mathrm{EE}}^{\overline{\mathrm{Multi}}}\right)$ where

- $F_{\mathrm{EE}}^{\overline{\mathrm{None}}}$ is a set of none end events for throwing the event triggers;
- $F_{\mathrm{EE}}^{\overline{\mathrm{Msg}}}$ is a set of message end events for throwing the event triggers;
- $F_{\mathrm{EE}}^{\overline{\mathrm{Err}}}$ is a set of error end events for throwing the event triggers;
- $F_{\mathrm{EE}}^{\overline{\mathrm{Cncl}}}$ is a set of cancel end events for throwing the event triggers;
- $F_{\mathrm{EE}}^{\overline{\mathrm{Cmpen}}}$ is a set of compensation end events for throwing the event triggers;
- $F_{\mathrm{EE}}^{\overline{\mathrm{Sign}}}$ is a set of signal end events for throwing the event triggers;
- $F_{\mathrm{EE}}^{\overline{\mathrm{Term}}}$ is a set of terminate end events for throwing the event triggers; and
- $F_{\mathrm{EE}}^{\overline{\mathrm{Multi}}}$ is a set of multiple end events for throwing the event triggers.

An end-event tuple divides end events into the following categories: none, message, error, cancel, compensation, signal, terminate and multiple.

Definition 4 (*Event tuple*) Suppose Γ_{SE} = {None, Msg, Timer, Cond, Sign, Multi}, Γ_{EE} = {None, Msg, Err, Cncl, Cmpen, Sign, Term, Multi}, Γ_{IE} = {None, Msg, Timer, Err, Cncl, Cmpen, Cond, Link, Sign, Multi}, $\Gamma_{\overline{\mathrm{IE}}}$ = {$\overline{\mathrm{Msg}}$, $\overline{\mathrm{Cmpen}}$, $\overline{\mathrm{Link}}$, $\overline{\mathrm{Sign}}$, $\overline{\mathrm{Multi}}$}, $F_{\mathrm{SE}} = \bigcup_{i\in\Gamma_{\mathrm{SE}}} F_{\mathrm{SE}}^{i}$, $F_{\mathrm{EE}} = \bigcup_{i\in\Gamma_{\mathrm{EE}}} F_{\mathrm{EE}}^{\bar{i}}$, $F_{\mathrm{IE}} = \bigcup_{i\in(\Gamma_{\mathrm{IE}}\cup\Gamma_{\overline{\mathrm{IE}}})} F_{\mathrm{IE}}^{i}$, $F_{\mathrm{E}} = \bigcup_{i\in\{\mathrm{SE},\mathrm{EE},\mathrm{IE}\}} F_i$, $S_{\mathrm{E}}^{\mathrm{Att}}$ is a set of event attributes and $S_{\mathrm{E}}^{\mathrm{AttV}}$ is a set of event attribute values. An event tuple is a 4-tuple $\Omega_{\mathrm{E}} = (\Omega_{\mathrm{SE}}, \Omega_{\mathrm{IE}}, \Omega_{\mathrm{EE}}, \Phi_{\mathrm{E}}^{\mathrm{Att}})$ where

- Ω_{SE} is a start-event tuple;
- Ω_{IE} is an intermediate-event tuple;
- Ω_{EE} is an end-event tuple; and

- $\Phi_{\mathrm{E}}^{\mathrm{Att}} : F_{\mathrm{E}} \times S_{\mathrm{E}}^{\mathrm{Att}} \rightarrow S_{\mathrm{E}}^{\mathrm{AttV}}$ relates an event and an event attribute to an event attribute value.

We define a function $\Phi_{\mathrm{E}}^{\mathrm{Att}}$ which returns the event attribute value for a particular event attribute of an event. An event tuple is specified in terms of a start-event tuple, an intermediate-event tuple, an end-event tuple and a function $\Phi_{\mathrm{E}}^{\mathrm{Att}}$.

Definition 5 (*Task tuple*) Suppose M_{L} represents the loop marker, M_{MI} represents the multiple instance marker, M_{C} represents the compensation marker, the valid combination of markers for tasks $S_{\mathrm{T}}^{\mathrm{M}} = \{\{M_{\mathrm{L}}\}, \{M_{\mathrm{MI}}\}, \{M_{\mathrm{C}}\}, \{M_{\mathrm{L}}, M_{\mathrm{C}}\}, \{M_{\mathrm{MI}}, M_{\mathrm{C}}\}\}$, the types of BPMN tasks Γ_{T} = {Service, Receive, Send, User, Script, Manual, Reference, None} and S_{TNames} is a set of task names. A task tuple is a 4-tuple $\Omega_{\mathrm{T}} = (F_{\mathrm{T}}, \Phi_{\mathrm{TM}}, \Phi_{\mathrm{Ttype}}, \Phi_{\mathrm{TName}})$ where

- F_{T} is a set of tasks;
- $\Phi_{\mathrm{TM}} : F_{\mathrm{T}} \rightarrow S_{\mathrm{T}}^{\mathrm{M}}$ defines for a task its set of markers;
- $\Phi_{\mathrm{Ttype}} : F_{\mathrm{T}} \rightarrow \Gamma_{\mathrm{T}}$ returns the type of a task; and
- $\Phi_{\mathrm{TName}} : F_{\mathrm{T}} \rightarrow S_{\mathrm{TNames}}$ maps a task to its name.

Definition 6 (*Subprocess tuple*) Suppose M_{CSP} represents the collapsed subprocess marker, M_{L} represents the loop marker, M_{MI} represents the multiple instance marker, M_{AD} represents the ad hoc marker, M_{C} represents the compensation marker, the valid combination of markers for collapsed subprocesses $S_{\mathrm{CSP}}^{\mathrm{M}} = \{\{M_{\mathrm{CSP}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{L}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{MI}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{AD}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{C}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{L}}, M_{\mathrm{AD}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{L}}, M_{\mathrm{C}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{MI}}, M_{\mathrm{AD}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{MI}}, M_{\mathrm{C}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{C}}, M_{\mathrm{AD}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{L}}, M_{\mathrm{AD}}, M_{\mathrm{C}}\}, \{M_{\mathrm{CSP}}, M_{\mathrm{MI}}, M_{\mathrm{AD}}, M_{\mathrm{C}}\}\}$, the valid combination of markers for expanded subprocesses $S_{\mathrm{ESP}}^{\mathrm{M}} = \{\{\}, \{M_{\mathrm{L}}\}, \{M_{\mathrm{MI}}\}, \{M_{\mathrm{AD}}\}, \{M_{\mathrm{C}}\}, \{M_{\mathrm{L}}, M_{\mathrm{AD}}\}, \{M_{\mathrm{L}}, M_{\mathrm{C}}\}, \{M_{\mathrm{MI}}, M_{\mathrm{AD}}\}, \{M_{\mathrm{MI}}, M_{\mathrm{C}}\}, \{M_{\mathrm{C}}, M_{\mathrm{AD}}\}, \{M_{\mathrm{L}}, M_{\mathrm{AD}}, M_{\mathrm{C}}\}, \{M_{\mathrm{MI}}, M_{\mathrm{AD}}, M_{\mathrm{C}}\}\}$, S_{NP} is a set of none-start-events processes, S_{P} is a set of processes and $\mathbb{B}$ is the set of Boolean values. A subprocess tuple is a 10-tuple $\Omega_{\mathrm{SP}} = (F_{\mathrm{SP}}^{\mathrm{Embed}}, F_{\mathrm{SP}}^{\mathrm{Reuse}}, F_{\mathrm{SP}}^{\mathrm{Ref}}, \Phi_{\mathrm{IsTX}}, \Phi_{\mathrm{SPM}}, \Phi_{\mathrm{SE}}^{\mathrm{Bdy}}, \Phi_{\mathrm{EE}}^{\mathrm{Bdy}}, \Phi_{\mathrm{NP}}, \Phi_{\mathrm{P}}, \Phi_{\mathrm{RP}})$ where

- $F_{\mathrm{SP}}^{\mathrm{Embed}}$ is a set of embedded subprocesses;
- $F_{\mathrm{SP}}^{\mathrm{Reuse}}$ is a set of reusable subprocesses;
- $F_{\mathrm{SP}}^{\mathrm{Ref}}$ is a set of reference subprocesses;
- $\Phi_{\mathrm{IsTX}} : F_{\mathrm{SP}}^{\mathrm{Embed}} \cup F_{\mathrm{SP}}^{\mathrm{Reuse}} \cup F_{\mathrm{SP}}^{\mathrm{Ref}} \rightarrow \mathbb{B}$ returns whether a subprocess is a transaction or not;
- $\Phi_{\mathrm{SPM}} : F_{\mathrm{SP}}^{\mathrm{Embed}} \cup F_{\mathrm{SP}}^{\mathrm{Reuse}} \cup F_{\mathrm{SP}}^{\mathrm{Ref}} \rightarrow S_{\mathrm{ESP}}^{\mathrm{M}} \cup S_{\mathrm{CSP}}^{\mathrm{M}}$ specifies for a subprocess its set of markers;
- $\Phi_{\mathrm{SE}}^{\mathrm{Bdy}} : \{x | x \in (F_{\mathrm{SP}}^{\mathrm{Embed}} \cup F_{\mathrm{SP}}^{\mathrm{Reuse}}) \wedge \Phi_{\mathrm{SPM}}(x) \in S_{\mathrm{ESP}}^{\mathrm{M}}\} \rightarrow 2^{F_{\mathrm{SE}}}$ returns the set of start events attached to the boundary of an expanded subprocess;
- $\Phi_{\mathrm{EE}}^{\mathrm{Bdy}} : \{x | x \in (F_{\mathrm{SP}}^{\mathrm{Embed}} \cup F_{\mathrm{SP}}^{\mathrm{Reuse}}) \wedge \Phi_{\mathrm{SPM}}(x) \in S_{\mathrm{ESP}}^{\mathrm{M}}\} \rightarrow 2^{F_{\mathrm{EE}}}$ returns the set of end events attached to the boundary of an expanded subprocess;
- $\Phi_{\mathrm{NP}} : F_{\mathrm{SP}}^{\mathrm{Embed}} \rightarrow S_{\mathrm{NP}}$ returns the associated none-start-events process;
- $\Phi_{\mathrm{P}} : F_{\mathrm{SP}}^{\mathrm{Reuse}} \rightarrow S_{\mathrm{P}}$ returns the called process; and
- $\Phi_{\mathrm{RP}} : F_{\mathrm{SP}}^{\mathrm{Ref}} \rightarrow \bigcup_{i \in \{\mathrm{Embed}, \mathrm{Reuse}, \mathrm{Ref}\}} F_{\mathrm{SP}}^{i}$ returns the subprocess being referenced.

A task tuple and a subprocess tuple comprise a collection of functions as well as, respectively, a set of tasks and sets of embedded subprocesses, reusable subprocesses and reference subprocesses. There are three kinds of task markers: loop markers, multiple instance markers and compensation markers. Likewise, four subprocess markers are allowed to use in both collapsed subprocesses and expanded subprocesses. These encompass loop markers, multiple instance markers, ad hoc markers and compensation markers.

Definition 7 (*Activity tuple*) Suppose Γ_{SP} = {Embed, Reuse, Ref}, Γ_{IE} = {None, Msg, Timer, Err, Cncl, Cmpen, Cond, Link, Sign, Multi}, Γ_{NLC} = {None, Link, Cncl}, Γ_{NL} = {None, Link}, $F_{\mathrm{A}} = F_{\mathrm{T}} \cup \bigcup_{i \in \Gamma_{\mathrm{SP}}} F_{\mathrm{SP}}^{i}$, $S_{\mathrm{TX}} = \{x | x \in \bigcup_{i \in \Gamma_{\mathrm{SP}}} F_{\mathrm{SP}}^{i} \wedge \Phi_{\mathrm{IsTX}}(x) = true\}$, $S_{\mathrm{A}}^{\mathrm{Att}}$ is a set of activity attributes and $S_{\mathrm{A}}^{\mathrm{AttV}}$ is a set of activity attribute values. An activity tuple is a 5-tuple $\Omega_{\mathrm{A}} = (\Omega_{\mathrm{T}}, \Omega_{\mathrm{SP}}, \Phi_{\mathrm{IE}}^{\mathrm{Bdy[-TX]}}, \Phi_{\mathrm{IE}}^{\mathrm{Bdy[TX]}}, \Phi_{\mathrm{A}}^{\mathrm{Att}})$ where

- Ω_{T} is a task tuple;
- Ω_{SP} is a subprocess tuple;
- $\Phi_{\mathrm{IE}}^{\mathrm{Bdy[-TX]}} : F_{\mathrm{A}} \backslash S_{\mathrm{TX}} \rightarrow 2^{\bigcup_{i \in \Gamma_{\mathrm{IE}} \backslash \Gamma_{\mathrm{NLC}}} F_{\mathrm{IE}}^{i}}$ returns the set of intermediate events attached to the boundary of an activity that is not a transaction;
- $\Phi_{\mathrm{IE}}^{\mathrm{Bdy[TX]}} : S_{\mathrm{TX}} \rightarrow 2^{\bigcup_{i \in \Gamma_{\mathrm{IE}} \backslash \Gamma_{\mathrm{NL}}} F_{\mathrm{IE}}^{i}}$ returns the set of intermediate events attached to the boundary of a transaction; and
- $\Phi_{\mathrm{A}}^{\mathrm{Att}} : F_{\mathrm{A}} \times S_{\mathrm{A}}^{\mathrm{Att}} \rightarrow S_{\mathrm{A}}^{\mathrm{AttV}}$ returns the activity attribute value of an activity and an activity attribute.

