Towards a General Framework for Business Process Modeling

Irfan Chishti
Department of Computing and Information Systems
University of Greenwich
London, United Kingdom

Abstract

This paper proposes grounding for the business process modeling (BPM) based on general time theory providing axiomatic system. First order logic is used to give a clear definition of abstract business process and corresponding temporal relations including derived relations using a single "Meets" relation. Temporal logic used here treats time interval and time point on equal footing. We use model theoretic approach, in which abstract business process is represented as a formal system and mapped to an instance/concrete realization. Also, we used resolution theorem to provide its soundness and completeness properties. Business Process Temporal Graph (BPTG) described as a directed graph is introduced with graphical notation defined to represent the temporal knowledge. Arcs representing time elements, vertex representing the 'Meets' relation and also allows expression of both logical AND and OR. However, a real world realization of the corresponding graph is considered an instance of an abstract business process. Reachability analysis is conducted to provide its soundness and completeness properties. The formal system provided is general enough to subsume the main BPM tools such as Business Process Modeling Notation (BPMN), Unified Modeling Language Activity Diagram (UML AD) and main charts i.e. flow charts used to model a business process.

1. Introduction

There is widely used business process modeling languages (BPMLs) whom semantics are "intuitive" but not formally founded. On the other hand, there is a variety of formal BPMLs, with operational semantics and provide concepts to specify aspects close to implementation. Former class of BPMLs support understandable models but lack formal semantics; and later class of BPMLs provide formal semantics but are often not accepted by the modellers and lack precision. There have been many attempts to bridge the gap between the two concerns specifically focusing on the need for a plain communication basis to provide unambiguous business process (BP) reasoning and representation.

Existing Workflow Management Systems (WFMSs) follow a pragmatic approach. They often use a proprietary modeling language with an intuitive graphical layout. The underlying semantics lack a formal foundation. As a consequence, analysis issues, such as proving correctness i.e. soundness and completeness, and reliable execution are not supported at design level. Failure detection is only possible while monitoring process execution. It is clear that failures that are only detected at run-time may be very costly and may cause all kinds of problems.

Many research approaches address this drawback and considerable efforts went into the formalization of informal business process modeling notation (BPMN), semi-formal modeling languages, such as Event-driven Process Chain (EPC) and Unified Modeling Language Activity Diagram (UML AD) but do not provide a solution. The formalization removes ambiguities and restricts the expressiveness. This leads to an academic approach available such as Petri nets which use graphical patterns replacing constructs of the primary BPML. The goal is to facilitate the understanding of the determined interpretation but lacks the degree of precision. Coming from either direction, the main idea is to define a comprehensive modeling approach which meets all requirements, i.e., to provide concepts that support the different levels of abstraction. As much as such a general language would simplify life, so far none of the proposed languages provide concepts to cover the different levels of abstraction separately, e.g. through a suitable formal temporal system [5].

Temporal facts can be expressed associated with points and intervals and generally be defined as *'A before B'* or *'A during B'* and so on. However, to reason and represent temporal facts a number of temporal theories are being used. The reason is that temporal logic provides consistency based on explicit axioms whereas business process modeling (BPM) deals with the practical problems from real life processes. But techniques used to model the business processes (BPs) produce some ambiguities and makes them cumbersome for modelers to handle too many different constructs/symbols while modeling their BPs. The reason is no logical foundation available for the concepts used for modeling by the practical techniques and if provided it could improve process understanding and its

representation with logical consistency. It is possible to use temporal logic to define unique meanings to the modeling terms used in the BPM discipline. It is the aim of this paper to provide precise meaning of modeling terms and structures of the BPM that can be based on unambiguous logical structure to bridge the gap by providing a plain communication method.

In the spectrum of temporal logic many temporal theories provided but always left some unanswered questions. In order to maintain knowledge about temporal intervals, Allen introduced a temporal theory based on intervals taken as primitive and their 13 qualitative temporal relationships in time [6]. Allen and Hayes introduced a temporal object called 'moment' i.e. an interval which cannot be broken down further, and used as an alternative to point [8]. This effort was made to avoid the problem in modeling time point and also to avoid the dividing instant problem [7] [12] [17]. However, Ma and Knight proposed a formal but general temporal theory which considers both time point and time interval on equal footing and will be used here to ground the BPM [9].

In this paper we will be using term business process (BP) and process interchangeably. In setting up a first order structure to express the abstractions of the BPM, we need to define a conceptual term to represent the activities, actions, tasks, events, sub-processes and processes of the BPM discipline. For this, first we need to understand BP, a BP is defined in the literature [15] [16], referring to a structure set of actions/tasks and events that has some start (input) and an end (goal), designed to show how work is done, rather than what is done. The actions/tasks are referred to that are usually work elements producing some component or subcomponent of a complete artefact. Actions/tasks are structured according to essential logical time ordering of component production. However, there is still a difference between meanings and use of the terms utilized in different BPMLs. In Adam Smith's famous pin factory process, actions are individual tasks that various workmen performed in the labour intensive manufacturing of a pin i.e. draw out wire, straight it, cut it, point it, …, and so on [3]. The entire BP may be thought of as a co-ordination in time of a set of tasks such as those of pin factory, together with some time events to synchronise with external activities, such as factory opening and closing times, or the arrival of a directive to cease production.

After examining the pin factory example, actions/tasks and events have occurred that brings ambiguity due to absence of formal meanings attached to the terms used in representation and modeling of a BP. In order to provide conceptualization and avoid confusion we introduce a conceptual term 'abstract business process', to represent the terms of the BPM discipline. Also formally define it using first order logic to provide consistent meanings based on axiomatic system. However, an abstract process occurs over a time element that can be associated with time interval or moment or point. If a process occurs over a time moment then it will represent an unit of work that must not be broken down into sub-units notated here as 'atomic process' as seen from pin factory example 'draw wire', cut wire' and so on. However, if a process occurs over a time point, it is considered to be a special process to represents the start and end of a process e.g. factory opening time. However, combination of atomic processes and special processes constitutes a BP that occurs over a time interval that can be broken down into smaller parts i.e. atomic processes.

