Using a Temporal Constraint Network for Business Process Execution[1]

Ruopeng Lu, Shazia Sadiq, Vineet Padmanabhan, Guido Governatori

School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, Australia
{ruopeng, shazia, vnair, guido}@itee.uq.edu.au

Abstract

Business process management (BPM) has emerged as a dominant technology in current enterprise systems and business solutions. However, the technology continues to face challenges in coping with dynamic business environments where requirements and goals are constantly changing. In this paper, we present a modelling framework for business processes that is conducive to dynamic change and the need for flexibility in execution. This framework is based on the notion of process constraints. Process constraints may be specified for any aspect of the process, such as task selection, control flow, resource allocation, etc. Our focus in this paper is on a set of scheduling constraints that are specified through a temporal constraint network. We will demonstrate how this specification can lead to increased flexibility in process execution, while maintaining a desired level of control. A key feature and strength of the approach is to use the power of constraints, while still preserving the intuition and visual appeal of graphical languages for process modelling.

Keywords: Process modelling; Workflows; Temporal constraints; Constraint Satisfaction

1 Introduction

It has been long established that automation of specific functions of enterprises will not provide the productivity gains for businesses unless support is provided for overall business process control and monitoring. Workflow systems have delivered effectively in this area for a class of business processes, but typical workflow systems have been under fire due to their lack of flexibility, i.e., their limited ability to adapt to changing business conditions. In the dynamic environment of e-business today, it is essential that technology supports the business to adapt to changing conditions, where different process models should be derived from existing ones to tailor individual process instances. However, this flexibility cannot come at the price of process control, which remains an essential requirement of process enforcement technologies.

[1]This work is partly supported by the Australian Research Council funded Project DP0558854

Providing a workable balance between flexibility and control is indeed a challenge, especially if generic solutions are to be offered. Clearly, there are parts of the process which need to be strictly controlled through fully predefined models. There can also be parts of the same process for which some level of flexibility must be offered, often because the process cannot be fully predefined due to lack of data at process design time. For example, in call centre responses, where customer inquiries and appropriate response cannot be completely pre-defined, or in health systems, where patient care procedures resulting from individual patient diagnosis cannot be anticipated.

In general, a process model needs to be capable of capturing multiple perspectives (Jablonski and Bussler 1996), in order to fully capture the business process. There are a number of proposals from academia and industry on the modelling environment (language) that allow these perspectives to be adequately described. Different proposals offer different level of expressiveness in terms of these perspectives. Basically these perspectives are intended to express the constraints under which the business process can be executed such that the targeted business goals can be effectively met.

We see three essential classes of constraints:

- **selection** constraints that define *what* activities constitute the process,
- **scheduling** constraints that define *when* these activities are to be performed, both in terms of ordering as well as temporal dependencies, and lastly
- **resource** constraints that define *which* resources are required to perform the activities.

These constraints are applicable at two different levels, *process* level and *activity* level. Process level constraints specify what activities must be included within the process, and the flow dependencies within these activities including the control dependencies (such as sequence, alternative, parallel etc.) and inter-activity temporal dependencies (such as relative deadlines). Activity level constraints constitute the specification of various properties of the individual activities within the process, including activity resources (applications, roles and performers, data), and time (duration and deadline constraints).

Although, the various constraints are inter-related, in this paper, we primarily focus on process level scheduling constraints. In typical process specifications, such constraints are specified using rigid control flow dependencies. One such specification approach is introduced in section 2. Although such approaches have had significant success for a large class of processes due to their intuitive and visual appeal, their appropriateness is debatable for processes that require much greater flexibility in execution. As an example, consider the following scenario:

In a Telco servicing organization, customer requests are received through a web portal. Requests are then assigned to supervising engineers. These supervising engineers are considered domain experts capable of diagnosing service requests and preparing a customized *service plan*. The service plan essentially consists of several diagnostic tests and subsequently one or more actions. This service plan is then executed and results of the services rendered are compiled into a service report and logged into the system. Actual execution of the service plan may be long duration and involve delegation to several field workers.

In this scenario, consider specifically the task that prepares the service plan. Suppose that a number of diagnostic tests, (say 5 tests, T1, T2, … T5), are available. Any number of these tests can be prescribed for a given request, and in a given order. The supervising engineer has the flexibility to design a plan that best suits the customer request. However, there are certain restrictions on the scheduling of these tests. For example, T4 and T5 must be performed at the same time, and T2 must always be performed before T3. Providing a complete specification of all valid configurations of these tests is clearly not feasible, but would be necessary in typical control flow based graphical languages.

In this paper we target the modelling and execution of processes which have requirements as identified in the above scenario. We propose a framework which **firstly** allows scheduling constraints to be captured through a temporal constraint network. Temporal Constraint Networks (TCN) have been widely studied (Allen 1983; Vilain, Kautz et al. 1989; van Beek 1990; Dechter, Merir et al. 1991; van Beek 1992; Meiri 1995; Nebel and Buckert 1995; Drakengren and Jonsson 1996). Essentially TCNs are defined through 13 interval relations (Allen 1983) describing the relative positions between each pair of objects, including *before, meets, during, overlaps, starts, finishes*, the inverse of these relations *after, met-by, contains, started-by, finished-by* and a special relation *equals*. Temporal knowledge of multiple time intervals can be expressed by these relations and reasoned about in such a TCN. Using well established results from literature, we will present a discussion on the properties of such networks, showing that they not only provide a highly expressible and succinct specification to meet advanced requirements as described above, but also viable reasoning techniques for determining network consistency (i.e. ensuring executable processes). We will cover this discussion in section 3.

The proposed framework **secondly** also provides an execution environment in which individual instances can be customized according to specific needs, but still conform to process constraints. Instance customization is offered in an intuitive graphical language, whereas analysis on the correctness of the instance template is provided through TCN reasoning. In section 4, we will deliberate on how this is achieved, by illustrating the concepts through the above scenario. A review of related literature is provided in section 5, and finally conclusions and potential extensions are presented in section 6.