A task tuple, a subprocess tuple and a number of functions constitute an activity tuple. None intermediate events and link intermediate events cannot be attached to the boundary of an activity or a transaction. Additionally, cancel intermediate events are restricted to be placed on the boundary of a transaction.

Definition 8 (*Exclusive gateway tuple*) An exclusive gateway tuple is a 4-tuple $\Omega_{\mathrm{XG}} = (F_{\mathrm{XDG}}^{\mathrm{D}}, F_{\mathrm{XMG}}^{\mathrm{D}}, F_{\mathrm{XDG}}^{\mathrm{E}}, F_{\mathrm{XMG}}^{\mathrm{E}})$ where

- $F_{\mathrm{XDG}}^{\mathrm{D}}$ is a set of data-based exclusive decision gateways (DXDGs);

- $F^{\mathrm{D}}_{\mathrm{XMG}}$ is a set of data-based exclusive merge gateways (DXMGs);
- $F^{\mathrm{E}}_{\mathrm{XDG}}$ is a set of event-based exclusive decision gateways (EXDGs); and
- $F^{\mathrm{E}}_{\mathrm{XMG}}$ is a set of event-based exclusive merge gateways (EXMGs).

Sets of data-based exclusive decision gateways, data-based exclusive merge gateways, event-based exclusive decision gateways and event-based exclusive merge gateways form an exclusive gateway tuple.

Definition 9 (*Inclusive gateway tuple*) An inclusive gateway tuple is a 2-tuple $\Omega_{\mathrm{IG}} = (F_{\mathrm{IDG}}, F_{\mathrm{IMG}})$ where

- F_{IDG} is a set of inclusive decision gateways (IDGs); and
- F_{IMG} is a set of inclusive merge gateways (IMGs).

Definition 10 (*Complex gateway tuple*) A complex gateway tuple is a 2-tuple $\Omega_{\mathrm{CG}} = (F_{\mathrm{CDG}}, F_{\mathrm{CMG}})$ where

- F_{CDG} is a set of complex decision gateways (CDGs); and
- F_{CMG} is a set of complex merge gateways (CMGs).

Definition 11 (*Parallel gateway tuple*) A parallel gateway tuple is a 2-tuple $\Omega_{\mathrm{PG}} = (F_{\mathrm{PFG}}, F_{\mathrm{PJG}})$ where

- F_{PFG} is a set of parallel fork gateways (PFGs); and
- F_{PJG} is a set of parallel join gateways (PJGs).

Definition 12 (*Gateway tuple*) Suppose $\Gamma_{\mathrm{XG}} = \{\mathrm{XDG}, \mathrm{XMG}\}$, $\Gamma_{\mathrm{IG}} = \{\mathrm{IDG}, \mathrm{IMG}\}$, $\Gamma_{\mathrm{CG}} = \{\mathrm{CDG}, \mathrm{CMG}\}$, $\Gamma_{\mathrm{PG}} = \{\mathrm{PFG}, \mathrm{PJG}\}$, $F_{\mathrm{XG}} = \bigcup_{i\in\{\mathrm{D,E}\}} \bigcup_{j\in\Gamma_{\mathrm{XG}}} F^{i}_{j}$, $F_{\mathrm{IG}} = \bigcup_{i\in\Gamma_{\mathrm{IG}}} F_i$, $F_{\mathrm{CG}} = \bigcup_{i\in\Gamma_{\mathrm{CG}}} F_i$, $F_{\mathrm{PG}} = \bigcup_{i\in\Gamma_{\mathrm{PG}}} F_i$, $F_{\mathrm{G}} = \bigcup_{i\in\{\mathrm{XG,IG,CG,PG}\}} F_i$, $S^{\mathrm{Att}}_{\mathrm{G}}$ is a set of gateway attributes and $S^{\mathrm{AttV}}_{\mathrm{G}}$ is a set of gateway attribute values. A gateway tuple is a 5-tuple $\Omega_{\mathrm{G}} = (\Omega_{\mathrm{XG}}, \Omega_{\mathrm{IG}}, \Omega_{\mathrm{CG}}, \Omega_{\mathrm{PG}}, \Phi^{\mathrm{Att}}_{\mathrm{G}})$ where

- Ω_{XG} is an exclusive gateway tuple;
- Ω_{IG} is an inclusive gateway tuple;
- Ω_{CG} is a complex gateway tuple;
- Ω_{PG} is a parallel gateway tuple; and
- $\Phi^{\mathrm{Att}}_{\mathrm{G}}$: $F_{\mathrm{G}} \times S^{\mathrm{Att}}_{\mathrm{G}} \rightarrow S^{\mathrm{AttV}}_{\mathrm{G}}$ defines for a gateway and a gateway attribute the corresponding gateway attribute value.

The two types of inclusive gateways are: inclusive decision gateways and inclusive merge gateways. There are two sorts of complex gateways: complex decision gateways and complex merge gateways. A parallel gateway tuple is composed of sets of parallel fork gateways and parallel join gateways. By combining an exclusive gateway tuple, an inclusive gateway tuple, a complex gateway tuple, a parallel gateway tuple and a function $\Phi^{\mathrm{Att}}_{\mathrm{G}}$, a gateway tuple is obtained.

Definition 13 (*Connecting-object tuple*) Suppose $\Gamma^{\mathrm{src}}_{\mathrm{IE}} = \{\mathrm{Msg}, \overline{\mathrm{Msg}}, \mathrm{Timer}, \mathrm{Cond}, \mathrm{Link}, \overline{\mathrm{Link}}, \mathrm{Sign}, \overline{\mathrm{Sign}}\}$, $\Gamma^{\mathrm{trg}}_{\mathrm{IE}} = \{\mathrm{None}, \mathrm{Msg}, \overline{\mathrm{Msg}}, \mathrm{Timer}, \mathrm{Cmpen}, \overline{\mathrm{Cmpen}}, \mathrm{Cond}, \mathrm{Link}, \overline{\mathrm{Link}}, \mathrm{Sign}, \overline{\mathrm{Sign}}\}$, $S_{\mathrm{F}} = \bigcup_{i\in\{\mathrm{E,A,G}\}} F_i$, $S^{\mathrm{Bdy[-Cmpen]}}_{\mathrm{IE}} = \bigcup_{A\in(F_{\mathrm{T}}\cup\bigcup_{i\in\Gamma_{\mathrm{SP}}} F^{i}_{\mathrm{SP}}\setminus S_{\mathrm{TX}})} \Phi^{\mathrm{Bdy[-TX]}}_{\mathrm{IE}}(A) \cup \bigcup_{Tx\in S_{\mathrm{TX}}} \Phi^{\mathrm{Bdy[TX]}}_{\mathrm{IE}}(Tx) \setminus F^{\mathrm{Cmpen}}_{\mathrm{IE}}$ is the set of non-compensation intermediate events attached to activities and transactions, $S^{\mathrm{Cmpen}}_{\mathrm{A}} = \{x | (x \in F_{\mathrm{T}} \wedge M_{\mathrm{C}} \in \Phi_{\mathrm{TM}}(x)) \vee (x \in \bigcup_{i\in\Gamma_{\mathrm{SP}}} F^{i}_{\mathrm{SP}} \wedge M_{\mathrm{C}} \in \Phi_{\mathrm{SPM}}(x))\}$ is the set of activities with the compensation marker, $S^{\mathrm{NF[src]}}_{\mathrm{IE}} = \bigcup_{i\in\Gamma^{\mathrm{src}}_{\mathrm{IE}}} F^{i}_{\mathrm{IE}} \setminus (\bigcup_{A\in(F_{\mathrm{T}}\cup\bigcup_{i\in\Gamma_{\mathrm{SP}}} F^{i}_{\mathrm{SP}}\setminus S_{\mathrm{TX}})} \Phi^{\mathrm{Bdy[-TX]}}_{\mathrm{IE}}(A) \cup \bigcup_{Tx\in S_{\mathrm{TX}}} \Phi^{\mathrm{Bdy[TX]}}_{\mathrm{IE}}(Tx))$ is the set of intermediate events that are sources of normal or uncontrolled flows, $S^{\mathrm{NF[trg]}}_{\mathrm{IE}} = \bigcup_{i\in\Gamma^{\mathrm{trg}}_{\mathrm{IE}}} F^{i}_{\mathrm{IE}} \setminus (\bigcup_{A\in(F_{\mathrm{T}}\cup\bigcup_{i\in\Gamma_{\mathrm{SP}}} F^{i}_{\mathrm{SP}}\setminus S_{\mathrm{TX}})} \Phi^{\mathrm{Bdy[-TX]}}_{\mathrm{IE}}(A) \cup \bigcup_{Tx\in S_{\mathrm{TX}}} \Phi^{\mathrm{Bdy[TX]}}_{\mathrm{IE}}(Tx))$ is the set of intermediate events that are targets of normal or uncontrolled flows, $S^{\mathrm{Att}}_{\mathrm{C}}$ is a set of connecting object attributes and $S^{\mathrm{AttV}}_{\mathrm{C}}$ is a set of connecting object attribute values. A connecting-object tuple is a 7-tuple $\Omega_{\mathrm{C}} = (A_{\mathrm{DO}}, C_{\mathrm{SF}}, C_{\mathrm{DA}}, S_{\mathrm{Cond}}, \Phi_{\mathrm{Cond}}, \Phi_{\mathrm{IsDf}}, \Phi^{\mathrm{Att}}_{\mathrm{C}})$ where

- A_{DO} is a set of data objects;
- $C_{\mathrm{SF}} \subseteq (S_{\mathrm{F}} \setminus (F_{\mathrm{EE}} \cup S^{\mathrm{Cmpen}}_{\mathrm{A}} \cup F_{\mathrm{IE}}) \cup (\bigcup_{P\in F^{\mathrm{Embed}}_{\mathrm{SP}}} \Phi^{\mathrm{Bdy}}_{\mathrm{EE}}(P) \cup S^{\mathrm{Bdy[-Cmpen]}}_{\mathrm{IE}} \cup S^{\mathrm{NF[src]}}_{\mathrm{IE}})) \times (S_{\mathrm{F}} \setminus (F_{\mathrm{SE}} \cup F_{\mathrm{IE}}) \cup (\bigcup_{P\in F^{\mathrm{Embed}}_{\mathrm{SP}}} \Phi^{\mathrm{Bdy}}_{\mathrm{SE}}(P) \cup S^{\mathrm{NF[trg]}}_{\mathrm{IE}}))$ is a set of sequence flows (SFs);
- $C_{\mathrm{DA}} \subseteq (F_{\mathrm{A}} \times A_{\mathrm{DO}}) \cup (A_{\mathrm{DO}} \times F_{\mathrm{A}}) \cup (F^{\mathrm{Cmpen}}_{\mathrm{IE}} \times F_{\mathrm{A}})$ is a set of directed associations;
- S_{Cond} is a set of conditions;
- $\Phi_{\mathrm{Cond}} : C_{\mathrm{SF}} \rightarrow S_{\mathrm{Cond}}$ returns the condition of a sequence flow;
- $\Phi_{\mathrm{IsDf}} : C_{\mathrm{SF}} \rightarrow \mathbb{B}$ returns whether a sequence flow is a default sequence flow; and
- $\Phi^{\mathrm{Att}}_{\mathrm{C}} : \bigcup_{i\in\{\mathrm{SF,DA}\}} C_i \times S^{\mathrm{Att}}_{\mathrm{C}} \rightarrow S^{\mathrm{AttV}}_{\mathrm{C}}$ relates a connecting object and a connecting object attribute to a connecting object attribute value.

In Definition 13, the expression $(S_{\mathrm{F}} \setminus (F_{\mathrm{EE}} \cup S^{\mathrm{Cmpen}}_{\mathrm{A}} \cup F_{\mathrm{IE}}) \cup (\bigcup_{P\in F^{\mathrm{Embed}}_{\mathrm{SP}}} \Phi^{\mathrm{Bdy}}_{\mathrm{EE}}(P) \cup S^{\mathrm{Bdy[-Cmpen]}}_{\mathrm{IE}} \cup S^{\mathrm{NF[src]}}_{\mathrm{IE}}))$ states that

(i) an end event cannot be a source flow object with the exception that it is attached to the boundary of an expanded subprocess; and
(ii) a compensation activity does not have any outgoing sequence flows.