In general, an abstract process that can be defined as a finite set of atomic process names, associated time elements and the corresponding duration. The duration distinct an abstract process with positive time duration that is associated with time moment is termed here as atomic process. Zero time duration that is associated with time point and considered as special process that starts and ends an abstract process. We also have noticed that pin factory example is comprised of 18 distinct atomic processes that form temporal relationship in some logical order and combines together to represent a BP [3]. To avoid, confusion, in this paper we refer atomic process associated with time moment considered as an unit of work i.e. task, action, process and sub-process if it cannot be broken down. However, a business process (BP) considered as combinations of unit of work elements associated with time interval that can broken down further and special process associated with time point refers to events.

To provide grounding for the BPM, we take a model theoretic approach, and define abstract BP based upon a first order structure of time [9]. In general, we take an abstract BP a non-empty set of collection of processes with their corresponding temporal relations. Probably the most important terminological difference between the model theoretic discipline and more practical disciplines, such as BPM is the meaning of the term 'model' itself. In practical modeling, the construction of a model is to construct a formal theory which describes and explains a system. So, for instance, Petri Net is a formal language which describes a process. Hence in this discipline, the model of a process is in fact a theory. In model theoretic terms, however, the formal Petri Net description is known as the theory, and a model is an interpretation of a theory expressed in the first order logic. However, in the field of modeling physical processes, the model means a structure to simulate the process, which may be run many times to simulate different instances of the physical process. In this case the model is more like a theory, which represents the essential common facts which apply as true to each of the instance of the model runs. To avoid confusion, in this paper we shall refer to the formal description of a BP as an abstract BP, and a concrete realization of it as an instance.

Section II provides temporal basis followed by formalization of BP and using resolution theorem to provide soundness and completeness properties of the formal system in section III. Section IV defines an abstract process model and maps to real world instances of the process. Section V provides graphical representation using Business process temporal graph (BPTG) of the abstract business model and provides analysis and comparison of some of the major BP tools, such as flow Charts, Pert Charts, Gant Charts and BPMN. Section VI provides an illustrative example together with reachability analysis to show the soundness and completeness of the formal system introduced, and conclusion is given in section VII.

2. Temporal Basis

For general treatment, in this paper we shall adopt the general time theory proposed, we take a nonempty set of processes and associated primitive time elements, with an immediate predecessor relation, '*Meets*', over time elements, and a duration assignment function, '*D(t)*', from time elements to non-negative real numbers [9]. If $D(t) = 0$, then t is called a point; otherwise $D(t) > 0$, and t is called an interval/moment. Intuitively, *Meets* (t_1, t_2) states that time element t_1 is one of the immediately predecessors of time element t_2, with no other time elements standing between them. On one hand, such a point-and-interval based time theory overcomes problematic puzzles such as the so-called Dividing Instant Problem [7] [12] [17]. On the other hand, it allows temporal references to both instantaneous phenomena with zero duration, and to periodic phenomena that last for some positive duration, and therefore no longer suffer from the limitations of Allen's interval based time theory as pointed out by Galton [1] [7].

The following conventions will be used throughout and a full range of connectives and quantifiers using their standard interpretation as:

- $\wedge$ Conjunction, $\vee$ Disjunction, $\neg$ Negation,
- $\Rightarrow$ Implication, $\Leftrightarrow$ Equivalence, $\neq$ not equal, $\vdash$ derivable, $\models$ logical entailment
- $\forall$Universal quantifier, and $\exists$ Existential quantifier

Analogous to the 13 relations introduced by Allen, Ma and knight extended the relationships on more general temporal theory considering point and interval on equal footing, accordingly 30 derived temporal relations over time elements including both time points and time intervals can be concluded, which can be derived from the single '*Meets*' order relation and defined as finite set of temporal relations given below which are classified into the following 4 groups [6] [9] [10]:

- *Relations relating a point to a point:* {Equal, Before, After}
- *Relations relating a point to an interval:* {Before, Meets, Starts, During, Finishes, Met-by, After}
- *Relations relating an interval to a point:*{Before, Meets, Started-by, Contains, Finished-by, Met-by, After}
- *Relations relating an interval to an interval:*{Equal, Before, Meets, Overlaps, Starts, During, Finishes, Finished-by, Contains, Started-by, Overlapped-by, Met-by, After}

In terms of the single '*Meets*' relation, other relative relations like those introduced, including Equal, Met_by, Before, After, Overlaps, Overlapped_by, Starts, Started_by, During, Contains, Finishes, Finished_by, etc. can be derively defined. For instance [9] [12]:

$$\text{Before}(t_1, t_2) \Leftrightarrow \exists t \in \mathbf{T}\ (\text{Meets}(t_1, t) \wedge \text{Meets}(t, t_2))$$

Detailed axiomatization with graphical representation of interval-and-point based time theory is given in table 1, that explains the occurrences of temporal objects with the respect to the four groups mentioned above.

The completeness of the 13 possible exclusive order relations between any two time elements can be simply characterized by a single axiom given below [9]:

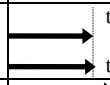
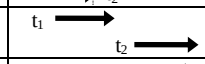
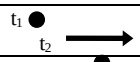
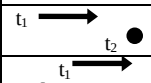
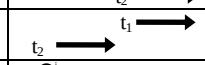
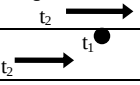
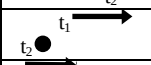
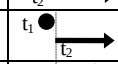
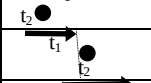
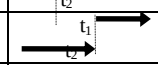
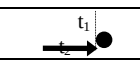
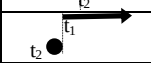
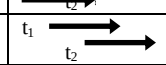
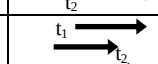
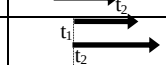
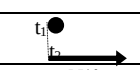
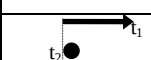
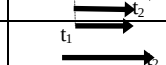
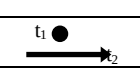
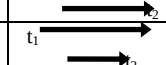
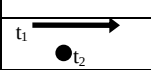
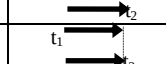
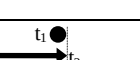
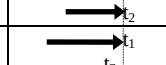
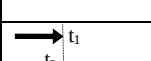
$$\forall t_1,t_2(\text{Equal}(t_1, t_2) \vee \text{Before}(t_1, t_2) \vee \text{After}(t_1, t_2) \vee \text{Meets}(t_1, t_2) \vee \text{Met-by}(t_1, t_2) \vee \text{Overlaps}(t_1, t_2) \vee \text{Overlapped-by}(t_1, t_2) \vee \text{Starts}(t_1, t_2) \vee \text{Started-by}(t_1, t_2) \vee \text{During}(t_1, t_2) \vee \text{Contains}(t_1, t_2) \vee \text{Finishes}(t_1, t_2) \vee \text{Finished-by}(t_1, t_2)) \quad \text{(Axiom 1)}$$