2 Background Concepts

In this section we provide essential concepts necessary for subsequent discussion. These concepts relate to typical business process modelling and execution, constraint satisfaction in general and temporal constraint networks in specific.

A substantial segment of the BPM space endorses the use of graphical models due to their intuitive and visual appeal (see e.g. (van der Aalst 1996; Coalition 1998; WfMC 1998; Sadiq and Orlowska 1999; Sadiq and Orlowska 2000)). An example of such a modelling language is given in figure 1.

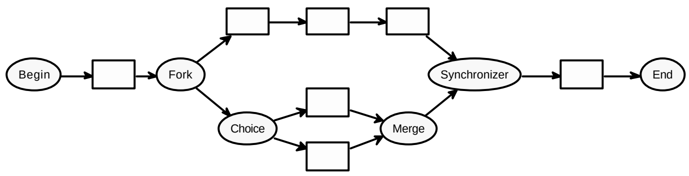

Figure 1. Graphical Modelling Language

The language consists of basic constructs such as sequence, fork, choice etc. Further details of this language can be found in (Sadiq and Orlowska 1999). We will use this simple notation to illustrate various examples in this paper. For example, figure 2 provides three acceptable process models for the telco scenario.

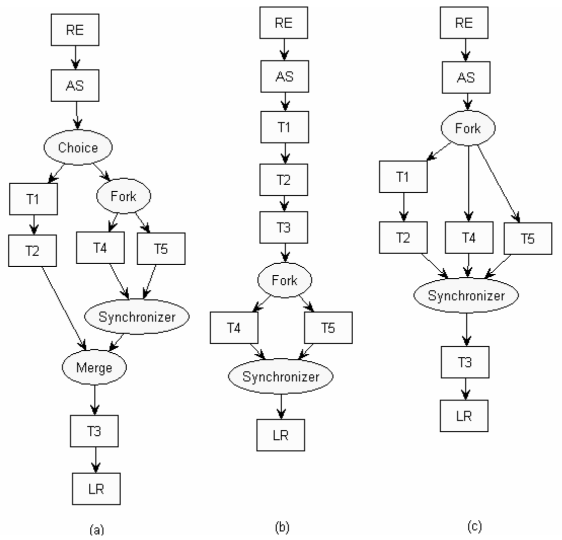

Figure 2: Process Models for Telco Scenario

Tasks RE, AS, T1, T2, T3, T4, T5, LR in figure 2 correspond to the following process activities in the scenario (respectively) - customer Request Enter, Assess Situation and preparation of service plan by supervisor engineer, Test 1 to Test 5, and finally Logging service Report.

Constraint satisfaction is a well known problem solving approach where a problem is formulated as a constraint satisfaction problem (CSP) and searching for solutions and reasoning about some hypotheses in a restricted domain of knowledge can be performed. The process of problem formulation is called *constraint modelling* and the process of knowledge reasoning and solution searching is called *constraint processing*. The problem to be solved is modelled and represented in a *constraint network* N, where N *is* a triple $< X, D, C >$[1]. X is a finite set of *variables* $X = \{X_1, X_2, \ldots X_n\}$ with respective *domains* $D = \{D_1, D_2, \ldots D_n\}$, which contain the possible values for each variable. C is a set of *constraints* $C = \{C_1, C_2, \ldots C_t\}$ where each constraint C_i is a relation that imposes a limitation on the values a variable, or a combination of variables may be assigned to. A constraint can be specified on single variable (unary constraint), or on a pair of variables (binary constraint). (Jeavons 1999; Dechter 2003). However, practical CSPs with higher order constraints are generally NP-complete (Cook and Mitchell 1997), since modelling of real world problems often requires a large number of variables with large domains. e.g. to determine the satisfiability of a formula containing three literals (3-SAT problem) is NP-complete.

A simple CSP can be given as finding an assignment of values from domain $\{1, 2, 3\}$ to variables x, y and z such that x > y and y > z hold, where $X=\{x, y, z\}$, $D = \{D_x, D_y, D_z\}$, $D_x = D_y = D_z = \{1, 2, 3\}$ and $C=\{C_{xy}, C_{yz}\}$, $C_{xy} = x > y$, $C_{yz} = y > z$.

Figure 3 shows the constraint graph of this problem. A vertex in the constraint graph represents a variable and the arc between two vertices represents the constraint between the two variables.

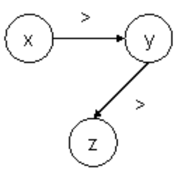

Figure 3: Constraint Graph

A solution of CPS is an assignment of a single value from its domain to each variable such that no constraint is violated. A problem may have one, many or no solution. A problem that has one or more solutions is *satisfiable* or *consistent*. The only possible solution to the previous example is x = 3, y = 2, z =1. Constraint satisfaction in general is a well-studied area and many techniques are available for reasoning and solving different class of CSPs.

The requirements to represent and reason about scheduling constraints in business processes necessitate a formal framework to capture temporal relations between process activities. Such temporal information is often incomplete and indefinite. (Allen 1983) has proposed a framework, called Interval Algebra (**IA**) network, for representing and reasoning about such information.

[1] To be more precise, we can define $N = < X, D, d, C >$, where d is a function $d : x \to 2^D$ which maps a variable to some value in the corresponding domain.

Relation	Symbol	Inverse
x before y	b	bi
x meets y	m	mi
x overlaps y	o	oi
x starts y	s	si
x during y	d	di
x finishes y	f	fi
x equal y	eq	eq

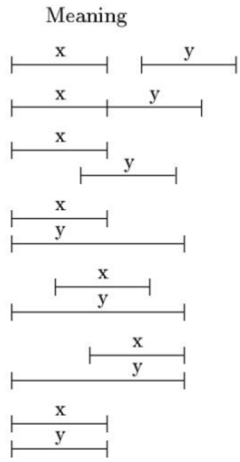

Figure 4: Basic Interval Relations (Allen 1983)

In (Allen 1983), 13 basic relations are given (see figure 4), which can hold between two intervals. In order to represent indefinite information, the relations between two intervals are allowed to be a disjunction of the basic relations. For example, the relation {*before*, *meets*} between intervals x and y restricts that x either finishes before y starts or x finishes immediately before y.