In a similar way, the expression $(S_{\mathrm{F}} \setminus (F_{\mathrm{SE}} \cup F_{\mathrm{IE}}) \cup (\bigcup_{P \in F_{\mathrm{SP}}^{\mathrm{Embed}}} \Phi_{\mathrm{SE}}^{\mathrm{Bdy}}(P) \cup S_{\mathrm{IE}}^{\mathrm{NF[trg]}}))$ says that a start event cannot be a target flow object except it is attached to the boundary of an expanded subprocess. A sequence flow is a subset of the cross product of these two expressions. The expression $(F_{\mathrm{A}} \times A_{\mathrm{DO}}) \cup (A_{\mathrm{DO}} \times F_{\mathrm{A}}) \cup (S_{\mathrm{IE}}^{\mathrm{Cmpen}} \times F_{\mathrm{A}})$ stipulates that a directed association connects

(i) a data object with an activity; or
(ii) a compensation intermediate event for catching the event trigger with an activity.

Definition 14 (*Process*) A process is a 4-tuple $P = (\Omega_{\mathrm{E}}, \Omega_{\mathrm{A}}, \Omega_{\mathrm{G}}, \Omega_{\mathrm{C}})$ where

- Ω_{E} is an event tuple;
- Ω_{A} is an activity tuple;
- Ω_{G} is a gateway tuple; and
- Ω_{C} is a connecting-object tuple.

A process consists of four components: an event tuple, an activity tuple, a gateway tuple and a connecting-object tuple.

Definition 15 (*None-start-events process*) Given a process P with a start-event tuple $\Omega_{\mathrm{SE}} = (F_{\mathrm{SE}}^{\mathrm{None}}, F_{\mathrm{SE}}^{\mathrm{Msg}}, F_{\mathrm{SE}}^{\mathrm{Timer}}, F_{\mathrm{SE}}^{\mathrm{Cond}}, F_{\mathrm{SE}}^{\mathrm{Sign}}, F_{\mathrm{SE}}^{\mathrm{Multi}})$. The process P is a none-start-events process if and only if $\bigwedge_{i \in \Gamma_{\mathrm{SE}} \setminus \{\mathrm{None}\}} (F_{\mathrm{SE}}^{i} = \emptyset)$.

Definition 16 The function Φ_{TP}, defined below, returns the task name, none-start-events process, called process or referenced subprocess depending on whether the parameter is a task, an embedded subprocess, a reusable subprocess or a reference subprocess.

$$\Phi_{\mathrm{TP}}(x) = \begin{cases} \Phi_{\mathrm{TName}}(x) & \text{if } x \in F_{\mathrm{T}} \\ \Phi_{\mathrm{NP}}(x) & \text{if } x \in F_{\mathrm{SP}}^{\mathrm{Embed}} \\ \Phi_{\mathrm{P}}(x) & \text{if } x \in F_{\mathrm{SP}}^{\mathrm{Reuse}} \\ \Phi_{\mathrm{RP}}(x) & \text{if } x \in F_{\mathrm{SP}}^{\mathrm{Ref}} \end{cases}$$

5 Classification of equivalences of BPMN processes

This section aims to study different kinds of equivalences of BPMN processes from a formal perspective. The principles of equivalences are specified precisely in the form of a number of definitions. In essence, two BPMN processes are regarded as equivalent if both of them can be transformed into a common graphical representation.

In what follows, definitions for DXMG-based-duplicate-free form and DXDG-based-duplicate-free form are provided. A discussion of two types of equivalences of BPMN processes, which are based on DXMG-based-duplicate-free form and DXDG-based-duplicate-free form, is offered. These encompass DXMG-DF-equivalence and DXDG-DF-equivalence. The motivation for introducing DXMG-DF-equivalence and DXDG-DF-equivalence is to reduce redundant activities by adjusting the positions of data-based exclusive merge gateway and data-based exclusive decision gateway.

Definition 17 (*DXMG-based-duplicate-free form*) Let P_1 be a process with $\Omega_{\mathrm{A}}, \Omega_{\mathrm{C}}, \Omega_{\mathrm{T}}, \Omega_{\mathrm{SP}}, C_{\mathrm{SF}}, F_{\mathrm{T}}, F_{\mathrm{SP}}^{\mathrm{Embed}}, F_{\mathrm{SP}}^{\mathrm{Reuse}}, F_{\mathrm{SP}}^{\mathrm{Ref}}$ are replaced by $\Omega_{\mathrm{A}^{(\mathrm{P}_1)}}, \Omega_{\mathrm{C}^{(\mathrm{P}_1)}}, \Omega_{\mathrm{T}^{(\mathrm{P}_1)}}, \Omega_{\mathrm{SP}^{(\mathrm{P}_1)}}, C_{\mathrm{SF}^{(\mathrm{P}_1)}}, F_{\mathrm{T}^{(\mathrm{P}_1)}}, F_{\mathrm{SP}^{(\mathrm{P}_1)}}^{\mathrm{Embed}}, F_{\mathrm{SP}^{(\mathrm{P}_1)}}^{\mathrm{Reuse}}, F_{\mathrm{SP}^{(\mathrm{P}_1)}}^{\mathrm{Ref}}, F_{\mathrm{A}^{(\mathrm{P}_1)}} = F_{\mathrm{T}^{(\mathrm{P}_1)}} \cup \bigcup_{i \in \Gamma_{\mathrm{SP}}} F_{\mathrm{SP}^{(\mathrm{P}_1)}}^{i}$ and $S_{\mathrm{F}^{(\mathrm{P}_1)}} = F_{\mathrm{A}^{(\mathrm{P}_1)}} \cup \bigcup_{i \in \{\mathrm{E},\mathrm{G}\}} F_i$. If $FO_1, FO_2 \in S_{\mathrm{F}^{(\mathrm{P}_1)}}, A_{1,1}, A_{1,2}, \ldots, A_{1,n}, A_{2,1}, A_{2,2}, \ldots, A_{2,n}, A_{3,1}, A_{3,2}, \ldots, A_{3,n}, \ldots, A_{m-1,1}, A_{m-1,2}, \ldots, A_{m-1,n}, A_{m,1}, A_{m,2}, \ldots, A_{m,n} \in F_{\mathrm{A}^{(\mathrm{P}_1)}}, G_1 \in F_{\mathrm{XDG}}^{\mathrm{D}}, G_2 \in F_{\mathrm{XMG}}^{\mathrm{D}}, c_1, c_2, \ldots, c_n \in S_{\mathrm{Cond}}, (FO_1, G_1), (G_1, A_{1,1}), (G_1, A_{1,2}), \ldots, (G_1, A_{1,n}), (A_{1,1}, A_{2,1}), (A_{1,2}, A_{2,2}), \ldots, (A_{1,n}, A_{2,n}), (A_{2,1}, A_{3,1}), (A_{2,2}, A_{3,2}), \ldots, (A_{2,n}, A_{3,n}), \ldots, (A_{m-1,1}, A_{m,1}), (A_{m-1,2}, A_{m,2}), \ldots, (A_{m-1,n}, A_{m,n}), (A_{m,1}, G_2), (A_{m,2}, G_2), \ldots, (A_{m,n}, G_2), (G_2, FO_2) \in C_{\mathrm{SF}^{(\mathrm{P}_1)}}, \Phi_{\mathrm{Cond}}((G_1, A_{1,i})) = c_i, SS_{\mathrm{A}^{(\mathrm{P}_1)}} = \{A_{2,1}, A_{2,2}, \ldots, A_{2,n}, A_{3,1}, A_{3,2}, \ldots, A_{3,n}, \ldots, A_{m-1,1}, A_{m-1,2}, \ldots, A_{m-1,n}, A_{m,1}, A_{m,2}, \ldots, A_{m,n}\}, CC_{\mathrm{SF}^{(\mathrm{P}_1)}} = \{(A_{1,1}, A_{2,1}), (A_{1,2}, A_{2,2}), \ldots, (A_{1,n}, A_{2,n}), (A_{2,1}, A_{3,1}), (A_{2,2}, A_{3,2}), \ldots, (A_{2,n}, A_{3,n}), \ldots, (A_{m-1,1}, A_{m,1}), (A_{m-1,2}, A_{m,2}), \ldots, (A_{m-1,n}, A_{m,n}), (A_{m,1}, G_2), (A_{m,2}, G_2), \ldots, (A_{m,n}, G_2), (G_2, FO_2)\}, =_{i=1}^{n} \Phi_{\mathrm{TP}^{(\mathrm{P}_1)}}(A_{j,i}), FF_{\mathrm{T}^{(\mathrm{P}_1)}} = \{x | x \in SS_{\mathrm{A}^{(\mathrm{P}_1)}} \wedge x \in F_{\mathrm{T}^{(\mathrm{P}_1)}}\}, FF_{\mathrm{SP}^{(\mathrm{P}_1)}}^{k} = \{x | x \in SS_{\mathrm{A}^{(\mathrm{P}_1)}} \wedge x \in F_{\mathrm{SP}^{(\mathrm{P}_1)}}^{k}\}$ for $i = 1, \ldots, n$, $j = 2, 3, \ldots, m-1, m$ and $k \in \Gamma_{\mathrm{SP}}$, then there is a unique process P_2 which is in DXMG-based-duplicate-free form such that

- $\Omega_{\mathrm{A}}, \Omega_{\mathrm{C}}, \Omega_{\mathrm{T}}, \Omega_{\mathrm{SP}}, C_{\mathrm{SF}}, F_{\mathrm{T}}, F_{\mathrm{SP}}^{\mathrm{Embed}}, F_{\mathrm{SP}}^{\mathrm{Reuse}}, F_{\mathrm{SP}}^{\mathrm{Ref}}$ are replaced by $\Omega_{\mathrm{A}^{(\mathrm{P}_2)}}, \Omega_{\mathrm{C}^{(\mathrm{P}_2)}}, \Omega_{\mathrm{T}^{(\mathrm{P}_2)}}, \Omega_{\mathrm{SP}^{(\mathrm{P}_2)}}, C_{\mathrm{SF}^{(\mathrm{P}_2)}}, F_{\mathrm{T}^{(\mathrm{P}_2)}}, F_{\mathrm{SP}^{(\mathrm{P}_2)}}^{\mathrm{Embed}}, F_{\mathrm{SP}^{(\mathrm{P}_2)}}^{\mathrm{Reuse}}, F_{\mathrm{SP}^{(\mathrm{P}_2)}}^{\mathrm{Ref}}$,
- $F_{\mathrm{A}^{(\mathrm{P}_2)}} = F_{\mathrm{T}^{(\mathrm{P}_2)}} \cup \bigcup_{i \in \Gamma_{\mathrm{SP}}} F_{\mathrm{SP}^{(\mathrm{P}_2)}}^{i}$,
- $S_{\mathrm{F}^{(\mathrm{P}_2)}} = F_{\mathrm{A}^{(\mathrm{P}_2)}} \cup \bigcup_{i \in \{\mathrm{E},\mathrm{G}\}} F_i$,
- $FO_1, FO_2 \in S_{\mathrm{F}^{(\mathrm{P}_2)}}$,
- $A_{1,1}, A_{1,2}, \ldots, A_{1,n}, A_2, A_3, \ldots, A_{m-1}, A_m \in F_{\mathrm{A}^{(\mathrm{P}_2)}}$,
- $(FO_1, G_1), (G_1, A_{1,1}), (G_1, A_{1,2}), \ldots, (G_1, A_{1,n}), (A_{1,1}, G_2), (A_{1,2}, G_2), \ldots, (A_{1,n}, G_2), (G_2, A_2), (A_2, A_3), \ldots, (A_{m-1}, A_m), (A_m, FO_2) \in C_{\mathrm{SF}^{(\mathrm{P}_2)}}$,
- $\Phi_{\mathrm{Cond}}((G_1, A_{1,i})) = c_i$,
- $SS_{\mathrm{A}^{(\mathrm{P}_2)}} = \{A_2, A_3, \ldots, A_{m-1}, A_m\}$,
- $CC_{\mathrm{SF}^{(\mathrm{P}_2)}} = \{(A_{1,1}, G_2), (A_{1,2}, G_2), \ldots, (A_{1,n}, G_2), (G_2, A_2), (A_2, A_3), \ldots, (A_{m-1}, A_m), (A_m, FO_2)\}$,
- $FF_{\mathrm{T}^{(\mathrm{P}_2)}} = \{A_i | A_i \in SS_{\mathrm{A}^{(\mathrm{P}_2)}} \wedge (A_{i,1}) \in F_{\mathrm{T}^{(\mathrm{P}_1)}}\}$,
- $FF_{\mathrm{SP}^{(\mathrm{P}_2)}}^{k} = \{A_i | A_i \in SS_{\mathrm{A}^{(\mathrm{P}_2)}} \wedge (A_{i,1}) \in F_{\mathrm{SP}^{(\mathrm{P}_1)}}^{k}\}$,
- $\bigwedge_{j=2}^{m} (\Phi_{\mathrm{TP}^{(\mathrm{P}_2)}}(A_j) = \Phi_{\mathrm{TP}^{(\mathrm{P}_1)}}(A_{j,1}))$,
- $F_{\mathrm{T}^{(\mathrm{P}_2)}} = F_{\mathrm{T}^{(\mathrm{P}_1)}} \setminus FF_{\mathrm{T}^{(\mathrm{P}_1)}} \cup FF_{\mathrm{T}^{(\mathrm{P}_2)}}$,
- $F_{\mathrm{SP}^{(\mathrm{P}_2)}}^{k} = F_{\mathrm{SP}^{(\mathrm{P}_1)}}^{k} \setminus FF_{\mathrm{SP}^{(\mathrm{P}_1)}}^{k} \cup FF_{\mathrm{SP}^{(\mathrm{P}_2)}}^{k}$ and
- $C_{\mathrm{SF}^{(\mathrm{P}_2)}} = C_{\mathrm{SF}^{(\mathrm{P}_1)}} \setminus CC_{\mathrm{SF}^{(\mathrm{P}_1)}} \cup CC_{\mathrm{SF}^{(\mathrm{P}_2)}}$

for $i = 1, \ldots, n$ and $k \in \Gamma_{\mathrm{SP}}$.