For the convenience of expression, Allen's non-exclusive relation 'In', which is defined for intervals alone, but extended by Ma and knight to accommodate both time intervals and points, using another temporal relation, 'Part' will be used here [7] [14]:

$$\text{In}(t_1,t_2) \Leftrightarrow \text{Starts}(t_1,t_2) \vee \text{During}(t_1,t_2) \vee \text{Finishes}(t_1, t_2) \quad \text{(R 1)}$$

$$\text{Part}(t_1, t_2) \Leftrightarrow \text{Equal}(t_1, t_2) \vee \text{In}(t_1, t_2) \quad \text{(R 2)}$$

Table 1. Derived temporal relations based on time point and time interval

Axioms Characterising Relations	Graphical Representation			
	Point t_1 to Point t_2	**Interval t_1 to Interval t_2**	**Point t_1 to Interval t_2**	**Interval t_1 to Point t_2**
Equal $(t_1, t_2) \Leftrightarrow \exists t',t'' \in \mathbf{T}$ (Meets $(t',t_1) \wedge$ Meets $(t',t_2) \wedge$ Meets $(t_1,t'') \wedge$ Meets $(t_2, t''))$			N/A	N/A
Before $(t_1, t_2) \Leftrightarrow \exists t \subset \mathbf{T}$ (Meets $(t_1, t) \wedge$ Meets (t, t_2))				
After $(t_1, t_2) \Leftrightarrow$ Before(t_2, t_1)				
Meets(t_1,t_2)	N/A		N/A	
Met-by $(t_1, t_2) \Leftrightarrow$Meets$(t_2,t_1)$	NA			
Overlaps $\Leftrightarrow \exists t,t_3,t_4 \in \mathbf{T}$ $(t_1 = t_3 \oplus t \wedge t_2 = t \oplus t_4)$	N/A		N/A	N/A
Overlapped-by $(t_1, t_2) \Leftrightarrow$ Overlaps(t_2, t_1)	N/A		N/A	N/A
Starts $(t_1, t_2) \Leftrightarrow \exists t \in \mathbf{T}$ $(t_2 = t_1 \oplus t)$	N/A			N/A
Started-by $(t_1, t_2) \Leftrightarrow$ Starts(t_2, t_1)	N/A		N/A	
During $(t_1, t_2) \Leftrightarrow \exists t_3,t_4 \in \mathbf{T}$ $(t_2 = t_3 \oplus t_1 \oplus t_4)$	N/A			N/A
Contains $(t_1, t_2) \Leftrightarrow$ During(t_2, t_1)	N/A		N/A	
Finishes $(t_1, t_2) \Leftrightarrow \exists t \in \mathbf{T}(t_2 = t \oplus t_1)$	N/A			N/A
Finished-by $(t_1, t_2) \Leftrightarrow$ Finishes(t_2,t_1)	N/A		N/A	

DERIVED TEMPORAL RELATIONS

Note: N/A stands for not applicable in the Table 1.

3. Abstract Business Process

Here we present a schema based on temporal basis introduced in section II for representing the idea of an abstract business process and provide core axiom to formally define it. Using predicate *Occurs* to represent an abstract process that occurs over a time element with a duration assignment and can be expressed as *Occurs (a, t, D(t))*. Where '*a*' stands for a process name, '*t*' represents time element with a duration assignment '*D(t)*'. In general, to provide axiomatization of an abstract process that applies to divisible intervals, non-divisible moments or time points is given in axiom 2 below using predicate *Occurs* and temporal relations R1 and R2:

$$Occurs(a, t, D(t)) \Rightarrow D(t) \geq 0 \wedge \forall t_1(In(t_1,t) \Rightarrow \exists t_2(Part(t_2,t_1) \wedge Occurs(a,t_2)))$$

(Axiom 2)

3.1. Definition 1: Atomic Process

An atomic process is the basic element of our abstract business process that may applies to non-divisible moments or time points and can be expressed as:

$$Occurs(a,t,D(t)) \Rightarrow \neg\exists(t_1 \wedge In(t_1,t) \wedge Occurs(a,t_1,D(t_1)))$$

Axiom 3)

Axiom 3 defines an atomic process that occurs over time moment. As stated in section I, atomic processes associated with time moments are referred to tasks, actions, sub-processes, and processes. However, an abstract process that is associated with a time point notated as special process and referred to event.

Now we will define business process, sub-process and the deduced temporal constraints to express the temporal relations between them.

3.2. Definition 2: Business Process

A *business process P* is defined as a pair (*A, R(A)*) where

$$A=\{Occurs(a_1,t_1,D(t_1)),\ldots Occurs(a_n,t_n,D(t_n))\} \quad \text{(Axiom 4)}$$

A is described as a finite set of abstract process names '*a*' with corresponding time elements '*t'* and associated duration assignments '*D(t)*'. However a business process (BP) defined here refers to activities, actions and processes that can be broken down further in the BPM discipline. Where a BP occurs over time interval '*i'* is decomposable. For instance, if there are two time elements 't_1' and 't_2' such that '$i = t_1 \oplus t_2$'; where 't_1' and 't_2' may refer to 2 atomic processes. The temporal relation between atomic processes is given as $R(A) = \{R(t_i, t_j) \mid 1 \leq i, j \leq n\}$.