A restricted class of IA networks (Vilain, Kautz et al. 1989), denoted **SA**, can be translated into the Point Algebra framework, called Point Algebra (**PA**) network in polynomial time without loss of information. A point algebra network is a network of binary relations where the variables represent time points, and the binary relations between variables are disjunctions of the basic point relations $\{<, =, >\}$. In SA networks, the allowed relations between two intervals are the subsets of IA that can be represented using the relations $\{<, =, >\}$ into conjunctions of relation between the endpoints (start and finish points) of the intervals (van Beek 1990). For example, let the (T_1^-, T_1^+) and (T_2^-, T_2^+) denote the start and finish points of interval T_1 and T_2 respectively, an IA relation $T_1\{meets\}T_2$ can be translated into SA as $(T_1^- < T_2^-) \wedge (T_1^- < T_2^+) \wedge (T_1^+ = T_2^-) \wedge (T_1^+ < T_2^+)$. The complete description of SA can be found in (van Beek 1990).

These concepts have been utilized in temporal constraint networks (TCN). A temporal constraint network is a subclass of constraint networks where the representations of temporal information can be viewed as binary constraint networks and constraint satisfaction techniques can be used to reason about the temporal information. The variables in TCNs represent time intervals and constraints represent sets of allowed temporal relations between them. A solution or *consistent instantiation* of the network is an instantiation of the variables such that all the constraints between the variables are satisfied (van Beek 1992; Dechter 2003)

There are two fundamental reasoning problems in temporal constraint networks (van Beek 1992; Nebel and Buckert 1995; Dechter 2003). Given a temporal constraint network *N*,

- decide whether there exist a consistent assignment of values to all variables such that no constraint is violated, also known as the SAT problem, and
- find the minimal network of *N*.

The first problem is to reason about whether the set of temporal relations modelled in *N* is *valid* by determining whether the given temporal information is consistent, that is, whether it is possible to find a scenario where the intervals can be arranged along the time line according to the given information.

The second problem is to find (if the information is consistent) the feasible relations between all pairs of intervals, that is find one, some or all arrangements of the intervals along the time line, each corresponding to a possible scenario.

The major advantage of the constraint satisfaction approach to solve process modelling problems is that all a process designer has to do is to provide an appropriate formulation of the CSP. Well established techniques from constraint processing can be utilized to determine network consistency, as well as to find solutions. As such, we apply temporal constraints to modelling scheduling requirements for ordering of process activities in business processes in a constraint network. In the following section, we will provide formal specifications to such a framework.

3 Business Process Constraint Network

Informally, we consider the business process as a set of tasks, where a task is either an atomic activity (a unit of work to be done) or a sub-process that contains one or more activities. The ordering of the tasks is specified by the temporal relations between the tasks.

3.1 Definition of BPCN

We follow Allen's IA network approach (Allen 1983) to represent and reason about temporal information of business process. A task *T* in a business process is modelled as a time interval, which is an ordered pair (T^-, T^+) such that $T^- < T^+$, where T^- and T^+ are interpreted as points on the time line. In particular, T^- is the point of time when task *T* starts execution, and T^+ is the point of time when *T* finishes execution. Henceforth, we refer to a task *T* as a time interval interpreted by the endpoints T^- and T^+.

The scheduling constraints between the tasks constituting a business process can be expressed by some combination of the 13 pair-wise interval relations (figurer 4) where each relation can be defined in terms of endpoint relations.

One or more relations can be defined on each pair of tasks. If more than one relation is defined on the same pair of tasks, we take the disjunction of the relations, which requires at least one relation must hold for all instances of the process. These relations describe a *partial order* of the tasks while a *total order* can be given if we assign exactly one relation between each pair of tasks. A process instance is a *totally ordered* instance if for every pair of tasks in the process either one of the 13 relations holds. The characteristic of partial order relations corresponds to the uncertain relationship between tasks, which allows for a large number of possibilities in which execution of tasks can be ordered according to different instances of a process.

Given a set of interval relations defined on the tasks of a business process, we can determine through logical inference whether a satisfiable ordering of task can be constructed. We define a Business Process Constraint Network (BPCN) based on a temporal constraint network adapted to represent scheduling constraints between tasks in the business process.

More specifically, a **BPCN** is a temporal constraint network $N = <X, D, C>$ where the set of variables $X = \{T_1, \ldots, T_n\}$ is the set of all tasks in the business process represented as time intervals, the set of domains $D = \{D_1, \ldots, D_n\}$ is the set of ordered pairs of discrete time values $\{(s, e) \mid s < e\}$, representing the start (s) and end (e) points of the corresponding task intervals. Binary constraints between pairs of interval variables are given as IA relations. The constraint C_{ij} between task T_i and T_j is defined as $C_{ij} \subseteq R = \{b, bi, m, mi, o, oi, s, si, f, fi, eq\}$, which describes the allowed relative locations of paired tasks in the discrete time line. A subset of basic relations corresponds to an ambiguous, disjunctive relationship between intervals. As a result, the set of constraints C in a BPCN defines the partial-order of the process execution model.

A *solution* to a BPCN is an assignment of a pair of values to each variable such that no constraint is violated. A solution can be established by assigning a single relation to each pair of tasks that is consistent with the constraint definition. A solution defines a total order of the process execution model.

3.2 Consistent BPCN

Consistency is used to describe the quality of the constraints defined in the constraint network. If conflicts exist between the constraints, or the inferred constraints, then we can conclude that the constraint network is inconsistent and hence no solution exists. The problem of determining whether a given BPCN is consistent can be mapped to the SAT problem.