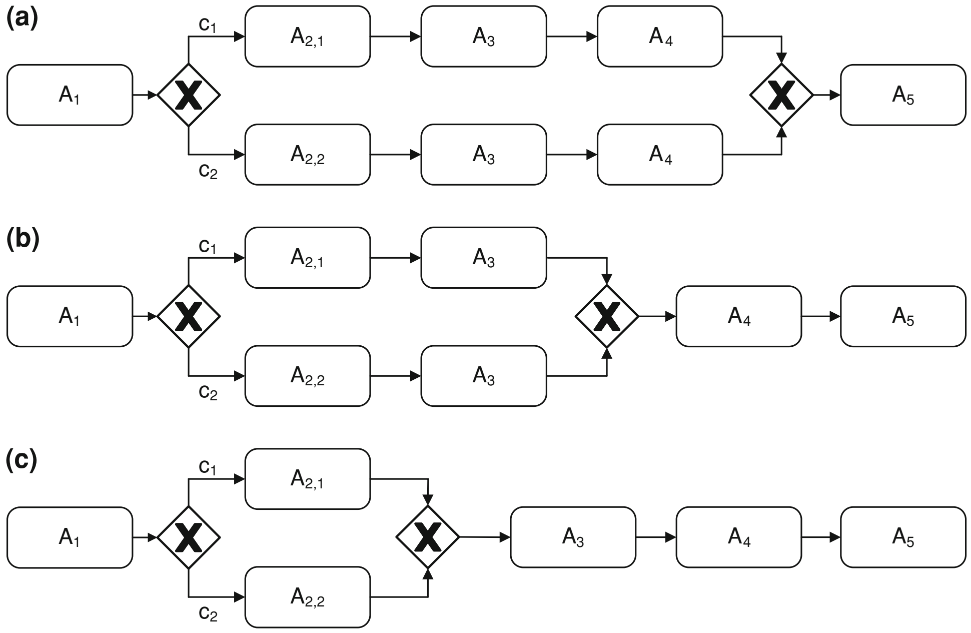


Fig. 5 DXMG-DF-equivalent BPMN processes

A BPMN process containing a data-based exclusive merge gateway preceded with sets of duplicate activities is transformable into a unique BPMN process by placing the data-based exclusive merge gateway before the sets of duplicate activities such that all duplicate activities can be eliminated. The sets $F_{\mathrm{T(P_1)}}$ and $F_{\mathrm{T(P_2)}}$ denote, respectively, the sets of tasks of P_1 and P_2. The set $F_{\mathrm{T(P_2)}}$ is the same as the set $F_{\mathrm{T(P_1)}}$ except that the set $FF_{\mathrm{T(P_1)}}$ is removed and the set $FF_{\mathrm{T(P_2)}}$ is added. The removal and addition of tasks eliminate the duplicate tasks in P_1. The sets $F^k_{\mathrm{SP(P_2)}}$ and $C_{\mathrm{SF(P_2)}}$ are defined in an analogous manner.

Definition 18 (*DXMG-DF-equivalence*) For any BPMN processes P_1 and P_2, P_1 is DXMG-DF-equivalent to P_2, denoted by $P_1 \approx^{\mathrm{DXMG}}_{\mathrm{DF}} P_2$, if and only if there is a BPMN process P_3 such that P_3 is a DXMG-based-duplicate-free form of P_1 and P_2.

Definition 18 stipulates that two BPMN processes are DXMG-DF-equivalent if they are capable of converting into a BPMN process which is in DXMG-based-duplicate-free form.

Proposition 1 *The relation* $\approx^{\mathrm{DXMG}}_{\mathrm{DF}}$ *is transitive.*

Proof Suppose $P_1 \approx^{\mathrm{DXMG}}_{\mathrm{DF}} P_2$ and $P_2 \approx^{\mathrm{DXMG}}_{\mathrm{DF}} P_3$. Since $P_1 \approx^{\mathrm{DXMG}}_{\mathrm{DF}} P_2$ and $P_2 \approx^{\mathrm{DXMG}}_{\mathrm{DF}} P_3$, it follows that P_4 is a DXMG-based-duplicate-free form of P_1, P_2 and P_3. Since P_4 is a DXMG-based-duplicate-free form of P_1 and P_3, we obtain $P_1 \approx^{\mathrm{DXMG}}_{\mathrm{DF}} P_3$. Thus, $\approx^{\mathrm{DXMG}}_{\mathrm{DF}}$ is transitive. □

Proposition 2 *The relation* $\approx^{\mathrm{DXMG}}_{\mathrm{DF}}$ *is an equivalence.*

Proof For reflexivity and symmetry, these follow immediately from Definition 18. Thus, $\approx^{\mathrm{DXMG}}_{\mathrm{DF}}$ is an equivalence as $\approx^{\mathrm{DXMG}}_{\mathrm{DF}}$ is transitive by Proposition 1. □

The concept of DXMG-DF-equivalence is illustrated in Fig. 5. The BPMN processes in Fig. 5a and b are DXMG-DF-equivalent as the BPMN process in Fig. 5c is a DXMG-based-duplicate-free form of them.

Definition 19 (*DXDG-based-duplicate-free form*) Let P_1 be a process with Ω_{A}, Ω_{C}, Ω_{T}, Ω_{SP}, C_{SF}, F_{T}, $F^{\mathrm{Embed}}_{\mathrm{SP}}$, $F^{\mathrm{Reuse}}_{\mathrm{SP}}$, $F^{\mathrm{Ref}}_{\mathrm{SP}}$ are replaced by $\Omega_{\mathrm{A(P_1)}}$, $\Omega_{\mathrm{C(P_1)}}$, $\Omega_{\mathrm{T(P_1)}}$, $\Omega_{\mathrm{SP(P_1)}}$, $C_{\mathrm{SF(P_1)}}$, $F_{\mathrm{T(P_1)}}$, $F^{\mathrm{Embed}}_{\mathrm{SP(P_1)}}$, $F^{\mathrm{Reuse}}_{\mathrm{SP(P_1)}}$, $F^{\mathrm{Ref}}_{\mathrm{SP(P_1)}}$, $F_{\mathrm{A(P_1)}} = F_{\mathrm{T(P_1)}} \cup \bigcup_{i\in\Gamma_{\mathrm{SP}}} F^i_{\mathrm{SP(P_1)}}$ and $S_{\mathrm{F(P_1)}} = F_{\mathrm{A(P_1)}} \cup \bigcup_{i\in\{\mathrm{E,G}\}} F_i$. If $FO_1, FO_2 \in S_{\mathrm{F(P_1)}}, A_{1,1}, A_{1,2}, \ldots, A_{1,n}, \ldots, A_{m-1,1}, A_{m-1,2}, \ldots, A_{m-1,n}, A_{m,1}, A_{m,2}, \ldots, A_{m,n} \in F_{\mathrm{A(P_1)}}$, $G_1 \in F^{\mathrm{D}}_{\mathrm{XDG}}$, $G_2 \in F^{\mathrm{D}}_{\mathrm{XMG}}$, $c_1, c_2, \ldots, c_n \in S_{\mathrm{Cond}}$, (FO_1, G_1), $(G_1, A_{1,1})$, $(G_1, A_{1,2})$, $\ldots$, $(G_1, A_{1,n})$, $\ldots$, $(A_{m-1,1}, A_{m,1})$, $(A_{m-1,2}, A_{m,2})$, $\ldots$, $(A_{m-1,n}, A_{m,n})$, $(A_{m,1}, G_2)$, $(A_{m,2}, G_2)$, $\ldots$, $(A_{m,n}, G_2)$, $(G_2, FO_2) \in C_{\mathrm{SF(P_1)}}$, $\Phi_{\mathrm{Cond}}((G_1, A_{1,i})) = c_i$, $SS_{\mathrm{A(P_1)}} = \{A_{1,1}, A_{1,2}, \ldots, A_{1,n}, \ldots, A_{m-1,1}, A_{m-1,2}, \ldots, A_{m-1,n}\}$, $CC_{\mathrm{SF(P_1)}} = \{(FO_1, G_1), (G_1, A_{1,1}), (G_1, A_{1,2}), \ldots, (G_1, A_{1,n}), \ldots, (A_{m-1,1}, A_{m,1}), (A_{m-1,2}, A_{m,2}), \ldots, (A_{m-1,n}, A_{m,n})\}$, $=^n_{i=1} \Phi_{\mathrm{TP(P_1)}}(A_j, A_i)$, $FF_{\mathrm{T(P_1)}} = \{x | x \in SS_{\mathrm{A(P_1)}} \wedge x \in F_{\mathrm{T(P_1)}}\}$, $FF^k_{\mathrm{SP(P_1)}} = \{x | x \in SS_{\mathrm{A(P_1)}} \wedge x \in F^k_{\mathrm{SP(P_1)}}\}$ for $i = 1, \ldots, n$, $j = 1, \ldots, m-1$ and $k \in \Gamma_{SP}$, then there is a unique process P_2 which is in DXDG-based-duplicate-free form such that

- Ω_{A}, Ω_{C}, Ω_{T}, Ω_{SP}, C_{SF}, F_{T}, $F^{\mathrm{Embed}}_{\mathrm{SP}}$, $F^{\mathrm{Reuse}}_{\mathrm{SP}}$, $F^{\mathrm{Ref}}_{\mathrm{SP}}$ are replaced by $\Omega_{\mathrm{A(P_2)}}$, $\Omega_{\mathrm{C(P_2)}}$, $\Omega_{\mathrm{T(P_2)}}$, $\Omega_{\mathrm{SP(P_2)}}$, $C_{\mathrm{SF(P_2)}}$, $F_{\mathrm{T(P_2)}}$, $F^{\mathrm{Embed}}_{\mathrm{SP(P_2)}}$, $F^{\mathrm{Reuse}}_{\mathrm{SP(P_2)}}$, $F^{\mathrm{Ref}}_{\mathrm{SP(P_2)}}$,
- $F_{\mathrm{A(P_2)}} = F_{\mathrm{T(P_2)}} \cup \bigcup_{i\in\Gamma_{\mathrm{SP}}} F^i_{\mathrm{SP(P_2)}}$,
- $S_{\mathrm{F(P_2)}} = F_{\mathrm{A(P_2)}} \cup \bigcup_{i\in\{\mathrm{E,G}\}} F_i$,
- $FO_1, FO_2 \in S_{\mathrm{F(P_2)}}$,
- $A_1, \ldots, A_{m-1}, A_{m,1}, A_{m,2}, \ldots, A_{m,n} \in F_{\mathrm{A(P_2)}}$,