The relation *R(A)* using '*Meets*' relation defines a BP of logical conjunction and disjunction. However, an abstract process can be taken as a special process, since for any abstract process '*a*',

$$R(\{a\}) = \emptyset \quad \text{(Axiom 5)}$$

3.3. Definition 3: Deduced Temporal Constraint

In this paper, for *R(A)* we use *DR(A)* to denote the deduced temporal constraint which contains all the relations plus all the other relations that can be derived from *R(A)*, and is given as:

$$DR(A) \vDash R(A) \quad \text{(Axiom 6)}$$

Let's prove it by deduction theorem,

1) Assumption 1: Every relation of DR(A) is also a relation of R(A). Let R{(a)} be any relation of DR(A), if R{(a}} is a relation of DR(A), then it follows that R{(a)} is also a relation of R(A). i.e. DR(A) ⊨ R(A).

2) Assumption 2: Let R{(a)} be any relation of R(A) that is also a relation of DR(A), i.e. DR(A) ⊨ R(A).

Axiom 6 is valid as it holds bidirectional and also there exists at least one transitive relation containing *R(A)*, and the disjunction of transitive relations is transitive. Hence the transitive closure of *DR(A)* is the disjunction of all transitive relations containing *R(A)*.

3.4. Properties of Abstract Business Process

Soundness and completeness are two major issues of a formal system in our case its abstract BP. While soundness refers to the correctness of the abstract process, completeness implicates that all the possible inferences can be derived by using the resolution algorithm in [2]. Formal definitions of these are presented here for convenience.

3) Definition 4: Abstract Business Process is Sound

An abstract BP is called *sound*, if any temporal relation *R(A)* has been proved from a set of deduced temporal constraint *DR(A)* by a proof procedure, such that

$$DR(A) \vdash R(A) \quad \text{(Axiom 7)}$$

it follows logically from *DR(A)*, i.e., Axiom 6 (DR(A) ⊨ R(A))

4) Definition 5: Abstract Business Process is Complete

An abstract BP is called *complete***,** if for any *R(A)*, that follows logically from a given set of deduced temporal constraint *DR(A)* and the proof procedure can prove *R(A)* , i.e. Axiom 7.

Now, we will follow a proof procedure presented and provide 2 theorems to prove the soundness and completeness of abstract model defined above [2].

a) Theorem I: Abstract Business Process is Sound

Proof: Given a set of deduced temporal constraints *DR(A)* and a goal *R(A)*. Suppose we obtain *R(A)* from *DR(A)* by the resolution theorem. We thus have Axiom 7. We want to prove that the derivation is logically sound.

Let us prove the theorem by the method of contradiction, presume that the consequent of Axiom 6, is false, which means $DR(A) \vDash \neg R(A)$. Thus, $\neg R(A)$ is satisfiable or true.

To satisfy, we assign truth values (true/false) to all temporal relations that are used in *R(A)*. We now claim that for such assignment, resolution of any two relations from *DR(A)* will be true. Thus the resulting temporal relation even after exhaustion of all possible temporal relations through resolution will not be false. Thus axiom 7 is a contradiction, which means $DR(A) \vdash \neg R(A)$.

Hence, the assumption $DR(A) \vDash \neg R(A)$ is false, and consequently axiom 6 holds, and proves that abstract BP is sound.

b) Theorem II: Abstract Business Process is Complete

Proof: Let *R(A)* be a goal such that from a given set of deduced temporal constraints *DR(A)*, then we have axiom 6. Where *R(A)* can be logically proven from *DR(A)*. We have to show there exists a proof procedure for *R(A)* i.e. axiom 7.

We shall prove it by the method of contradiction, let's assume axiom 7 is false that means $DR(A) \vdash \neg R(A)$. In other words, $R(A)$ is not derivable by a proof procedure from $DR(A)$.

Now we use *ground resolution theorem* as given in [2] that states "if a set of ground derived temporal relations (relations with no variables) is false, then the resolution closure of those relations contains the 'false' relation. Thus $\neg R(A)$, the resolution closure of $DR(A)$ yields the null relation, which causes a contradiction to axiom 7. Thus the assumption $DR(A) \vdash \neg R(A)$ is false. Hence, axiom 6 satisfies and proves that abstract BP is complete.

3.5. Definition 6: Sub Process

A BP $P_1 = (A_1, R(A_1))$ is called a sub-process of BP $P = (A, R(A))$, iff

$$A_1 \subseteq A$$
(Axiom 8)

$$DR(A_1) \subseteq DR(A)$$
(Axiom 9)

So, we can say that $A \Leftrightarrow (A \wedge (\neg A \vee A_1))$ and $DR(A) \Leftrightarrow (DR(A) \wedge (\neg DR(A) \vee DR(A_1)))$. From axiom 6, we can have $DR(A_1) \vDash .R(A_1)$. Now from axiom 8 and 9, we could say that $(A_1, R(A_1))$ is a sub-process of process $(A, R(A))$ i.e. P_1 is a sub-process of P.

In the next section we would provide a concrete realization of the abstract process model using a mapping function.

4. Mapping (Theory – Model)

So far, the definitions of atomic process, BP and sub-process given in section III, have been abstract. We may refer to the abstraction as *theory* and the interpretation as concrete world *realization*, or to the abstraction as *process type* and the interpretation as *process token*, or to the abstraction as *process class*, and the interpretation as *process instance*. However, we need to formally define the meaning of the relationship of theory to model, type to token, or process class to process instance.

In this paper, we follow the axiomatic method initiated first by David Hilbert in *Grundlagen der geometrie* [4]. That defines theory as an abstract axiomatic system, and a concrete realization as an interpretation of the theory in some real world domain. By an interpretation, we mean that a domain of real world can be chosen and constant individuals and predicates taken in such a way that, with this particular specification of the primitive elements of the theory and all axioms are true propositions. First of all, we introduce the concepts of abstract process instances, time instances and duration assignment instances as below:

We take the real world domain to be a set of abstract process instances A_R, containing processes names a with its occurring time instances t_R, and corresponding duration assignment instances $D(t_R)$. The function $D(t(a_R))$ into $D(t_R)$ is just the real duration assignment of the occurrences of the abstract processes. In this case the real world model is an instance of the abstract process model.