It is desirable that given a BPCN, one can derive multiple process instances to suit different process requirements. Thus, we need to make sure that at least one satisfiable instance can be found, i.e. to determine the given BPCN is consistent (satisfiable).

Since the set of 13 interval relations are totally disjoint, and in Allen's IA network we allow multiple relations defined on the same pair of variables, conflicts of constraints in the BPCN can only exist between different pairs of variables. For example, we have a network of three variables, $X = \{T1, T2, T3\}$ and the constraints $C = \{C_{12}, C_{13}, C_{23}\}$, where C_{12}= T1{*b*, *m*}T2, C_{13}= T1{*s*, *eq*} T3 and C_{23} = T2{*b*, *m*} T3. The definition for each C_{ij} does not cause conflicts since for a particular process instance we only require one relation to hold for each pair of variables, i.e. T1{b}T2. However, the network is inconsistent since from C_{12} and C_{23} we can infer C'_{13} = T1{b}T3, but $C'_{13} \cap C_{13} = \varnothing$, which means we cannot find a scenario that satisfies C_{12}, C_{23} and C_{13} at the same time. Hence the network is inconsistent.

The technique to determine consistency for BPCN is based on enforcing local consistency on the network. Before defining local consistency, we first give a formal definition for a consistent BPCN.

A BPCN $N = < X, D, C >$ is said to be *consistent* if we can find a consistent scenario of network N. A network N' is a consistent scenario of a network N if and only if (iff):

1. there exists exactly one relation between each pair of variables (T_i, T_j) in N', namely, $|R'_{ij}|=1$; and
2. every such relation R'_{ij} in N' is a subset of the relation between the same pair of variables in N, namely, $R'_{ij} \subseteq R_{ij}$; and
3. there exists a consistent instantiation of N'.

We assume that each variables have sufficient large domains, as such when condition 1 and 2 hold, we can determine N' is a consistent scenario of N. To find a consistent scenario we simply search through the different possible Ns that satisfy conditions 1 and 2 (van Beek 1992).

Given a BPCN N where the set of relations C is restricted by SA subclass, to determine the consistency of N only requires to determine whether N is *path-consistent* (Vilain, Kautz et al. 1989; van Beek 1990). To define path-consistency in BPCN, we need to define the following operations.

IA describes all possible relations between two intervals, as such the universal relation between two intervals (which means there is no constraint defined on them) is the set of basic relations R.

Being part of Allen's IA, the inverse, intersection and composition operations on pairs of variables are also defined, which are given as follows (Allen 1983; Dechter 2003):

The *inverse* $R^{\cup}$ of the relation R is the relation $R^{\cup} = \{(a,b) \mid (b,a) \in R\}$.

The *intersection* of two IA relations R' and R'', denoted by $R' \cap R''$, is the set theoretic intersection of R' and R''. For example, given $R' = \{o, s, f, m\}$ and $R'' = \{s, f, d\}$, $R' \cap R'' = \{s, f\}$.

The *composition* of two basic IA relations r' and r'', denoted by $r' \otimes r''$, is defined by the transitive table (see figure 5). For example, the basic relations T1 *meets* T2, T2 *before* T3 induce a new (single or composite) relation T1 *before* T3. The composition of two composite relations R' and R'', denoted by $R' \otimes R''$, is the composition of the constituent basic relations:

$$R' \otimes R'' = \{r' \otimes r'' \mid r' \in R', r'' \in R''\}$$

r' \ r''	**b**	**s**	**d**	**o**	**m**
b	b	b	b o m d s	b	b
s	b	s	d	b o m	b
d	b	d	d	b o m d s	b
o	b	o	o d s	b o m	b
m	b	m	o d s	b	b

Figure 5: Portion of the transitivity table defined by (Allen 1983)

A binary constraint C_{ij} is path-consistent relative to a variable T_k iff $(C_{ij} \cap (C_{ik} \otimes C_{kj})) \neq \varnothing$. A BPCN is a path-consistent BPCN iff for every relation R_{ij} (including universal relations) and for every $k \neq i, j$, R_{ij} is path-consistent relative to T_k.

Validation of the constraint definition on a BPCN can be achieved by applying a generic path-consistency algorithm.

```
Input: An IA network N
Output: A path-consistent IA network
1    for k:= 1 to n do
2       for i,j:=1 to n do begin
3          C_ij ← C_ij ∩ (C_ik ⊗ C_kj)
4          if C_ij = ∅ then break
```

Figure 6: Path-Consistency Algorithm (Dechter 2003)

We repeatedly apply the above algorithm to the network N until no further changes can be made to the current constraints or some constraint becomes empty, indicating inconsistency. The operation $C_{ij} \leftarrow C_{ij} \cap (C_{ik} \otimes C_{kj})$ applied to each constraint is called the relaxation operation (Dechter 2003). For some class of IA relations, this algorithm is guaranteed to determine consistency in $O(n^3)$ time, where n is the number of intervals in N. Further discussions can be found in section 4.

4 Execution Framework

As explained in the previous section, process definition consists of a pool of activities and a small number of constraints defined on those activities. However, the process instances are allowed to follow a very large number of execution paths. As long as the given constraints are met, any execution path dynamically constructed at runtime is considered legal. This ensures flexible execution while maintaining a desired level of control through the specified constraints. The key feature and strength of the approach is to use the power of constraints, while still preserving the advantages of graphical languages.

Below we explain the core functions of the process management system based on the concepts presented above. The discussion is presented as a series of steps in the specification and deployment of the Telco example process. Figure 7 provides an overview diagram of these steps and associated functions.