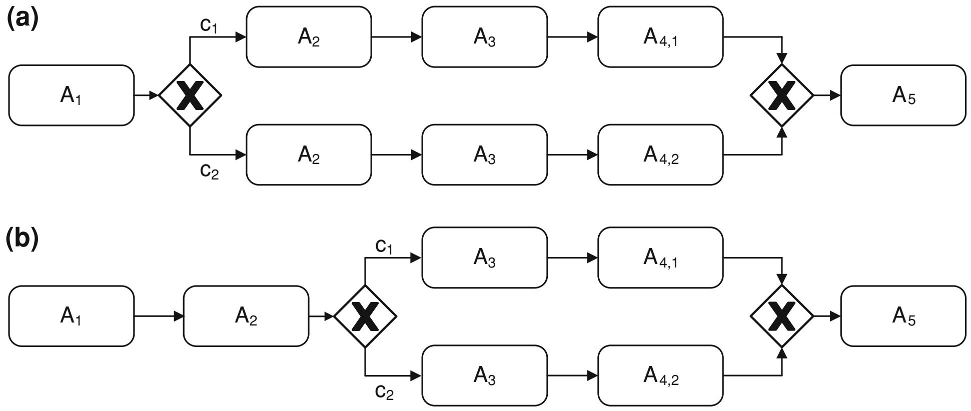


Fig. 6 DXDG-DF-equivalent BPMN processes

- $(FO_1, A_1), (A_{m-1}, G_1), (G_1, A_{m,1}), (G_1, A_{m,2}), \ldots, (G_1, A_{m,n}), (A_{m,1}, G_2), (A_{m,2}, G_2), \ldots, (A_{m,n}, G_2), (G_2, FO_2) \in C_{\mathrm{SF}^{(\mathrm{P}_2)}}$,
- $\Phi_{\mathrm{Cond}}((G_1, A_{m,i})) = c_i$,
- $SS_{\mathrm{A}^{(\mathrm{P}_2)}} = \{A_1, \ldots, A_{m-1}\}$,
- $CC_{\mathrm{SF}^{(\mathrm{P}_2)}} = \{(FO_1, A_1), \ldots, (A_{m-1}, G_1), (G_1, A_{m,1}), (G_1, A_{m,2}), \ldots, (G_1, A_{m,n})\}$,
- $FF_{\mathrm{T}^{(\mathrm{P}_2)}} = \{A_j | A_j \in SS_{\mathrm{A}^{(\mathrm{P}_2)}} \wedge (A_{j,1}) \in F_{\mathrm{T}^{(\mathrm{P}_1)}}\}$,
- $FF^k_{\mathrm{SP}^{(\mathrm{P}_2)}} = \{A_j | A_j \in SS_{\mathrm{A}^{(\mathrm{P}_2)}} \wedge (A_{j,1}) \in F^k_{\mathrm{SP}^{(\mathrm{P}_1)}}\}$,
- $\bigwedge_{j=1}^{m-1} (\Phi_{\mathrm{TP}^{(\mathrm{P}_2)}}(A_j) = \Phi_{\mathrm{TP}^{(\mathrm{P}_1)}}(A_{j,1}))$,
- $F_{\mathrm{T}^{(\mathrm{P}_2)}} = F_{\mathrm{T}^{(\mathrm{P}_1)}} \backslash FF_{\mathrm{T}^{(\mathrm{P}_1)}} \cup FF_{\mathrm{T}^{(\mathrm{P}_2)}}$,
- $F^k_{\mathrm{SP}^{(\mathrm{P}_2)}} = F^k_{\mathrm{SP}^{(\mathrm{P}_1)}} \backslash FF^k_{\mathrm{SP}^{(\mathrm{P}_1)}} \cup FF^k_{\mathrm{SP}^{(\mathrm{P}_2)}}$ and
- $C_{\mathrm{SF}^{(\mathrm{P}_2)}} = C_{\mathrm{SF}^{(\mathrm{P}_1)}} \backslash CC_{\mathrm{SF}^{(\mathrm{P}_1)}} \cup CC_{\mathrm{SF}^{(\mathrm{P}_2)}}$

for $i = 1, \ldots, n$, $j = 1, \ldots, m-1$ and $k \in \Gamma_{\mathrm{SP}}$.

We consider a BPMN process which is composed of a data-based exclusive decision gateway followed by sets of duplicate activities. To eliminate duplicate activities, the data-based exclusive decision gateway is moved after the sets of duplicate activities. The definitions of $F_{\mathrm{T}^{(\mathrm{P}_2)}}$ and $F^k_{\mathrm{SP}^{(\mathrm{P}_2)}}$ specify the elimination of duplicate activities. The removal of duplicate activities results in a structurally different process P_2. The set $C_{\mathrm{SF}^{(\mathrm{P}_2)}}$ is obtained by deleting and adding, respectively, the collections of sequence flows $CC_{\mathrm{SF}^{(\mathrm{P}_1)}}$ and $CC_{\mathrm{SF}^{(\mathrm{P}_2)}}$.

Definition 20 (*DXDG-DF-equivalence*) For any BPMN processes P_1 and P_2, P_1 is DXDG-DF-equivalent to P_2, denoted by $P_1 \approx^{\mathrm{DXDG}}_{\mathrm{DF}} P_2$, if and only if there is a BPMN process P_3 such that P_3 is a DXDG-based-duplicate-free form of P_1 and P_2.

Like DXMG-DF-equivalence, two BPMN processes are DXDG-DF-equivalent if both of them can be transformed into a common representation in DXDG-based-duplicate-free form.

Proposition 3 *The relation* $\approx^{\mathrm{DXDG}}_{\mathrm{DF}}$ *is transitive.*

Proof By similar argument as Proposition 1. □

Proposition 4 *The relation* $\approx^{\mathrm{DXDG}}_{\mathrm{DF}}$ *is an equivalence.*

Proof Analogous to Proposition 2. □

Figure 6 shows an example of DXDG-DF-equivalence. The BPMN processes in Fig. 6a and b are DXDG-DF-equivalent.

Definition 21 (*Implicit-PG form*) Let P_1 be a process with Ω_{A}, Ω_{G}, Ω_{C}, Ω_{SP}, Ω_{PG}, F_{T}, $F^{\mathrm{Embed}}_{\mathrm{SP}}$, $F^{\mathrm{Reuse}}_{\mathrm{SP}}$, $F^{\mathrm{Ref}}_{\mathrm{SP}}$, F_{PFG}, F_{PJG}, C_{SF} are replaced by $\Omega_{\mathrm{A}^{(\mathrm{P}_1)}}$, $\Omega_{\mathrm{G}^{(\mathrm{P}_1)}}$, $\Omega_{\mathrm{C}^{(\mathrm{P}_1)}}$, $\Omega_{\mathrm{SP}^{(\mathrm{P}_1)}}$, $\Omega_{\mathrm{PG}^{(\mathrm{P}_1)}}$, $F_{\mathrm{T}^{(\mathrm{P}_1)}}$, $F^{\mathrm{Embed}}_{\mathrm{SP}^{(\mathrm{P}_1)}}$, $F^{\mathrm{Reuse}}_{\mathrm{SP}^{(\mathrm{P}_1)}}$, $F^{\mathrm{Ref}}_{\mathrm{SP}^{(\mathrm{P}_1)}}$, $F_{\mathrm{PFG}^{(\mathrm{P}_1)}}$, $F_{\mathrm{PJG}^{(\mathrm{P}_1)}}$, $C_{\mathrm{SF}^{(\mathrm{P}_1)}}$, $F_{\mathrm{PG}^{(\mathrm{P}_1)}} = \bigcup_{i \in \Gamma_{\mathrm{PG}}} F_{i^{(\mathrm{P}_1)}}$, $F_{\mathrm{G}^{(\mathrm{P}_1)}} = \bigcup_{i \in \{\mathrm{XG,IG,CG}\}} F_i \cup F_{\mathrm{PG}^{(\mathrm{P}_1)}}$, $F_{\mathrm{A}^{(\mathrm{P}_1)}} = F_{\mathrm{T}^{(\mathrm{P}_1)}} \cup \bigcup_{i \in \Gamma_{\mathrm{SP}}} F^i_{\mathrm{SP}^{(\mathrm{P}_1)}}$, $S_{\mathrm{F}^{(\mathrm{P}_1)}} = F_{\mathrm{E}} \cup \bigcup_{i \in \{\mathrm{A,G}\}} F_{i^{(\mathrm{P}_1)}}$. If $FO_1, FO_2 \in S_{\mathrm{F}^{(\mathrm{P}_1)}}$, $A_i \in F_{\mathrm{A}^{(\mathrm{P}_1)}}$, $G_1 \in F_{\mathrm{PFG}^{(\mathrm{P}_1)}}$, $G_2 \in F_{\mathrm{PJG}^{(\mathrm{P}_1)}}$, $(FO_1, G_1), (G_1, A_1), (G_1, A_2), \ldots, (G_1, A_n), (A_1, G_2), (A_2, G_2), \ldots, (A_n, G_2), (G_2, FO_2) \in C_{\mathrm{SF}^{(\mathrm{P}_1)}}$, $SS_{\mathrm{A}^{(\mathrm{P}_1)}} = \{A_1, A_2, \ldots, A_n\}$, $CC_{\mathrm{SF}^{(\mathrm{P}_1)}} = \{(FO_1, G_1), (G_1, A_1), (G_1, A_2), \ldots, (G_1, A_n), (A_1, G_2), (A_2, G_2), \ldots, (A_n, G_2), (G_2, FO_2)\}$, $FF_{\mathrm{T}^{(\mathrm{P}_1)}} = \{x | x \in SS_{\mathrm{A}^{(\mathrm{P}_1)}} \wedge x \in F_{\mathrm{T}^{(\mathrm{P}_1)}}\}$, $FF^k_{\mathrm{SP}^{(\mathrm{P}_1)}} = \{x | x \in SS_{\mathrm{A}^{(\mathrm{P}_1)}} \wedge x \in F^k_{\mathrm{SP}^{(\mathrm{P}_1)}}\}$ for $i = 1, \ldots, n$ and $k \in \Gamma_{\mathrm{SP}}$, then there is a unique process P_2 which is in implicit-PG form such that

- $\Omega_{\mathrm{A}}, \Omega_{\mathrm{G}}, \Omega_{\mathrm{C}}, \Omega_{\mathrm{SP}}, \Omega_{\mathrm{PG}}, F_{\mathrm{T}}, F^{\mathrm{Embed}}_{\mathrm{SP}}, F^{\mathrm{Reuse}}_{\mathrm{SP}}, F^{\mathrm{Ref}}_{\mathrm{SP}}, F_{\mathrm{PFG}}, F_{\mathrm{PJG}}, C_{\mathrm{SF}}$ are replaced by $\Omega_{\mathrm{A}^{(\mathrm{P}_2)}}, \Omega_{\mathrm{G}^{(\mathrm{P}_2)}}, \Omega_{\mathrm{C}^{(\mathrm{P}_2)}}, \Omega_{\mathrm{SP}^{(\mathrm{P}_2)}}, \Omega_{\mathrm{PG}^{(\mathrm{P}_2)}}, F_{\mathrm{T}^{(\mathrm{P}_2)}}, F^{\mathrm{Embed}}_{\mathrm{SP}^{(\mathrm{P}_2)}}, F^{\mathrm{Reuse}}_{\mathrm{SP}^{(\mathrm{P}_2)}}, F^{\mathrm{Ref}}_{\mathrm{SP}^{(\mathrm{P}_2)}}, F_{\mathrm{PFG}^{(\mathrm{P}_2)}}, F_{\mathrm{PJG}^{(\mathrm{P}_2)}}, C_{\mathrm{SF}^{(\mathrm{P}_2)}}$,
- $F_{\mathrm{PG}^{(\mathrm{P}_2)}} = \bigcup_{i \in \Gamma_{\mathrm{PG}}} F_{i^{(\mathrm{P}_2)}}$,
- $F_{\mathrm{G}^{(\mathrm{P}_2)}} = \bigcup_{i \in \{\mathrm{XG,IG,CG}\}} F_i \cup F_{\mathrm{PG}^{(\mathrm{P}_2)}}$,
- $F_{\mathrm{T}^{(\mathrm{P}_2)}} = F_{\mathrm{T}^{(\mathrm{P}_1)}} \backslash FF_{\mathrm{T}^{(\mathrm{P}_1)}}$,
- $F^{\mathrm{Embed}}_{\mathrm{SP}^{(\mathrm{P}_2)}} = F^{\mathrm{Embed}}_{\mathrm{SP}^{(\mathrm{P}_1)}} \backslash FF^{\mathrm{Embed}}_{\mathrm{SP}^{(\mathrm{P}_1)}} \cup \{SP_1\}$,
- $F^{\mathrm{Reuse}}_{\mathrm{SP}^{(\mathrm{P}_2)}} = F^{\mathrm{Reuse}}_{\mathrm{SP}^{(\mathrm{P}_1)}} \backslash FF^{\mathrm{Reuse}}_{\mathrm{SP}^{(\mathrm{P}_1)}}$,
- $F^{\mathrm{Ref}}_{\mathrm{SP}^{(\mathrm{P}_2)}} = F^{\mathrm{Ref}}_{\mathrm{SP}^{(\mathrm{P}_1)}} \backslash FF^{\mathrm{Ref}}_{\mathrm{SP}^{(\mathrm{P}_1)}}$,
- P_1 is an embedded subprocess,
- $(FO_1, SP_1), (SP_1, FO_2) \in C_{\mathrm{SF}^{(\mathrm{P}_2)}}$,
- $CC_{\mathrm{SF}^{(\mathrm{P}_2)}} = \{(FO_1, SP_1), (SP_1, FO_2)\}$,
- $F_{\mathrm{PFG}^{(\mathrm{P}_2)}} = F_{\mathrm{PFG}^{(\mathrm{P}_1)}} \backslash \{G_1\}$,
- $F_{\mathrm{PJG}^{(\mathrm{P}_2)}} = F_{\mathrm{PJG}^{(\mathrm{P}_1)}} \backslash \{G_2\}$,
- $C_{\mathrm{SF}^{(\mathrm{P}_2)}} = C_{\mathrm{SF}^{(\mathrm{P}_1)}} \backslash CC_{\mathrm{SF}^{(\mathrm{P}_1)}} \cup CC_{\mathrm{SF}^{(\mathrm{P}_2)}}$ and