4.1. Definition 7: Abstract Process Instance

An *abstract process instance* is a triad of an process name, an instance of a time element, and a function of duration assignment that is $(a, t, D(t))$ where $a \in \mathbf{A}$, $t \in \mathbf{t_R}$ and $D(t) \geq 0$. We use a_i, a_{i1}, a_{i2}, …, etc., to denote abstract process instances A_R, where $a_i, a_{ij} \in \mathbf{A_R}$, j = 1, 2, …. For each abstract process instance a_R, we may write it as $a_R = [Name(a_R), t(a_R), D(t(a_R))]$.

where *Name* is a function from the set of process instances A_R to the set of process names A, t is a function from the set of process instances A_R to the set of time elements (instances) in t_R, $D(t)$ is a function from the set of process instances A_R to time instances t_R duration assignment, where $D(t) = 0$ represents the special processes i.e. events, and $D(t) > 0$ represents tasks, actions,activities, processes and sub-processes. In terms of reified temporal logic presented, we use temporal predicate *Occurs* to denote abstract process a_R occurs over time interval/moment/point (instances) t_R [11]:

$$\forall a_R \in A_R\, (Occurs(a_R, t(a_R), D(t(a_R)) \geq 0))$$
(Axiom 10)

Since each process instance is distinct, therefore, for any abstract process instance a_R, we impose that:

$$\forall a_R \in A_R\, \forall t \in t_R\, \forall D(t) \in D(t_R)\, (Occurs(a_R, t, D(t)) \Rightarrow t = t(a_R), D(t) = D(t(a_R)))$$
(Axiom 11)

Note that the existence of the real world interpretation ensures automatically the temporal consistency of the abstract model itself. To do this, we introduce the concept of mapping function from abstract to real world.

4.2. Definition 8: Mapping Function

We use mapping ϕ for interpretation of an abstract model to its corresponding instances from real world. We consider A_R a set of abstract process instances such that there exists a mapping ϕ between the set of processes 'A' in the abstract model and those in the instance/realisation. We denote this mapping as $\phi(A) \rightarrow A_R$.

However, a set of abstract processes is comprised of process names, occurring time

elements and their corresponding duration assignment. For convenience, we denote the mapping ϕ of a process element *p* as p_R, so that $\phi(p) = p_R$. Similarly, there exists mapping ϕ, for time instances t_R and duration assignment of corresponding insrances of time elements $D(t_R)$ to the time elements *t* and duration assignments *D(t)* in the abstract model and those in the instance, as $\phi(t) = t_R$, where

$$\forall t \in t_{R,}\ (D(t) = r) \Leftrightarrow D(\phi(t_R) = r) \in D(t_R)$$
(Axiom 12)

This mapping function allows us to define instances of a BP class and sub-process class as a realization of the axiomatic system presented in *Definition 2* and *6*. In the sub0section below, BP instances, sub-process instances and corresponding temporal relation instances defined using *Meets* relation.

4.3. Definition 9: Business Process(BP) Instance

A *BP instance* of an abstract BP *P (A, R(A))* is an actual realization of P_R $(A_R, R(A_R))$, i.e. $\phi(A) \rightarrow A_R$, shows the mapping of abstract process to real process.

However, $R(A_R)$ is a set of temporal relation instances such that there exists a mapping ϕ between the temporal relations *R(A)* in the abstract model and those in the instance. We denote the mapping as: $\phi(R(A)) \rightarrow R(A_R)$. We define $R(A_R)$ in the interpretation, as

$$\forall t_i, t_j \in t(Meets(t_i, t_j) \in R(A) \Leftrightarrow Meets(\phi(t_i), \phi(t_j)) \in R(A_R))$$
(Axiom 13)

For a concrete BP to be an instance of the abstract BP, we must be able to establish the mapping 'ϕ', identifying the abstract BPs from the concrete BPs, with real world times associated expressing their duration. But the real world processes must satisfy the same sequencing constraints as are specified in the abstract BP, i.e. $P_R = (A_R, R(A_R))$ must be temporally consistent.

4.4. Definition 10: Sub Process Instance

A *sub-process instance* of an abstract sub-process $P_1(A_1, R(A_1))$ is an actual realization of P_{R1} $(A_{R1}, R(A_{R1}))$, we derive from mapping function of $\phi(A)$, that shows $\phi(A_1) \rightarrow A_{R1}$.

However, $R(A_{R1})$ is a set of temporal relation instances such that there exists a mapping between the temporal relations $R(A_1)$ in the abstract model and those in the instance and denote the mapping as: $\phi(R(A_1)) \rightarrow R(A_{R1})$. We define $R(A_{R1})$ in the interpretation, as

$$\forall t', t'' \in t(Meets(t', t'') \in R(A_1) \Leftrightarrow Meets(\phi(t'), \phi(t'')) \in R(A_{R1})$$
(Axiom 14)

An abstract sub-process is a part of a parent abstract BP which is consistent such that there exists mapping ϕ of abstract sub-process to concrete real world sub-process and also must be temporally consistent that must satisfies the sequencing constraints as are specified in the abstract sub process, i.e. $P_{R1} = (A_{R1}, R(A_{R1}))$.

We are going to introduce business process temporal graph (BPTG) in the next section that can also be identified as valid in the concrete realization. For example, if a process p_1 occurs before process p_2 in the abstract model, then process instance p_{I1} will occur before p_{I2} in the corresponding concrete realization, with real world times associated with processes. The same applies to all the mapped processes in P_1.

5. Business Process Temporal Graph (BPTG)

In this section, we show the mapping of an abstract BP model to its corresponding case (instance/realization) is sound and complete using BPTG that is considered as a directed graph. However, an abstract BP *P* = (*A, R(A)*) has temporal reference, from axiom 4, we can write *A* as

$$A = \{p_1, \ldots\ldots, p_n\}$$
(Axiom 15)

Where *A* is a finite set of abstract processes, where, each process is represented as $p_i = Occurs\ (a, t, D(t))$. However, '*a*' is an abstract process name with corresponding occurrences of time element '*t*' expressing the knowledge involved; and $D(t)=\{D(t_i) = r_i \mid$ for some i where $1 \leq i \leq n\}$, and is a collection of duration assignments (possibly incomplete) to time element '*t*'**.** However, *R(A)* is the conjunction of disjunctions of *Meets* relations over '*t*'. In this paper we use BPTG and process temporal graph interchangeably.