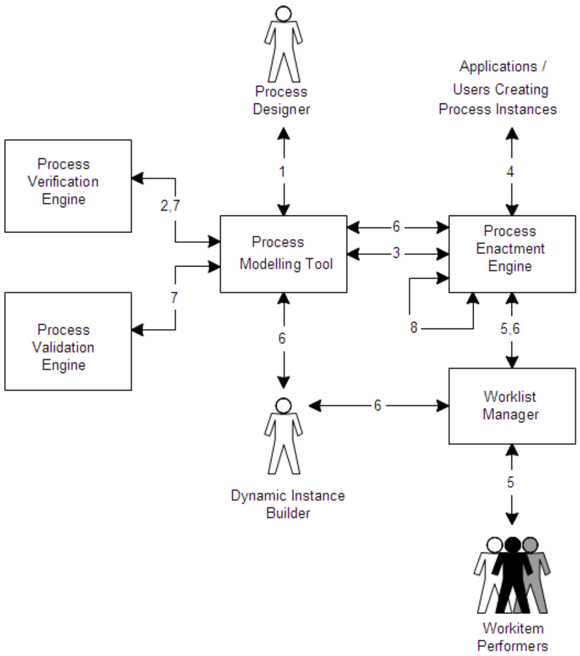

Figure 7: Framework Overview

- **Step 1**: The definition of the (flexible) process model takes place. The pool of activities and associated constraints are defined.

We specify the telco business process using BPCN. The process is represented as $N = \langle X, D, C\rangle$. There are 8 tasks in this business process, X = {*RE, AS, T1, T2, T3, T4, T5, LR*}, which correspond to customer request, situation assessment by Supervisor Engineer, test 1 to test 5, and logging service report, respectively.

The set of constraints C is defined according to the scheduling requirements. Consider the following restrictions on the scheduling of the tests: Test 1 must start before test 2 starts. If both tests do not solve the problem then further test 3 is ordered. Test 3 must not start before test 2 finishes. Test 4 and test 5 must start at the same time. Besides, no tests can start before Supervisor Engineer starts assessing the report, and no tests can be started after service report is logged. One can define the set of scheduling constraints C = {$C_{RE\text{-}AS}$, $C_{AS\text{-}T1}$, $C_{AS\text{-}T4}$, $C_{T1\text{-}T2}$, $C_{T2\text{-}T3}$, $C_{T4\text{-}T5}$, $C_{AS\text{-}LR}$,, $C_{T2\text{-}LR}$, $C_{T3\text{-}LR}$, $C_{T5\text{-}LR}$}, where

$C_{RE\text{-}AS}$= RE{*b, m, o*}AS $C_{T4\text{-}T5}$ = T4{*s, si, eq*}T5

$C_{AS\text{-}T1}$= AS{*b, m, o*}T1 $C_{AS\text{-}LR}$ = AS{*b, m*}LR

$C_{AS\text{-}T4}$= AS{*b, m, o*}T4 $C_{T2\text{-}LR}$ = T2 {*b, m*}LR

$C_{T1\text{-}T2}$= T1 {*b,m,o,di,fi*}T2 $C_{T3\text{-}LR}$ = T3 {*b, m*}LR

$C_{T2\text{-}T3}$ = T2{*b, m*}T3 $C_{T5\text{-}LR}$ = T5 {*b, m*}LR

For example, constraint $C_{T2\text{-}T3}$ = T2{*b, m*}T3 defines a precedence order on the executions of T2 and T3, which requires T2 must finish execution before or meets T3 (Figure 8).

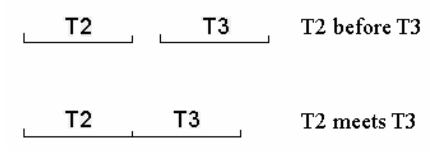

Figure 8: Valid relations between T2 and T3

Figure 9 shows some of the **many** valid orders of execution for T2 and T3 in some instance templates based on graphical model. The unnamed tasks represent any valid tasks in the process.

On the graphical model, relations {*o, oi, s, si, d, di, f, fi, eq*} correspond to concurrent execution pattern, i.e., there is no path between two tasks. Relations {*b, bi, m, mi*} correspond to either concurrent or serial execution pattern. The interval relation between two tasks in concurrent execution pattern requires consideration of the execution duration of the tasks, i.e. for each task, its estimated maximum duration must be provided. T2 and T3 in figure9 (a) (b) (c) execute in serial, while in figure9 (d) execute in concurrent threads of control.

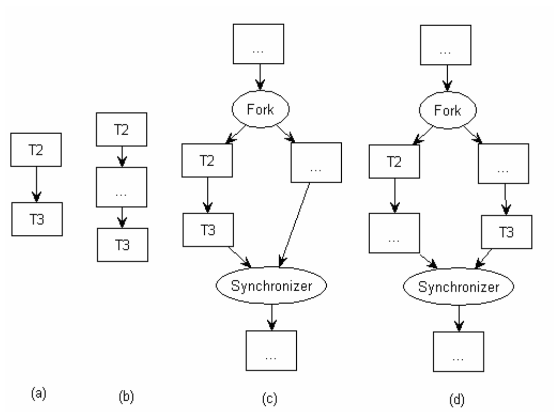

Figure 9: Valid execution order of T2 and T3. (a)(c) and (d) correspond to T2{*meets*}T3, (b) corresponds to T2{*before*}T3

If no constraint is defined on a pair of tasks, then universal constraint applies, which means any 13 relations can be assigned to this pair of tasks.

Figure 10 shows the constraint graph of the BPCN network *N*.

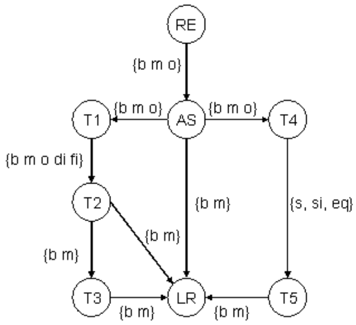

Figure 10: Constraint graph of *N*

- **Step 2**: The process definition is verified for structural errors (Sadiq and Orlowska 2000). The validation of the given constraint set may take place at this time.

This step is to determine whether the BPCN is consistent. We apply the algorithm shown in figure 6 to the BPCN, the resulting consistent network is shown in figure 11.