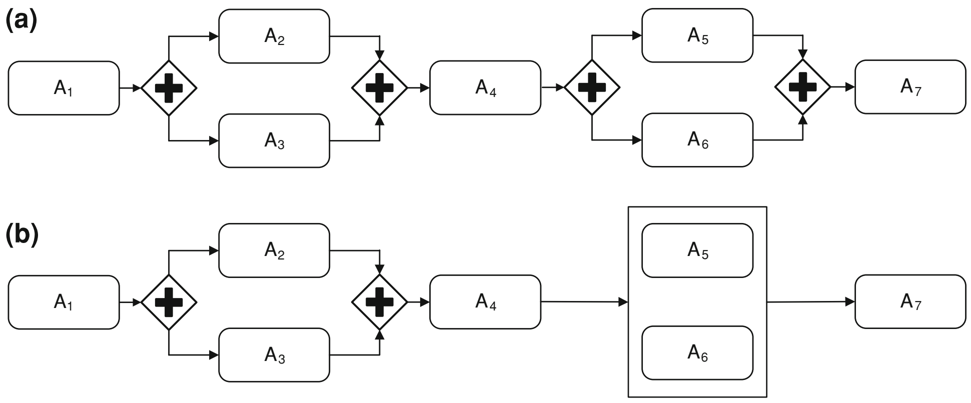


Fig. 7 Impl-PG-equivalent BPMN processes

– $\Phi_{\mathrm{NP}}(\mathrm{SP}_1) = P_3$ which is a process where

(i) $F_{\mathrm{T}^{(P_3)}} = FF_{\mathrm{T}^{(P_1)}}$
(ii) $F^k_{\mathrm{SP}^{(P_3)}} = FF^k_{\mathrm{SP}^{(P_1)}}$
(iii) $F_{\mathrm{A}^{(P_3)}} = F_{\mathrm{T}^{(P_3)}} \cup \bigcup_{i\in\Gamma_{\mathrm{SP}}} F^i_{\mathrm{SP}^{(P_3)}}$
(iv) $A_i \in F_{\mathrm{A}^{(P_3)}}$
(v) $\bigwedge_{i\in\Gamma_{\mathrm{SE}}} F^i_{\mathrm{SE}^{(P_3)}} = \emptyset$
(vi) $\bigwedge_{i\in\Gamma_{\mathrm{EE}}} F^i_{\mathrm{EE}^{(P_3)}} = \emptyset$
(vii) $\bigwedge_{i\in(\Gamma_{\mathrm{IE}}\cup\Gamma_{\overline{\mathrm{IE}}})} F^i_{\mathrm{IE}^{(P_3)}} = \emptyset$
(viii) $\bigwedge_{i\in\{\mathrm{D,E}\}} \bigwedge_{j\in\Gamma_{\mathrm{XG}}} F^i_{j^{(P_3)}} = \emptyset$
(ix) $\bigwedge_{i\in\Gamma_{\mathrm{IG}}} F_{i^{(P_3)}} = \emptyset$
(x) $\bigwedge_{i\in\Gamma_{\mathrm{CG}}} F_{i^{(P_3)}} = \emptyset$
(xi) $\bigwedge_{i\in\Gamma_{\mathrm{PG}}} F_{i^{(P_3)}} = \emptyset$
(xii) $A_{\mathrm{DO}^{(P_3)}} = \emptyset$
(xiii) $C_{\mathrm{SF}^{(P_3)}} = \emptyset$
(xiv) $C_{\mathrm{DA}^{(P_3)}} = \emptyset$
(xv) $S_{\mathrm{Cond}^{(P_3)}} = \emptyset$

for $i \in 1,\dots,n$ and $k \in \Gamma_{\mathrm{SP}}$.

Definition 21 specifies that a BPMN process consisting of a set of concurrent activities enclosed within an embedded subprocess is substitutable for a BPMN process comprising the set of concurrent activities connected to a pair of parallel fork gateway and parallel join gateway. The former provides a more compact representation of the latter through the use of a hierarchical structure. The deletion of the parallel fork gateway G_1 is defined by $F_{\mathrm{PFG}^{(P_2)}}$, whereas the removal of the parallel join gateway G_2 is specified by $F_{\mathrm{PJG}^{(P_2)}}$.

Definition 22 (*Impl-PG-equivalence*) For any BPMN processes P_1 and P_2, P_1 is impl-PG-equivalent to P_2, denoted by $P_1 \approx^{\mathrm{Impl}}_{\mathrm{PG}} P_2$, if and only if there is a BPMN process P_3 such that P_3 is an implicit-PG form of P_1 and P_2.

Definition 22 is a principle that states when two BPMN processes are equal in terms of implicit-PG form.

Proposition 5 *The relation* $\approx^{\mathrm{Impl}}_{\mathrm{PG}}$ *is transitive.*

Proof By similar argument as Proposition 1. □

Proposition 6 *The relation* $\approx^{\mathrm{Impl}}_{\mathrm{PG}}$ *is an equivalence.*

Proof Analogous to Proposition 2. □

Figure 7 is an illustration on implicit-PG-equivalence. Figure 7b improves the visual clarity of Fig. 7a by removing a pair of parallel fork gateway and parallel join gateway as well as enclosing the activities A_5 and A_6 within an embedded subprocess.

Definition 23 (*Activity-based-implicit-PFG form*) Let P_1 be a process with Ω_{G}, Ω_{C}, Ω_{PG}, F_{PFG}, C_{SF} are replaced by $\Omega_{\mathrm{G}^{(P_1)}}, \Omega_{\mathrm{C}^{(P_1)}}, \Omega_{\mathrm{PG}^{(P_1)}}, F_{\mathrm{PFG}^{(P_1)}}, C_{\mathrm{SF}^{(P_1)}}, F_{\mathrm{PG}^{(P_1)}} = F_{\mathrm{PFG}^{(P_1)}} \cup F_{\mathrm{PJG}}$, $F_{\mathrm{G}^{(P_1)}} = \bigcup_{i\in\{\mathrm{XG,IG,CG}\}} F_i \cup F_{\mathrm{PG}^{(P_1)}}$ and $S_{\mathrm{F}^{(P_1)}} = F_{\mathrm{G}^{(P_1)}} \cup \bigcup_{i\in\{\mathrm{E,A}\}} F_i$. If $FO_i \in S_{\mathrm{F}^{(P_1)}}$, $A_1 \in F_{\mathrm{A}}$, $G_1 \in F_{\mathrm{PFG}^{(P_1)}}$, $(FO_1, A_1), (A_1, G_1), (G_1, FO_2), (G_1, FO_3), \dots, (G_1, FO_n) \in C_{\mathrm{SF}^{(P_1)}}$, $CC_{\mathrm{SF}^{(P_1)}} = \{(A_1, G_1), (G_1, FO_2), (G_1, FO_3), \dots, (G_1, FO_n)\}$ for $i = 1, \dots, n$, then there is a unique process P_2 which is in activity-based-implicit-PFG form such that

– $\Omega_{\mathrm{G}}, \Omega_{\mathrm{C}}, \Omega_{\mathrm{PG}}, F_{\mathrm{PFG}}, C_{\mathrm{SF}}$ are replaced by $\Omega_{\mathrm{G}^{(P_2)}}, \Omega_{\mathrm{C}^{(P_2)}}, \Omega_{\mathrm{PG}^{(P_2)}}, F_{\mathrm{PFG}^{(P_2)}}, C_{\mathrm{SF}^{(P_2)}}$,
– $F_{\mathrm{PG}^{(P_2)}} = F_{\mathrm{PFG}^{(P_2)}} \cup F_{\mathrm{PJG}}$,
– $F_{\mathrm{G}^{(P_2)}} = \bigcup_{i\in\{\mathrm{XG,IG,CG}\}} F_i \cup F_{\mathrm{PG}^{(P_2)}}$,
– $S_{\mathrm{F}^{(P_2)}} = F_{\mathrm{G}^{(P_2)}} \cup \bigcup_{i\in\{\mathrm{E,A}\}} F_i$,
– $FO_i \in S_{\mathrm{F}^{(P_2)}}$,
– $A_1 \in F_{\mathrm{A}}$,
– $(FO_1, A_1), (A_1, FO_2), (A_1, FO_3), \dots, (A_1, FO_n) \in C_{\mathrm{SF}^{(P_2)}}$,
– $CC_{\mathrm{SF}^{(P_2)}} = \{(A_1, FO_2), (A_1, FO_3), \dots, (A_1, FO_n)\}$,
– $F_{\mathrm{PFG}^{(P_2)}} = F_{\mathrm{PFG}^{(P_1)}} \backslash \{G_1\}$ and
– $C_{\mathrm{SF}^{(P_2)}} = C_{\mathrm{SF}^{(P_1)}} \backslash CC_{\mathrm{SF}^{(P_1)}} \cup CC_{\mathrm{SF}^{(P_2)}}$

for $i = 1, \dots, n$.

Definition 23 says that an activity with multiple outgoing sequence flows is an alternative representation of using a sequence flow for connecting the activity to a parallel fork gateway with multiple outgoing sequence flows.

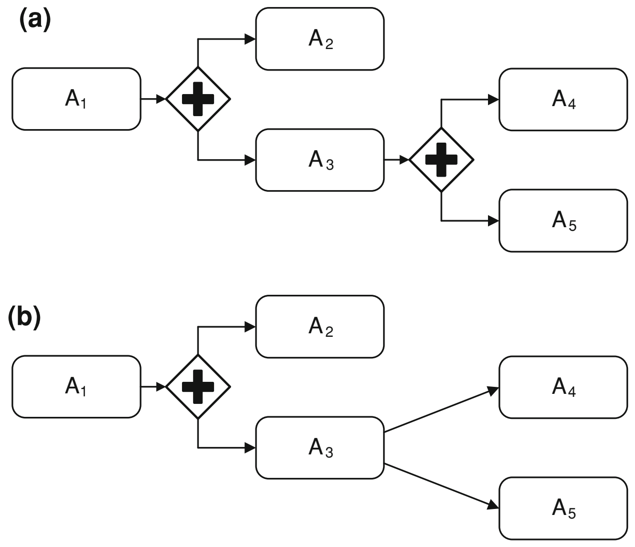


Fig. 8 AImpl-PFG-equivalent BPMN processes

Definition 24 (*AImpl-PFG-equivalence*) For any BPMN processes P_1 and P_2, P_1 is AImpl-PFG-equivalent to P_2, denoted by $P_1 \approx_{\text{PFG}}^{\text{AImpl}} P_2$, if and only if there is a BPMN process P_3 such that P_3 is an activity-based-implicit-PFG form of P_1 and P_2.

Proposition 7 *The relation* $\approx_{\text{PFG}}^{\text{AImpl}}$ *is transitive.*

Proof By similar argument as Proposition 1. □

Proposition 8 *The relation* $\approx_{\text{PFG}}^{\text{AImpl}}$ *is an equivalence.*

Proof Analogous to Proposition 2. □

Figure 8 depicts two BPMN processes which are AImpl-PFG-equivalent.

Definition 25 (*CDG-implicit-start-event form*) Let P_1 be a process with Ω_{E}, Ω_{C}, Ω_{SE}, $F_{\text{SE}}^{\text{None}}$, C_{SF} are replaced by $\Omega_{\text{E}^{(\text{P}_1)}}$, $\Omega_{\text{C}^{(\text{P}_1)}}$, $\Omega_{\text{SE}^{(\text{P}_1)}}$, $F_{\text{SE}^{(\text{P}_1)}}^{\text{None}}$, $C_{\text{SF}^{(\text{P}_1)}}$, $F_{\text{SE}^{(\text{P}_1)}} = F_{\text{SE}^{(\text{P}_1)}}^{\text{None}} \cup \bigcup_{i\in\Gamma_{\text{SE}}\setminus\{\text{None}\}} F_{\text{SE}}^{i}$, $F_{\text{E}^{(\text{P}_1)}} = F_{\text{SE}^{(\text{P}_1)}} \cup \bigcup_{i\in\{\text{EE,IE}\}} F_i$ and $S_{\text{F}^{(\text{P}_1)}} = F_{\text{E}^{(\text{P}_1)}} \cup \bigcup_{i\in\{\text{G,C}\}} F_i$. If $FO_i \in S_{\text{F}^{(\text{P}_1)}}$, $E_1 \in F_{\text{SE}^{(\text{P}_1)}}^{\text{None}}$, $G_1 \in F_{\text{CDG}}$, (E_1, G_1), (G_1, FO_1), (G_1, FO_2), …, $(G_1, FO_n) \in C_{\text{SF}^{(\text{P}_1)}}$ for $i = 1, \ldots, n$, then there is a unique process P_2 which is in CDG-implicit-start-event form such that

- Ω_{E}, Ω_{C}, Ω_{SE}, $F_{\text{SE}}^{\text{None}}$, C_{SF} are replaced by $\Omega_{\text{E}^{(\text{P}_2)}}$, $\Omega_{\text{C}^{(\text{P}_2)}}$, $\Omega_{\text{SE}^{(\text{P}_2)}}$, $F_{\text{SE}^{(\text{P}_2)}}^{\text{None}}$, $C_{\text{SF}^{(\text{P}_2)}}$,
- $F_{\text{SE}^{(\text{P}_2)}} = F_{\text{SE}^{(\text{P}_2)}}^{\text{None}} \cup \bigcup_{i\in\Gamma_{\text{SE}}\setminus\{None\}} F_{\text{SE}}^{i}$,
- $F_{\text{E}^{(\text{P}_2)}} = F_{\text{SE}^{(\text{P}_2)}} \cup \bigcup_{i\in\{\text{EE,IE}\}} F_i$,
- $S_{\text{F}^{(\text{P}_2)}} = F_{\text{E}^{(\text{P}_2)}} \cup \bigcup_{i\in\{\text{G,C}\}} F_i$,
- $FO_i \in S_{\text{F}^{(\text{P}_2)}}$,
- $G_1 \in F_{\text{CDG}}$,
- $(G_1, FO_1), (G_1, FO_2), \ldots, (G_1, FO_n) \in C_{\text{SF}^{(\text{P}_2)}}$,
- $F_{\text{SE}^{(\text{P}_2)}}^{\text{None}} = F_{\text{SE}^{(\text{P}_1)}}^{\text{None}} \setminus \{E_1\}$ and
- $C_{\text{SF}^{(\text{P}_2)}} = C_{\text{SF}^{(\text{P}_1)}} \setminus \{(E_1, G_1)\}$

for $i = 1, \ldots, n$.