The process temporal graph of a process case P_R is the same as that of abstract process *P*, i.e. connected with unique start and end vertices. These vertices correspond to the *process start* and *end instances* i.e. special processes (events). For any given process instance a_R, we accordingly term the two special processes as: $p_S(a_R)$ and $p_E(a_R)$. The deduced temporal constraint on any process and its corresponding special processes at the both extremes i.e. start and end, can be given as below:

$$\forall a_R \in A_R\ \exists p_S, p_E, t', t'' \in t_R\ (Occurs(p_S(a_R), t') \wedge Occurs(p_E(a_R), t'') \wedge (Meets(t', t(a_R)) \nabla Starts(t', t(a_R))) \wedge (Meets(t(a_R), t'') \nabla Finishes(t'', t(a_R))))$$
(Axiom 16)

Here, ∇ stands the logical "exclusive OR".

Axiom 16 characterizes the temporal relation between a process instance and its special processes instances i.e. start instance. Process start instance, either '*Meets*' the process instance, or 'Starts' the process instance; that is, either the process instance interval is open at its start point, or closed at its start point. Also, axiom 16 characterizes the temporal relation between an abstract process instance and its special process i.e. end instance. The process instance either "*Meets*" the process end instance, or the process end instance "Finishes" the process instance; that is, either the process instance interval is open at its end point, or closed at its end point.

Process temporal graph can be expressed in terms of a directed, partially weighted simple graph in which:

BPTG[1] Each time element is denoted as an arrowed-arc with a beginning-node and an ending node; and for time elements with known duration, the corresponding arcs are weighted by their durations respectively.

BPTG[2] The relation Meets(t_i, t_j) is presented by means of unifying the ending-node of time element t_i and the beginning-node of time element t_j. In other words, Meets(t_i, t_j) is denoted by the node structure where t_i is an in-arc and t_j is an out-arc of a same node, respectively.

BPTG[3] Logical expressions using "$\wedge$", "$\vee$" and combination of both using *Meets* relation are presented in fig 1. Where fig 1(a) defines t_i as an in-arc and t_j and t_k as two out-arcs of the same node. Fig 1(b) defines t_i and t_j as two in-arcs and t_k as out-arcs of the same node. Fig 1(c) defines t_i as duplicated identical out-arcs of a node, and defining one of the two t_is as an in-arc and t_j as an out-arc of another node; and defining the other t_i as as an in-arc and t_k as an out-arc of the third node. Fig 1(d) defines t_k as duplicated identical in-arc by defining t_i as an in-arc and one of the two t_ks as an out-arc of another node; and defining t_j as as an in-arc and the other t_k as an out-arc of the third node.

It is interesting to note that, the graphical representation used here allows uncertain temporal knowledge in terms of the expression of logical "OR" actually characterizes the branch time structure. In fact, duplicated identical out-arcs (e.g., t_i in fig 1 (c)) of a common node refer to various temporal paths branching forwards in the future, and, by symmetry, duplicated identical in-arcs (e.g., t_k in fig 1(d)) of a common node refer to different paths branching backwards in the past. Also, fig 1(e), (f), (g) and (h) show ccombination of conjunction and disjunction.

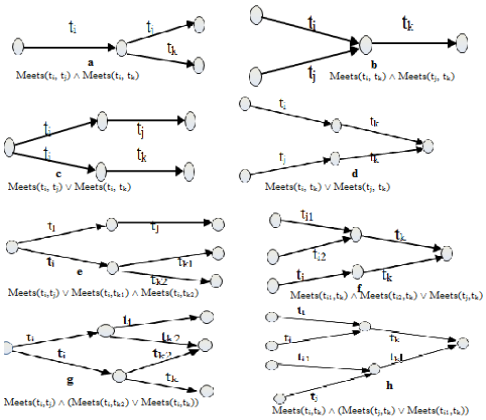


Figure 1. Meets relations showing $\wedge$, $\vee$ and combination

Using BPTG, for the representation of absolute and relative temporal information, we consider an illustrative explain of a BP given in the next sub-section.

6. An Illustrative Example

We take a BP from insurance industry and consider an example of making a claim for insurance for motor vehicle damage. The BP begins with a start event to execute the first atomic process *Record the Claim (RC).* After the atomic process *RC, Calculating Insurance Sum (CIS)* starts and splits the flow, because the decision has to be made if the insurance sum has a minor amount or major amount. If the insurance sum has a minor amount, then only atomic process *Contacting the Garage (CTG)* is processed. But if the insurance sum has a major amount then *CTG* occurs after *Checking History of the Customer (CHC)* using logical operator *OR.* After the *Examination of Results (EOR)* the decision has to be made if the company *Pays for the Damage (PD) or Does Not Pay for the Damage (DPD).* After that decision the *Case is Closed (CCL).*

6.1. BPTG for Business Process

A BP '*P*', where P= (*A, R (A)*), and comma "," in *A* and *R (A)* stands for logical connective "$\wedge$" and from the above example, we can have

A = {(RC, t_1, dur(t_1) = 0 mins), (CIS, t_2, dur (t_2) = 1.3 mins),…}

Where *R(A)* = {*Meets*(RC, t), *Meets*(t, CIS), ...}. In fig 2 below, we show the abstract process model of making a claim for insurance process.

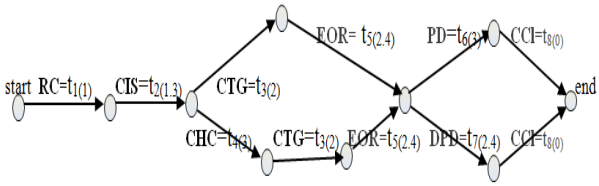


Figure 2. Business Process temporal graph of Making Claim for Insurance process

The above BPTG shows that "Checking the customer History" comes before "Contacting the Garage", both of which processes can be in parallel with "If Insurance Sum is higher" and "Contacting the Garage" "if the insurance sum is lower" which also shows duration knowledge of instance of the of abstract process.

6.2. BPTG for Sub-Process

The concept of a sub-process is an important one in the BPM, discipline and we have given formal definition in section III. From above choosing the OR alternative, we can have a sub-process denoted as P_1. In effect, we form sub-processes by selecting sub-process temporal graph which is connected to the rest of the BPTG by means of two ordered vertices. By ordered here, we mean that they lie on a directed path.