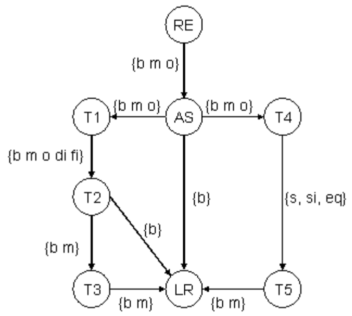

Figure 11 Path-Consistent Network of *N*

In the original network, T2{m}LR is not consistent with respect to $C_{T2\text{-}T3}$. and $C_{T3\text{-}LR}$ since from T2{b,m}T3 and T3{b,m}LR we can infer through transitive table T2{b}LR but not T2{m}LR. Thus T2{m}LR has been deleted from $C_{T2\text{-}LR}$. Similarly AS{m}LR has been deleted from $C_{AS\text{-}LR}$.

It is important to point out that the choice of constraints that will be removed as a result of conflicts is a design issue. The framework will only identify which constraints have conflicts. Process designers then have to make a decision based on process requirements, as to which constraint can be removed.

- **Step 3**: The definition created above is uploaded to the process engine. This process model is now ready for deployment.

- **Step 4**: For each case of the process model, the user or application would create an instance of the process model. On instantiation, the engine creates a copy of the process definition and stores it as an instance template. This process instance is now ready for execution.

- **Step 5**: The available process activities of the newly created instance are assigned to performers through work lists and activity execution takes place as usual, until the instance needs to be dynamically adapted to particular requirements arising at runtime (as shown in next step).

- **Step 6**: The knowledge worker or expert user, shown as the dynamic instance builder, will invoke a special build function, and undertake the task of dynamically adapting the instance template with available pool of activities, while guided by the specified constraint set. This revises the instance template. The build function is thus the key feature of this approach and requires the capability to load and revise instance templates for active instances.

An instance template is a particular customization of a given instance to suit runtime requirements, e.g. a particular configuration of tests prescribed by a service plan. Instance templates define total order of task execution. Figure 2(b)(c) are examples of valid instance templates for the Telco scenario.

- **Step 7**: The next step is to validate the new template, to ensure that it conforms to the correctness properties of the language as well as the given constraints.

An instance template is a totally ordered process instance, where for each pair of tasks, there must be exactly one relation between them. The total ordering in the given template is defined by the assignment of values to the endpoints of each task instance and visualised in the graph-based model. The template validation service first translates the total ordering of task instances from graph-based model into the interval model and checks whether the given sequence of task execution conforms to the constraints defined in the network. The objective of the translation is to find out the implicit temporal relations between tasks and check against the process constraints defined as interval relations on the tasks.

Take the instance template shown in figure 2(c) as the instance template to be validated. Through the PA-IA translation table (van Beek 1990) given in figure 12, we can work out the interval relation between each pair of tasks (as shown in figure 13). Then we check for each translated relation r'_{ij}. If r'_{ij} belongs to R_{ij} of the consistent relations given in figure 11, then r_{ij} is a valid relation between T_i and T_j. If r_{ij} is valid for all T_i and T_j in the instance template, then this template is a valid process instance according to N (a consistent scenario of N).

IA \ PA	$T_i^- T_j^-$	$T_i^- T_j^+$	$T_i^+ T_j^-$	$T_i^+ T_j^+$
{eq}	=	<	>	=
{b}	<	<	<	<
{d}	>	<	>	<
{o}	<	<	>	<
{m}	<	<	=	<
{s}	=	<	>	<
{f}	>	<	>	=
{di}	<	<	>	>
{oi}	>	<	>	>
(fi)	<	<	>	=

Figure 12: PA-IA translation for basic IA relations (van Beek 1992)

The validation procedure corresponds to determining whether a given network instance is a consistent scenario of the original network, as discussed in section 3.2.

PA / IA	$T_i^- T_j^-$	$T_i^- T_j^+$	$T_i^+ T_j^-$	$T_i^+ T_j^+$	Translated Relation R'_{ij}
RE-AS	<	<	=	<	$\{m\}$
AS-T1	<	<	=	<	$\{m\}$
AS-T4	<	<	<	<	$\{b\}$
T1-T2	<	<	=	<	$\{m\}$
T2-T3	<	<	=	<	$\{m\}$
T4-T5	=	<	>	=	$\{eq\}$
AS-LR	<	<	<	<	$\{b\}$
T2-LR	<	<	<	<	$\{b\}$
T3-LR	<	<	=	<	$\{b\}$
T5-LR	<	<	<	<	$\{b\}$

Figure 13: PA-IA translation for the given instance template

Since every translated relation r_{ij} in the given instance template belongs to R_{ij}, we can determine that the instance template is valid. We can say that the instance template as a totally ordered process instance is consistent with regard to the constraint definition in *N*.

- **Step 8**: On satisfactory validation results the newly defined (or revised) instance template resumes execution. Execution will now continue as normal, until completion or until re-invocation of the build function, in which case steps 6-8 will be performed again.

Discussion

The execution framework presented above is based on the fact that consistency of BPCN can be determined by algorithm shown in figure 6, and the translations between IA and PA are made possible without loss of information.

It has been shown (Allen 1983; Vilain, Kautz et al. 1989) that the algorithm applied in step 2 is sound but incomplete for the full IA network. It is sound because it does not introduce invalid relations to the network. It is incomplete because in some cases consistency cannot be determined. Determining the minimal network for some class of IA network is known to be NP-complete. However, in the execution framework, we only consider the *SA*, a subclass of *IA* network that can be translated into *PA*, because we also need such translations when verifying instance templates given in graphical process models. *SA* networks are tractable (Vilain, Kautz et al. 1989; Schwalb and Vila 1998), where enforcing path-consistency correctly decides consistency of the network (SAT) in $O(n^3)$ where *n* is the number of intervals.