Removing a none start event which precedes a complex decision gateway with multiple outgoing sequence flows, a respective BPMN process which is in CDG-implicit-start-event form is obtained.

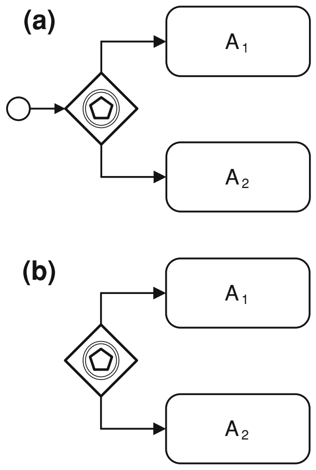


Fig. 9 CDG-ImplSE-equivalent BPMN processes

Definition 26 (*CDG-ImplSE-equivalence*) For any BPMN processes P_1 and P_2, P_1 is CDG-ImplSE-equivalent to P_2, denoted by $P_1 \approx_{\text{ImplSE}}^{\text{CDG}} P_2$, if and only if there is a BPMN process P_3 such that P_3 is a CDG-implicit-start-event form of P_1 and P_2.

Proposition 9 *The relation* $\approx_{\text{ImplSE}}^{\text{CDG}}$ *is transitive.*

Proof By similar argument as Proposition 1. □

Proposition 10 *The relation* $\approx_{\text{ImplSE}}^{\text{CDG}}$ *is an equivalence.*

Proof Analogous to Proposition 2. □

Figure 9 delineates two BPMN processes in which the BPMN process in Fig. 9a is CDG-ImplSE-equivalent to the BPMN processes in Fig. 9b.

Definition 27 (*DXDG-default-SF form*) Let P_1 be a process with Ω_{C}, S_{Cond} are replaced by $\Omega_{\text{C}^{(\text{P}_1)}}$, $S_{\text{Cond}^{(\text{P}_1)}}$. If $FO_i \in S_{\text{F}}$, $G_1 \in F_{\text{XDG}}^{\text{D}}$, $c_1, c_2, \ldots, c_{n-2}$, $\neg(c_1 \vee c_2 \vee \ldots \vee c_{n-2}) \in S_{\text{Cond}}$, (FO_1, G_1), (G_1, FO_2), (G_1, FO_3), …, (G_1, FO_{n-1}), $(G_1, FO_n) \in C_{\text{SF}}$, $\Phi_{\text{Cond}}((G_1, FO_j)) = c_{j-1}$, $\Phi_{\text{Cond}}((G_1, FO_n)) = \neg(c_1 \vee c_2 \vee \ldots \vee c_{n-2})$ for $i = 1, \ldots, n$ and $j = 2, \ldots, n-1$, then there is a unique process P_2 which is in DXDG-default-SF form such that

- Ω_{C}, S_{Cond} are replaced by $\Omega_{\text{C}^{(\text{P}_2)}}$, $S_{\text{Cond}^{(\text{P}_2)}}$,

- $FO_i \in S_F$,
- $G_1 \in F^{D}_{XDG}$,
- $c_1, c_2, \ldots, c_{n-2} \in S_{Cond}$,
- $(FO_1, G_1), (G_1, FO_2), (G_1, FO_3), \ldots, (G_1, FO_{n-1}), (G_1, FO_n) \in C_{SF}$,
- $\Phi_{Cond}((G_1, FO_j)) = c_{j-1}$,
- $\Phi_{IsDf}((G_1, FO_j)) =$ true and
- $S_{Cond^{(P_2)}} = S_{Cond^{(P_1)}} \backslash \{\neg(c_1 \vee c_2 \vee \ldots \vee c_{n-2})\}$

for $i = 1, \ldots, n$ and $j = 2, \ldots, n-1$.

A data-based exclusive decision gateway with $n-2$ outgoing sequence flows associated with conditions $c_1, c_2, \ldots, c_{n-2}$ and a default flow is another way of expressing a data-based exclusive decision gateway with $n-1$ outgoing sequence flows associated with conditions $c_1, c_2, \ldots, c_{n-2}$ and $\neg(c_1 \vee c_2 \vee \ldots \vee c_{n-2})$. This is stated in Definition 27.

Definition 28 (*DXDG-DefSF-equivalence*) For any BPMN processes P_1 and P_2, P_1 is DXDG-DefSF-equivalent to P_2, denoted by $P_1 \approx^{DXDG}_{DefSF} P_2$, if and only if there is a BPMN process P_3 such that P_3 is a DXDG-default-SF form of P_1 and P_2.

Proposition 11 *The relation* $\approx^{DXDG}_{DefSF}$ *is transitive.*

Proof By similar argument as Proposition 1. □

Proposition 12 *The relation* $\approx^{DXDG}_{DefSF}$ *is an equivalence.*

Proof Analogous to Proposition 2. □

We illustrate the concept of DXDG-DefSF-equivalence with an example (see Fig. 10). The BPMN processes in Fig. 10a and b are DXDG-DefSF-equivalent.

Definition 29 (*IDG-free form*) Let P_1 be a process with Ω_G, Ω_C, Ω_{IG}, F_{IDG}, C_{SF} are replaced by $\Omega_{G^{(P_1)}}$, $\Omega_{C^{(P_1)}}$, $\Omega_{IG^{(P_1)}}$, $F_{IDG^{(P_1)}}$, $C_{SF^{(P_1)}}$, $F_{IG^{(P_1)}} = F_{IDG^{(P_1)}} \cup F_{IMG}$, $F_{G^{(P_1)}} = \bigcup_{i\in\{XG,CG,PG\}} F_i \cup F_{IG^{(P_1)}}$, $S_{F^{(P_1)}} = F_{G^{(P_1)}} \cup \bigcup_{i\in\{E,A\}} F_i$. If $FO_i \in S_{F^{(P_1)}}$, $A_1 \in F_A$, $G_1 \in F_{IDG^{(P_1)}}$, (A_1, G_1), (G_1, FO_1), (G_1, FO_2), ..., $(G_1, FO_n) \in C_{SF^{(P_1)}}$, $\Phi_{Cond}((G_1, FO_i)) = c_i$, $CC_{SF^{(P_1)}} = \{(A_1, G_1), (G_1, FO_1), (G_1, FO_2), \ldots, (G_1, FO_n)\}$ for $i = 1, \ldots, n$, then there is a unique process P_2 which is in IDG-free form such that

- Ω_G, Ω_C, Ω_{IG}, F_{IDG}, C_{SF} are replaced by $\Omega_{G^{(P_2)}}$, $\Omega_{C^{(P_2)}}$, $\Omega_{IG^{(P_2)}}$, $F_{IDG^{(P_2)}}$, $C_{SF^{(P_2)}}$,
- $F_{IG^{(P_2)}} = F_{IDG^{(P_2)}} \cup F_{IMG}$,
- $F_{G^{(P_2)}} = \bigcup_{i\in\{XG,CG,PG\}} F_i \cup F_{IG^{(P_2)}}$,
- $S_{F^{(P_2)}} = F_{G^{(P_2)}} \cup \bigcup_{i\in\{E,A\}} F_i$,
- $FO_i \in S_{F^{(P_2)}}$,
- $A_1 \in F_A$,
- $(A_1, FO_1), (A_1, FO_2), \ldots, (A_1, FO_n) \in C_{SF^{(P_2)}}$,
- $CC_{SF^{(P_2)}} = \{(A_1, FO_1), (A_1, FO_2), \ldots, (A_1, FO_n)\}$,

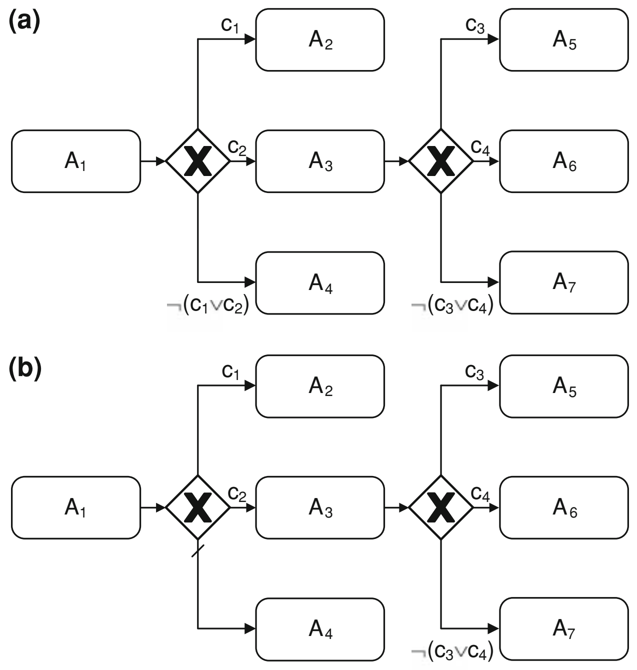


Fig. 10 DXDG-DefSF-equivalent BPMN processes

- $\Phi_{Cond}((A_1, FO_i)) = c_i$,
- $F_{IDG^{(P_2)}} = F_{IDG^{(P_1)}} \backslash \{G_1\}$ and
- $C_{SF^{(P_2)}} = C_{SF^{(P_1)}} \backslash CC_{SF^{(P_1)}} \cup CC_{SF^{(P_2)}}$

for $i = 1, \ldots, n$.

Definition 29 describes an activity which connects to an inclusive decision gateway with multiple outgoing sequence flows associated with conditions can be represented as an activity with multiple outgoing conditional sequence flows.

Definition 30 (*IDG-F-equivalence*) For any BPMN processes P_1 and P_2, P_1 is IDG-F-equivalent to P_2, denoted by $P_1 \approx^{IDG}_{F} P_2$, if and only if there is a BPMN process P_3 such that P_3 is an IDG-free form of P_1 and P_2.

Proposition 13 *The relation* $\approx^{IDG}_{F}$ *is transitive.*

Proof By similar argument as Proposition 1. □

Proposition 14 *The relation* $\approx^{IDG}_{F}$ *is an equivalence.*

Proof Analogous to Proposition 2. □

A pair of BPMN processes that are IDG-F-equivalent is given in Fig. 11.

With the theory of equivalences for BPMN processes in place, the rest of this section is concerned with the use of an elaborated example to demonstrate the practical application of the proposed equivalences. We consider two structurally different BPMN processes as shown in Figs. 12 and 13.

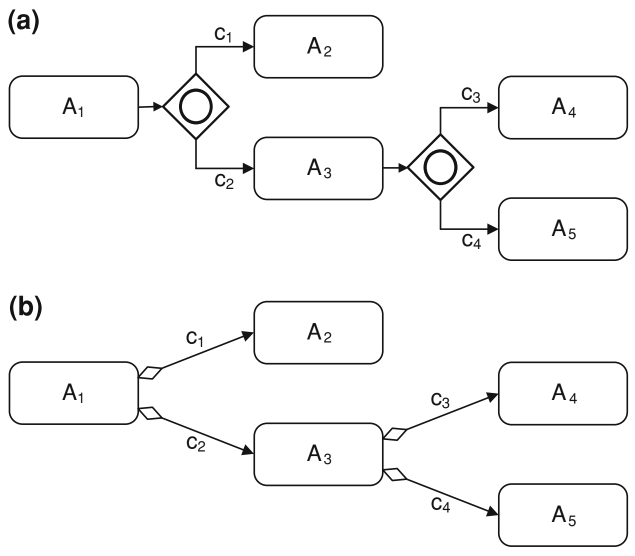


Fig. 11 IDG-F-equivalent BPMN processes

BPMN process 1 (Fig. 12) contains a number of notational elements that can be transformed into equivalent representations by repeated applications of Definitions 17, 19, 21, 23, 25, 27 and 29. These encompass the none start event, data-based exclusive decision gateways associated with conditions $\neg(c_1 \vee c_2)$ and $\neg(c_8 \vee c_9)$, inclusive decision gateways, parallel fork gateways, data-based exclusive decision gateway followed by the duplicate activity A_{24} and data-based exclusive merge gateway preceded with the duplicate activity A_{27}.