A sub-process of an abstract BP *P(A, R(A))* is a set of abstract processes A_1 and a set of temporal relations $R(A_1)$, such that:

- A_1 is a subset of A and $R(A_1)$ is a subset of $R(A)$, therefore P_1 is a subset of P with the property that the *BPTG* of P_1, is a sub graph of P which can be disconnected by removing all in-arcs to a start vertex v_1 and all out-arcs from an end vertex v_2, where v_1 and v_2 lie on a directed path of BPTG. Fig 3(a) and fig 3(b) will express the sub-process temporal model.

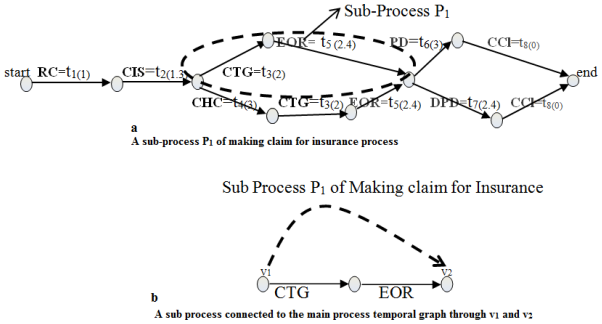


Figure 3. A sub-process P_1 of making claim for insurance process

The process temporal model of making claim for insurance process derived from a set of temporal facts relating to instances of 7 atomic processes. In the same way, the process temporal model P_1 can be derived from the same set of instances for the two atomic processes {contacting the garage, examining result}

So $BPTG_1 = (A_1, R(A_1))$, which is a maximal sub-graph of the graph of $BPTG = (A, R(A))$, with no duplicated time elements. A sub-process is said to be consistent if duration weights can be assigned to all unassigned time elements, so that the signed sum of weights in each graph cycle is zero [13]. There are three facts which follow directly from this:

- A sub-process is also an abstract process in its own right. The sub graph has its own start and end vertices, and is consistent in view of the consistency of the parent process temporal graph.
- Every atomic process is a sub-process, being connected to the main graph only through its start and end vertices.
- Sub processes allow a simplification of BPTG by replacing sub graphs with a single arc to represent an entire sub-process.

Using above facts, fig 3(b) shows a sub-process P_1 of insurance claim process which is disconnected by removing in-arcs to vertex v_1, and out-arcs from vertex v_2. Note that P_1 defines the atomic processes set {Contacting the garage, Examining the result} and its corresponding relations. We see that these two atomic processes also define a BP, with a start vertex and an end vertex. We call this a sub-process of making insurance claim process. The definition of a sub-process defined here is consistent with a sub-process occurring within a parent process.

Notice that any atomic process in P is automatically is a sub-process, so atomic processes are also sub-processes. However, sub-processes of a parent process may be assigned to many different agents, whereas atomic processes must be assigned to a single agent.

We may also express the main structure of the process temporal model, by means showing just the intervals for major sub-processes, as in fig 4, below. The detailed structure of each sub-process can then be shown separately.

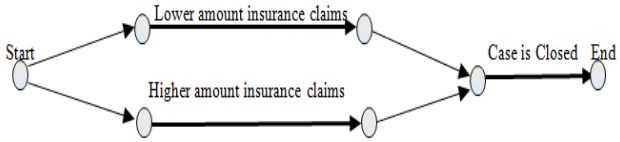


Figure 4. Process temporal model showing major sub-processes in making claim for insurance

The BPTG presented here is akin to the many process diagrams used in the BPM discipline, such as flow charts, UML activity diagrams, and BPMN. In addition, BPM diagrams have constructs for selection of alternative processes and for repetition of processes. Selection in BPM diagrams is shown as a branching graph, and corresponds to the XOR node

in the BPTG. Fig 5 exemplifies this comparison, by showing an activity diagram symbol together with the equivalent BPTG structure. Fig 5(a) shows Flow Chart Selection Symbol and fig 5(b) shows the equivalent BPTG.

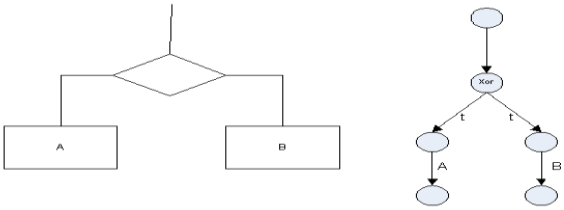


Figure 5. Flow Chart…(b) Process Temporal Graph

Temporal structure does not handle the meaning of the loop as used in BPM. However a loop in the directed graph would correspond to an inconsistent temporal structure, where time would seem to flow back on itself. To discuss this further and extend BPTG, we consider an example of Business Process Modeling Notation (BPMN) discussion cycle [1] as shown in fig 6(a) below, where a loop as in the BPMN discussion cycle does not mean that the same instance of process *P* is invoked again, but rather that a second instance of the same abstract process *P* is invoked again. Figure 6(b) represents the BPMN discussion cycle in the BPTG.

The only construct in BPM which is not natural to the temporal structure is the repeat (loop) construct, which allows cycles in the BPMN. In order to accommodate this within the BPTG, we use an arc, not representing a time element, but rather the invocation of another process instance. This arc is shown as a dashed arrow shown in fig 6(c) below:

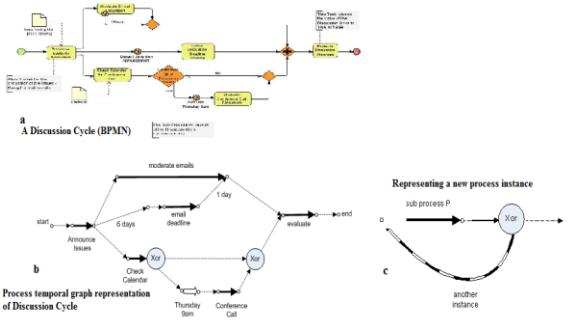


Figure 6. A Discussion Cycle and a loop construct

With the addition of the looping structure with the meaning that a new instance is to be invoked, the BPTG described in this paper is equivalent to most of the practical BPM tools in current use. We may thus use the BPTG equivalent of a practical BP chart or graphical notation to form a bridging the gap as discussed in the section I. In effect, the BPTG can act as a standard for BPM, with a straightforward mapping for the various current tools and techniques.