Besides, many tractable subclasses of IA network have been identified, including pointisable IA by (Vilain, Kautz et al. 1989; van Beek 1990), tractable subclass of the Point-Interval algebra by (Drakengren and Jonsson 1996; Jonsson, Drakengren et al. 1996), ORD-Horn subclass of IA (Nebel and Buckert 1995) and Interval-point algebra (IPA) network by (Meiri 1995). A complete classification of tractability in Allen's Algebra has been given by (Drakengren and Jonsson 1997). On the other hand, approximation can also be made to express intractable subclass of relations by the tractable counterpart(van Beek 1989).

Furthermore, it is also worthwhile to consider how many relations are sufficient to describe scheduling relations for certain classes of business processes. If no constraint is specified on a BPCN, we permit any combinations of ordering of the tasks in any process instance. If too many constraints are specified, the constraint network is too rigid, and corresponds to the over-constraint problem (Beaumont, Sattar et al. 2001). The minimal requirement is that any constraint definition should permit at least one consistent scenario (to make constraint network consistent). The practical requirement however is that the constraint definition should permit a *large* number of consistent scenarios. This is the case when the full potential of the proposed framework will be realized (as illustrated previously in this section).

5 Related Work

Constraints have been incorporated with business process modelling. (Crampton 2004) identifies a generic class of constraints, called entailment constraints, which restrict the execution order of process tasks with respect to authorisation. e.g."Task 2 must be performed by a role that is more senior than the role that performed task 1"(Crampton 2004).

In (Tsang 2003), constraint satisfaction in business process modelling is aligned with Distributed Constraint Satisfaction (DCSP), a branch of CSP in a collaborative agent environment. An additional set of constraints, *E*, called open constraints is used to capture external and uncertain information. Many studies on DCSP are available in literature (Yokoo 2001).

Planning and scheduling are major applications in constraint satisfaction. (Barták 1999) provides a classification for resource allocation and temporal constraints, as well as dynamic models for reasoning about such information. Particularly, time is modelled in either discrete model or event-based model. In discrete time model, the timeline is divided into a sequence of discrete time intervals with some duration. The variables are time intervals describing durations of process activities. This model is applicable for modelling processes where time intervals represent individual tasks. The event-based model only capture time points when change takes place. This model associate mostly with resources constraints. In our execution framework, the temporal information is modelled as discrete time intervals.

Temporal reasoning techniques have been applied to business process modelling, mostly to capture relative and absolute deadline constraints (Marjanovic and Orlowska 1999; Marjanovic 2000; Li and Yang 2004; Eder, Panagos et al. 1999; Eder, Gruber et al. 2000; Combi and Pozzi 2003). In particular, (Marjanovic 2000) provides a dynamic verification algorithm for absolute and relative deadline constraints in workflow, where the algorithm is based on execution durations represented by metric points. (Combi and Pozzi 2003) describe absolute and relative deadline constraints based on endpoints of intervals, as well as some considerations to represent fork and merge operators. It is obvious that these constraints can be described by a small subset of *PA*, but there is no constraint validation algorithm given in this approach. (Eder, Panagos et al. 1999) present a timed workflow graph approach to express the upper and lower bound constraints of task execution, where the constraint semantics is based on the execution durations and relative deadlines of process tasks. (Bettini, Wang et al. 2002) present a quantitative temporal constraints model which supports multiple time granularity. In this model, constraints are defined on quantitative time points (i.e. seconds, hours), such time points are regarded as variables, and the constraints are defined as temporal distances. In some process modelling approaches, scheduling constraints are incorporated with resource allocation constraints, such as (Li and Yang 2004; Li and Yang 2004; Tan, Crampton et al. 2004; Tan, Crampton et al. 2004).

The distinctions between the framework proposed in the paper and the previous work can be made as follows: Firstly, we have shown that a large subset of full IA network, called the SA network can be used to represent temporal relations in business processes within the BPCN. Secondly, we have shown through a case study that using generic constraint propagation techniques (path-consistency algorithm shown in figure 6) is sufficient to provide validation for such information in BPCN. Last but not least, we have shown translations between graph-based process description to interval-based process description where the former enables intuitive model expression and the later provides a wealth of reasoning techniques.

6 Summary and Outlook

In summary, we have presented a framework that allows for flexible business process execution. The framework is based on the notion of process constraints, and in this paper a particular sub class of process constraints has been considered. In general, we see the level of definition of these constraints along a continuum of specification. There is the completely predefined model on one end, and the model with no predefinition on the other. Thus the former only has strong constraints (e.g. X and Y are activities of a given process, and Y must immediately follow X), and the latter no constraints at all. The former extreme is too prescriptive and not conducive to dynamic business environments; and the latter extreme defeats the purpose of process enforcement, i.e. with insufficient constraints, the process goals may be compromised and quality of service for the process cannot be guaranteed. Finding the exact level of specificity along this continuum will mostly be domain dependent. However, technology support must be offered at a generic level. This work has accordingly attempted to address the need to provide a modelling environment wherein the level of specification can be chosen by the process designer such that the right balance between flexibility and control can be achieved.

We see significant potential in expanding this framework to incorporate other classes of constraints, and especially to study the interplay between them. For example if two tasks X and Y have a scheduling constraint on them, defined by an overlap relation $X\{o\}Y$, and then a resource constraint is also defined on them, say by the *binding of duty* (Li and Yang 2004) relation (i.e. X and Y must be performed by the same resource), what impact does this have on the overall constraint network. There can be several interpretations of this problem, which need to be analysed to formulate workable solutions. However, the essence of the framework will still hold true, that is, a small number of constraints can potentially be specified to realize a very large number of valid instances at runtime.

Another interesting problem is to augment the template construction (see section 4) with an intelligent search function for *best* template. This requires at a minimum a facility to build an objective function into the *BPCN* and furthermore a facility to search the solution space of the *BPCN* for solutions meeting the objective. Example of such an objective function can be minimum time span of part or whole of the process, minimum consumption of a given resource, maximal concurrent execution of process activities etc. Such a service could greatly enhance the productivity of the knowledge worker who is dynamically building the template, by not only allowing them to incorporate domain experience in to the template construction, but also providing guidelines on best practice.