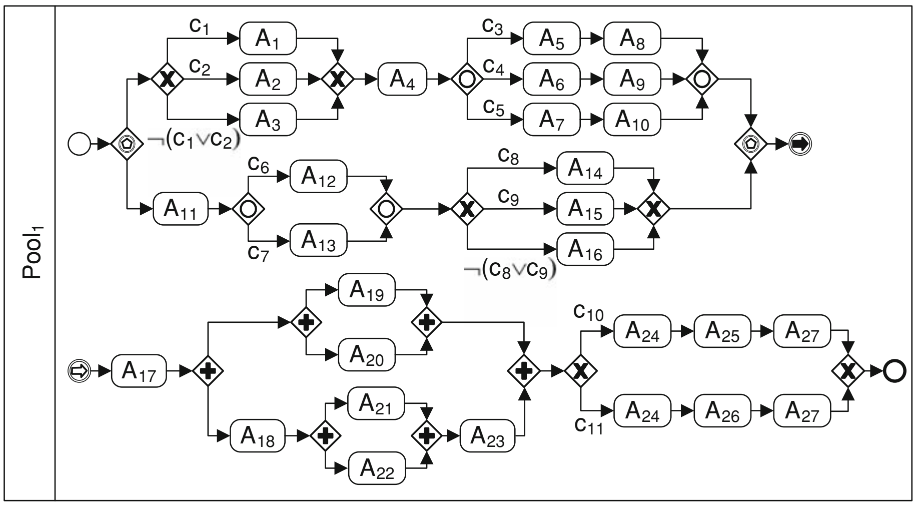
Likewise, BPMN process 2 (Fig. 13) embodies graphical constructs that can be represented as equivalent forms. These comprise the inclusive decision gateway with an incoming sequence flow from the activity A_{11}, the data-based exclusive decision gateway associated with condition $\neg(c_1 \vee c_2)$, the parallel fork gateway with an incoming sequence flow from the activity A_{17} as well as the data-based exclusive decision gateway followed by the duplicate activity A_{24}.

To determine whether BPMN processes 1 and 2 exhibit the same behaviour, we need to show that through a sequence of transformations they are both capable of converting into the same representation in DXMG-based-duplicate-free form, DXDG-based-duplicate-free form, implicit-PG form, activity-based-implicit-PFG form, CDG-implicit-start-event form, DXDG-default-SF form and IDG-free form. Consider the none start event in BPMN process 1 (Fig. 12). Applying Definition 25, the none start event is eliminated to yield an equivalent BPMN process in CDG-implicit-start-event form as depicted in Fig. 14. Replacing the conditions $\neg(c_1 \vee c_2)$ and $\neg(c_8 \vee c_9)$ of the data-based exclusive decision gateways with default flows, we get an equivalent BPMN process in DXDG-default-SF form in accordance to Definition 27. By substituting the activities A_4 and A_{11} with outgoing conditional sequence flows (Fig. 14) for the activities A_4 and A_{11} that connect to inclusive decision gateways with outgoing sequence flows associated with conditions (Fig. 12), an equivalent BPMN process in IDG-free form is obtained as specified in Definition 29.

The use of an embedded subprocess to enclose the activities A_{19} and A_{20} eliminates the corresponding pair of parallel fork gateway and parallel join gateway (Definition 21). The other two parallel fork gateways in Fig. 12 are removed through two successive applications of Definition 23 as shown in Fig. 14. An equivalent representation for BPMN process 1 as delineated in Fig. 14 is ultimately produced by using Definitions 19 and 17 to eliminate the duplicate activities A_{24} and A_{27} in Fig. 12.

In the same way, we apply Definitions 29 and 27 to BPMN process 2 (Fig. 13) for converting the inclusive decision gateway with an incoming sequence flow from the activity A_{11} and the data-based exclusive decision gateway associated with condition $\neg(c_1 \vee c_2)$ into the activity A_{11} with outgoing

Fig. 12 BPMN process 1



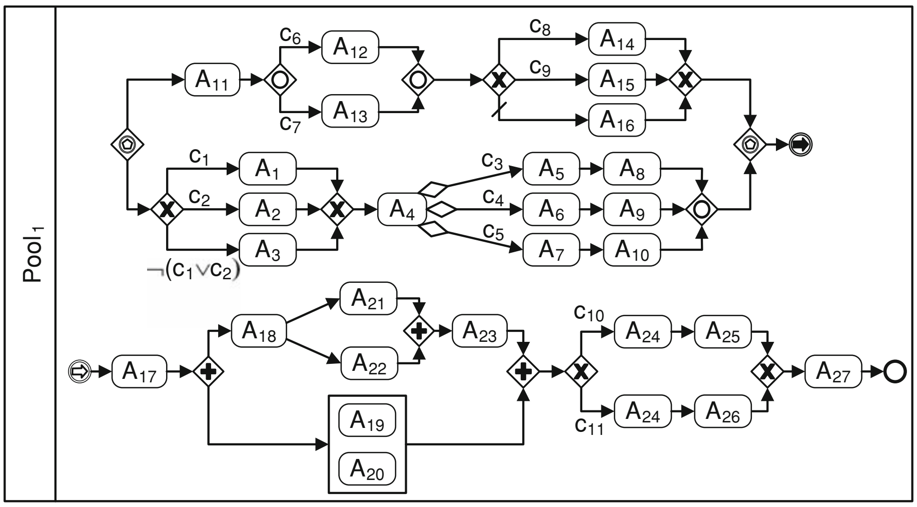


Fig. 13 BPMN process 2

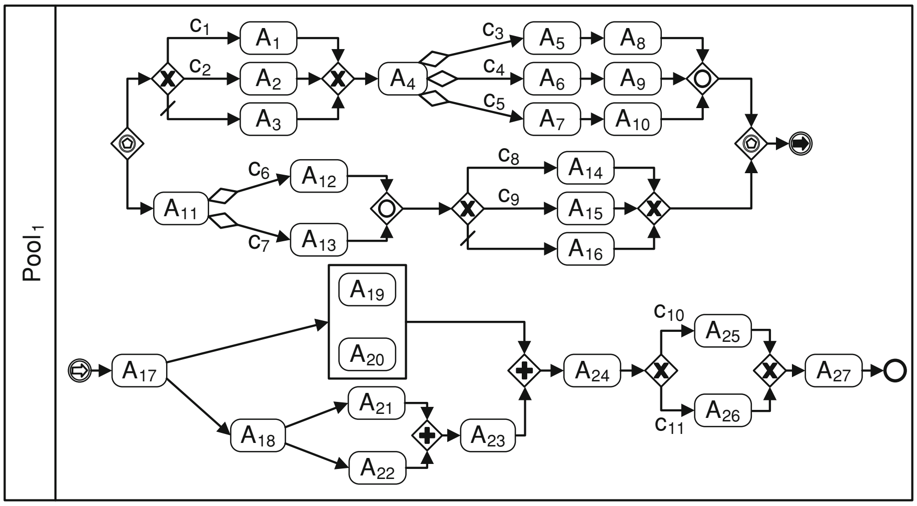


Fig. 14 An equivalent BPMN process for BPMN processes 1 and 2

conditional sequence flows and the data-based exclusive gateway with default flow as illustrated in Fig. 14. The elimination of the parallel fork gateway with an incoming sequence flow from the activity A_{17} and the duplicate activity A_{24} yields an equivalent representation for BPMN process 2 as depicted in Fig. 14. As there is a common representation for both BPMN processes 1 and 2, they are regarded as equivalent and can be used interchangeably.

6 Conclusions

This paper is positioned as part of a larger effort to explore the equivalences of business processes in the domain of business process management. Our earlier work is devoted to the classification of equivalences of UML ADs. In this study, it takes on the challenge of categorizing various types of equivalences for BPMN processes. To formalize the notions of equivalences, a formal foundation of BPMN processes has been developed. Based on this underlying model, seven kinds of equivalences have been proposed. These encompass DXMG-DF-equivalence, DXDG-DF-equivalence, Impl-PG-equivalence, AImpl-PFG-equivalence, CDG-ImplSE-equivalence, DXDG-DefSF-equivalence and IDG-F-equivalence. The theoretical framework contributes to the design of business processes through the application of mathematical techniques for

(i) the equivalence checking of BPMN processes;
(ii) the generation of alternative representations for BPMN processes; and
(iii) the simplification and restructuring of BPMN processes.

We intend to continue pursuing this line of investigation in a series of further studies. An extended discussion of other equivalences is among one of the many topics to be explored in future research.

References

1. OMG (2006) Business process modeling notation specification, February 2006. http://www.bpmn.org/. Accessed 28 December 2007
2. OMG (2008) Business process modeling notation, v1.1, January 2008. http://www.bpmn.org/. Accessed 7 January 2009
3. Havey M (2005) Essential business process modeling. O'Reilly Media, Inc., Sebastopol
4. Bog A, Puhlmann F, Weske M (2007) The PiVizTool: Simulating choreographies with dynamic binding. In: Demo session of the 5th international conference on business process management, 2007. http://bpt.hpi.uni-potsdam.de/pub/Public/FrankPuhlmann/bpm2007-piviztool.pdf. Accessed 17 February 2008
5. Bog A, Puhlmann F (2006) A tool for the simulation of π-calculus systems. In: Open.BPM 2006: Geschäftsprozessmanagement mit Open Source-Technologien, 2006. http://bpt.hpi.uni-potsdam.de/pub/Public/FrankPuhlmann/PiSimulator_openBPM.pdf. Accessed 9 January 2009
6. Bog A (2006) Introduction to the PiVizTool. Hasso Plattner Institute, University of Potsdam, 2006. http://bpt.hpi.uni-potsdam.de/pub/Piworkflow/Simulator/piviztool-intro.pdf. Accessed 13 January 2009
7. Bog A (2006) A visual environment for the simulation of business processes based on the pi-calculus. Master's thesis, Hasso Plattner Institute, University of Potsdam, 2006. http://bpt.hpi.uni-potsdam.de/pub/Public/FrankPuhlmann/AnjaBogThesisFinal.pdf. Accessed 13 January 2009
8. Puhlmann F (2007) Soundness verification of business processes specified in the pi-calculus. In: CoopIS 2007. LNCS, vol 4803, pp 6–23
9. Briais S (2005) The ABC User's Guide, 2005. http://lamp.epfl.ch/~sbriais/abc/abc_ug.pdf. Accessed 17 February 2008
10. Ou-Yang C, Lin YD (2008) BPMN-based business process model feasibility analysis: A Petri Net approach. Int J Prod Res 46(14):3763–3781
11. Ratzer AV, Wells L, Lassen HM, Laursen M, Qvortrup JF, Stissing MS, Westergaard M, Christensen S, Jensen K (2003) CPN tools for editing, simulating, and analysing coloured petri nets. In: ICATPN 2003. LNCS, vol 2679. Springer, Berlin, pp 450–462
12. Raedts I, Petkovic M, Usenko YS, van der Werf JM, Groote JF, Somers L (2007) Transformation of BPMN models for behaviour analysis. In: MSVVEIS 2007, pp 126–137
13. Dijkman RM, Dumas M, Ouyang C (2007) Formal semantics and analysis of BPMN process models using Petri Nets. Preprint (2007); available at http://eprints.qut.edu.au/archive/00007115/01/7115.pdf. Accessed 6 July 2008
14. Dijkman RM, Dumas M, Ouyang C (2007) Formal semantics and automated analysis of BPMN process models. Preprint (2007); available at http://eprints.qut.edu.au/archive/00006859/. Accessed 6 July 2008
15. Wong PYH, Gibbons J (2008) A process semantics for BPMN. In: Proceedings of 10th international conference on formal engineering methods. LNCS 5256, pp 355–374
16. Hoare CAR (1985) Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs
17. Formal Systems (Europe) Ltd. (2003) Failures-divergence refinement: FDR2 user manual, May 2003. http://www.fsel.com/fdr2_download.html. Accessed 20 January 2005
18. Gruber W (2003) Modelling and transformation of workflows with temporal constraints. PhD thesis, Vienna University of Technology, 2003. http://www.isys.uni-klu.ac.at/PDF/2003-0178-WLG.pdf. Accessed 13 January 2009
19. Eder J, Gruber W, Pichler H (2005) Transforming workflow graphs. In: First international conference on interoperability of enterprise software and applications, pp 23–25
20. Lam VSW (2008) Theory for classifying equivalences of UML activity diagrams. IET Softw J 2(5):391–403
21. Lam VSW Formal analysis of BPMN models: a NuSMV-based approach. (submitted)