[1] BPMN Discussion Cycle, *wikipedia.org*, (Accessed on 15th November, 2014), http://en.wikipedia.org/wiki/File:BPMN-DiscussionCycle.jpg

6.3. Reachability Analysis of BPTG

In the previous section, we have seen that BPTG is a concrete realization of abstract process model. Now, we are going to conduct a reachability analysis to show that BPTG is sound and correct.

For a directed process temporal graph *BPTG* = (*A*, *R(A)*), with a set of abstract processes *A* and a set of temporal relations *R(A)* that is also comprised of all possible derived temporal relations, then the reachability relation of BPTG is the transitive closure of *R(A)* which is to say the set of all ordered pairs (A_S, A_E) of vertices in *A* for which there exists a sequence of vertices $v_0 = a_S$, v_1, v_2, …$v_k = a_E$, such that the node v_{i-1}, v_i is in *R(A)* for all $1 \leq I \leq K$.

If process temporal graph is acyclic, then its reachability relation is a partial order; any partial order may be defined in this way, for instance as the reachability relation of its transitive reduction. A noteworthy consequence of this is that since partial orders are anti-symmetric, if a_S can reach a_E, then we know that a_E cannot reach a_S. Intuitively, if we could travel from a_S to a_E and back to a_S, then process temporal would contain a cycle, contradicting that it is acyclic. If BPTG is directed but not acyclic (i.e. it contains at least one cycle), then its reachability relation will correspond to a preorder instead of a partial order and also accommodates the looping structure for occurrence of new instances. Therefore, its correctness is given below:

- A BPTG is sound and complete if it has exactly one initial node, exactly one final node, and all other nodes lie on a path between the initial and the final node.
- A BP has no deadlock iff a BPTG has a final node that can be reached from every reachable vertices; a node is final iff each vertice has reached an end node. Also, BPTG with a preorder structure and based on cyclic behavior an abstract process is repeated with a new instance (loop).

7. Conclusion

In this paper a model theoretic approach adopted to provide a general framework for the business process modeling (BPM). It is also proposed that this approach provides a firm foundation for the BPM. It is intended that the logical foundation using a well-accepted TL gives clarity to the terms employed in the BPM discipline. The framework proposed here shows a grounding theory for the BPM that is in reality an interpretation of a formal system/model. A system without validation is not correct i.e. sound and complete. A proof procedure is provided to show that abstract process model is sound and complete. To achieve this resolution theorem is used. to show its correctness.

As main BPM tools are based on graphical representation to model its BPs. Therefore graphical

representation is used here as well based on general time theory which is consistent. Abstract process model proposed here is mapped to a real world instance of a particular abstract process as a set of real occurrences with real time-stamps using graphical representation. With this mapping, if the deduced temporal constraints represented by the abstract temporal model hold, then we say that the real world process is an instance of the abstract business process. Time structure does not allow loop behavior and to overcome this problem we have used a loop graphical construct. Additionally, we have used reachability analysis to show that the BPTG has the ability to provide a new process instance whenever it has been invoked without conflicting it with the temporal structure.

We have also provided a comparison of our formal system with the prevailing BPM tools and found that our framework is general and expressive than others. This comparison also shows that the formal system can serve as a standard for the BPM. The formal system introduced here is based on a general temporal theory that treats time point and time interval on equal footing.

It is also noted that the BPTG is akin to many BPM tools, and techniques such as flow-chart, activity, Gantt and Pert charts, UML AD and BPMN. The BPTG allows for the expression of both absolute and relative temporal relationship between the time elements, and includes both logical conjunction and disjunction to indicate alternative branching pathways. It can serve as a standard for the BPM, with a clear syntax and semantics of the formal system.

8. References

[1] A. Galton. “Critical Examination of Allen's Theory of Action and Time”, *Artificial Intelligence*, 42, pp.159-188, 1990.

[2] A. Konar. “Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain”, CRC press, 1999.

[3] A. Smith. “An Inquiry into the Nature and Causes of the Wealth of Nations, edited by Edwin Cannan”. University of Chicago Press, 1776.

[4] D. Hilbert. “*Grundlagen die geometrie”*. Leipzig: Dritte Auflage, 1909.

[5] D. Juliane, and W. M. P. Van der Aalast. "Bridging the gap between business models and workflow specifications." *International Journal of Cooperative Information Systems*: pp 289-332, 2004.

[6] J. Allen. “Maintaining knowledge about temporal intervals”, *Communications of the ACM,* v.26 (11), pp.832-843, 1983.

[7] J. Allen. “Towards a General Theory of Action and Time”, Artificial Intelligence, v.23, pp.123-154, 1984.

[8] J. Allen and P. Hayes. “Moments and Points in an Interval-based Temporal-based Logic”, *Computational Intelligence*, v.5, pp.225-238, 1989.

[9] J. Ma, and B. Knight. “A General Temporal Theory”, *the Computer Journal* .37(2): pp 114-123, 1994.

[10] J. Ma, B. Knight, and M. Petridis. "A revised theory of action and time based on intervals and points." *The Computer Journal* 37.10, pp 847-857, 1994

[11] J. Ma, and B. Knight. “Reified Temporal Logics: An Overview”. *Artificial Intelligence Review*, Vol 15, pp 189-217, 2001

[12] J. Ma, and B. Knight. “Representing the Dividing Instant”, *The Computer Journal*, 46(2), pp.213-222, 2003.

[13] J. Ma and P. Hayes. “Primitive Intervals Vs Point-Based Intervals: Rivalsor Allies? the Computer Journal,49(1): 32-41, 2006.

[14] J. Ma. “Ontological Considerations of Time, Meta-Predicates and Temporal Propositions”. *Applied Ontology*, 2(1), pp 37-66, 2007.

[15] M. Hammer, and J. Champy. “Reengineering the Corporation: A Manifesto for Business Revolution”. New York, USA, 1993.

[16] T. H. Davenport. “Process Innovation: Reengineering Work through Information Technology”. Harvard Business School Press, Boston, MA, USA, 1993.

[17] V. J. Benthem. “The Logic of Time”, Kluwer Academic, Dordrech, 1983