7 References

Allen, J. F. (1983): Maintaining knowledge about temporal intervals. *Communications of the ACM* **26**: 832 - 843.

Barták, R. (1999): Dynamic Constraint Models for Planning and Scheduling Problems. *Lecture Notes In Computer Science*; Vol. 1865, Selected papers from the Joint ERCIM/Compulog Net Workshop on New Trends in Constraints, Springer-Verlag London, UK**:** 237 - 255.

Beaumont, M., A. Sattar, et al. (2001): Solving Overconstrained Temporal Reasoning Problems. *14th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, Springer-Verlag.

Bettini, C., X. S. Wang, et al. (2002): Temporal Reasoning in Workflow Systems. *Distributed and Parallel Databases* **11**(3 (May 2002)): 269 - 307\6.

Combi, C. and G. Pozzi (2003): Temporal Conceptual Modelling of Workflows. *Lecture Notes in Computer Science*, Springer-Verlag. **2813:** 59 - 76.

Cook, S. A. and D. G. Mitchell (1997): Finding Hard Instances of the Satisfiability Problem: A Survey. *The DIMACS Workshop on Satisfiability Problems*, American Mathematical Society.

Crampton, J. (2004): On the satisfiability of authorization constraints in workflow systems. RHUL--MA--2004--1, Department of Mathematics, Royal Holloway, University of London.

Dechter, R. (2003): *Constraint Processing*, Morgan Kaufmann Publishers.

Dechter, R., I. Merir, et al. (1991): Temporal Constraint Networks. *Artificial Intelligence* **49**: 61 - 95.

Drakengren, T. and P. Jonsson (1996): Maximal Tractable Subclasses of Allen's Interval Algebra: Preliminary Report. *AAAI/IAAI.* **Vol. 1:** 389-394.

Drakengren, T. and P. Jonsson (1997): Towards a Complete Classification of Tractability in Allen's Algebra. *IJCAI:* 1466-1475.

Eder, J., W. Gruber, et al. (2000): Temporal Modeling of Workflows with Conditional Execution Paths. *Database and Expert Systems Applications: 11th International Conference, DEXA 2000*, London, UK, Springer-Verlag GmbH.

Eder, J., E. Panagos, et al. (1999): Time Constraints in Workflow Systems. *Advanced Information Systems Engineering: 11th International Conference, CAiSE'99*, Heidelberg, Germany,, Springer-Verlag GmbH.

Jablonski, S. and C. Bussler (1996): *Workflow Management - Modeling Concepts, Architecture and Implementation*, International Thomson Computer Press.

Jeavons, P. (1999): Constructing Constraints. *The 4th International Conference on Principles and Practice of Constraint Programming,* Springer-Verlag.

Jonsson, P., T. Drakengren, et al. (1996): Tractable subclasses of the point-interval algebra: A complete classification. *KR'96*, AAAI Press/The MIT Press.

Li, H. and Y. Yang (2004): Dynamic checking of temporal constraints for concurrent workflow. *Electronic Commerce Research and Applications* **4**(2005): 124-142.

Li, H. and Y. Yang (2004): Verification of Temporal Constraints for Concurrent Workflows. *Advanced Web Technologies and Applications, 6th Asia-Pacific Web Conference, APWeb 2004*, Hangzhou, China, Springer.

Marjanovic, O. (2000): Dynamic verification of temporal constraints in production workflows. *Proceedings of 11th Australasian Database Conference, 2000. ADC 2000*.

Marjanovic, O. and M. E. Orlowska (1999): Dynamic Verification of Absolute and Relative Deadline Constraints in Production Workflows, Technical Report, No. 446, Department of Computer Science and Electrical Engineering, University of Queensland, Australia.

Meiri, I. (1995): "Combining qualitative and quantitative constraints in temporal reasoning." *Artificial Intelligence* **87**: 343 - 385.

Nebel, B. and H.-J. Buckert (1995): "Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra." *Journal of ACM* **42**(1): 43 - 66.

Sadiq, W. and M. E. Orlowska (1999): On Capturing Process Requirements of Workflow Based Business Information System. *3rd International Conference on Business Information Systems (BIS '99)*, Poznan, Poland, Springer-Verlag.

Sadiq, W. and M. E. Orlowska (2000): "Analyzing Process Models using Graph Reduction Techniques." *Information Systems* **25**(2): 117 - 134.

Schwalb, E. and L. Vila (1998): Temporal Constraints: A Survey. *Constraints* **3**(2 - 3): 129-149.

Tan, K., J. Crampton, et al. (2004): The consistency of task-based authorization constraints in workflow systems. *Proceedings of 17th IEEE Computer Security Foundations Workshop, 2004*.

Tsang, E. P. K. (2003): Constraint Satisfaction in Business Process Modelling, http://www.econ.uba.ar/servicios/publicaciones/journal7/tsang.htm, Accessed 2005.

van Beek, P. (1989): Approximation algorithms for temporal reasoning. *The 11th International Joint Conference on Artificial Intelligence*, Detroit, Mich, Morgan Kaufmann.

van Beek, P. (1990): Exact and Approximate Reasoning about Qualitative Temporal Relations. Dept. of Computer Science, University of Alberta, PhD Thesis.

van Beek, P. (1992): "Reasoning about Qualitative Temporal Information." *Artificial Intelligence* **58**: 297 - 326.

van der Aalst, W. M. P. (1996): Three Good reasons for Using a Petri-net-based Workflow Management System. *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96):* S. Navathe and T. Wakayama. Camebridge, Massachusetts: 179 - 201.

Vilain, M., H. Kautz, et al. (1989): Constraint propagation algorithms for temporal reasoning: a revised report. *Readings in qualitative reasoning about physical systems*. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc: 373--381.

WfMC (1998): Interface 1: Process Definition Interchange, Process Model, Workflow Management Coalition.

Yokoo, M. (2001): *Distributed Constraint Satisfaction*, Springer.