Automated Conflict Detection Between Medical Care Pathways

P. Weber†, J. B. F. Filho†, B. Bordbar†, M. Lee†, I. Litchfield‡, R. Backman‡

†School of Computer Science, University of Birmingham, B15 2TT, UK
‡Institute of Applied Health Research, University of Birmingham, B15 2TT, UK

SUMMARY
Clinical guidelines specify sequences of steps (care pathways) to treat patients with single conditions. Increasingly many patients exhibit ‘multimorbidity’, several chronic conditions needing concurrent treatment. However, applying multiple guidelines in parallel can lead to conflicts, e.g. between prescribed drugs, lifestyle intervention recommendations, or treatment schedules. In computer science, process languages used to design and reason about software development and business process management are similar to clinical pathways. Using formal model transformation, composition and analysis methods, models can be combined and conflicts detected and resolved. We propose BPMN+V, a data-driven formal model for clinical care pathways, as an extension of BPMN. We describe a method for conflict detection using a transformation of BPMN+V to Coloured Petri Nets, and a state-space method for detection of conflict in composed models. We present results from a case study, showing that common conflicts are successfully detected, and propose extension to a complete framework for efficiently recommending resolutions to medical conflicts in composed care pathway models. Copyright © 2017 John Wiley & Sons, Ltd.

Received …

KEY WORDS: BPMN, Workflow models, Coloured Petri Nets, Model Transformation, Clinical Guidelines, Care Pathways, Conflict Detection, Multimorbidity

1. INTRODUCTION

The Business Process Model and Notation [1] (BPMN) is the de facto and also ISO† standard [2] for process modelling, providing support for modelling control flow, data flow and resource allocation. BPMN’s intuitive graphical model [3] is particularly suitable for capturing business processes by domain experts who may not have development skills [4, 5, 6]. The ability to handover BPMN specifications to automatically assist execution via languages such as Business Process Execution Language (BPEL [7]) reduces the time and cost from the design of a business process to its production. As a result, BPMN is widely adopted [8] within industry‡ and via various open source Business Process Management (BPM) tools. BPMN has been widely used in various application domains [9], including government [10], software development [11] and service management [12], construction [13], education [14], and healthcare (e.g. [4, 5, 6, 15, 16, 17]).

Clinical guidelines document the best available evidence for care of patients with specific medical conditions (‘morbidities’). In the United Kingdom (UK) they are used in combination with national guidance and local National Health Service (NHS) policy to provide appropriate care in a local
context. Guidelines have to date been published by the National Institute of Clinical Excellence [18] for 253 individual morbidities (e.g. Type Two Diabetes Mellitus [19] and Chronic Obstructive Pulmonary Disease (COPD) [20]). Flowchart-style care pathways can be generated from these guidelines and have been found to improve patient outcomes [21]. These are informal in the sense that they are not endowed with a rigorous mathematical semantics.

Patients with multiple concurrent chronic conditions (referred to as ‘multimorbidity’) are rapidly rising in prevalence in the UK [22]. Guidance relating to the management of these patients is limited [23, 24], since guidelines focus on single morbidities, and when new treatments or interventions are assessed for effectiveness within a clinical trial setting, co-morbidities are excluded where possible to limit confounding the findings. Therefore it is challenging to form guidance tailored to a patient as an individual, rather than focussed on each clinical condition in isolation [25, 26]. In this work we investigate extending BPMN to model medical guidelines, as a vehicle for studying in a patient-focussed manner the interactions between treatments or interventions when the patient is suffering from multiple morbidities.

BPMN models are analysable, and suitable restrictions of the notation can be endowed with a strong mathematical semantics (e.g. [27, 28, 29, 30, 31, 32]). Most research has focused on the control flow aspects of BPMN, whereas analysis of data flow aspects of BPMN is a less studied topic — in BPMN the semantics of handling data remain unspecified and open to interpretation [33, 34]. In recent years modelling of data objects and mapping them to Petri nets has received considerable attention (e.g. [33, 35]): such research paves the way for analysis of BPMN models. However, to the best of our knowledge existing research focuses on single BPMN models. We address the question of detecting execution paths in two BPMN models which violate (or not) a set of logical constraints. For example, consider two BPMN models that represent two medical guidelines applied concurrently to a given patient. Medications are often prescribed through both pathways, but some of these medications cannot be prescribed simultaneously. How can we automatically detect the execution paths (treatment steps) that can use a combination of drugs with minimal conflict?

We present a method for detecting execution paths in two BPMN models that violate a given set of constraints. The outline of our approach is as follows: Firstly, we propose a data-enriched subset of BPMN, named BPMN+V, appropriate for modelling clinical guidelines and based on the Workflow Graphs proposed by Vanhatalo et al. [36]. Extending the semantics of Workflow Graphs to include data models, we obtain a data-rich execution semantics for our subset of BPMN. Secondly, we outline a transformation from BPMN+V to Coloured Petri Nets (CPN) [37] which provide a formalisation of BPMN+V to facilitate analysis. Thirdly, CPN models corresponding to two guidelines are composed and enhanced with logical constraints representing potential conflicts, such as drug incompatibility, between the associated data models. Finally, the state space analysis properties of CPN (such as deadlock detection) enable detection of conflicts in the composed model.

The paper is organised as follows. After reviewing background material in Section 2, in Section 3 we propose a simple data model for BPMN to facilitate the problem outlined in Section 4, of detecting conflict between care pathways. We describe the proposed approach in several parts in Section 5: the data-enriched BPMN model and associated semantics in Section 5.1 and following; mapping BPMN+V to CPN in Section 5.9; composition of clinical guidelines mapped as CPN, and detection of conflicts, in Sections 5.10 and 5.11. A case study (Section 2.1) is used to guide the discussion throughout the paper. The study uses two self-contained fragments of medical guidelines, for Osteoarthritis and COPD, which demonstrate conflicts between drug and lifestyle advice interactions. The final evaluation of the case study is in Section 6. Finally, in Section 7 we outline future research to extend this framework to enable many different types of conflict to be detected efficiently using logical analysis and constraint solvers such as Alloy [38] or Z3-SMT [39].

2. BACKGROUND

In this section, we present reference material on which our approach builds. We first describe our case study modelling conflicts between two care pathway fragments. We then give a brief background on business processes in general and clinical guidelines and care pathways in particular.

"
Automated Conflict Detection Between Medical Care Pathways

Finally we review the Workflow Graph notation proposed by Vanhatalo et al. [36], which can be viewed as a subset of BPMN notation; and Coloured Petri Nets.

2.1. Case Study

To illustrate the proposed concepts and techniques, we have modelled two clinical care pathway excerpts using BPMN 2.0 notation [34]. These are illustrated in Fig. 1 and briefly described here. The review of medication depicted in the models would form part of a 10 minute review appointment with a General Practitioner (GP) in the UK, for patients with COPD [40] (top) and Osteoarthritis [41] (bottom). The pathways have been mapped from the NICE guidance as representative fragments of much larger models describing the treatment of these diseases.

COPD Medication Review: The review takes place only if the patient reports breathlessness. Four medications (roflumilast, mucolytics, theophylline and corticosteroids) are then reviewed in parallel, meaning there is no restriction on the order in which the reviews are carried out. In the case of theophylline, there are further criteria constraining prescription, and plasma level monitoring needs to be arranged. For corticosteroids, several further process steps are triggered. First, the reason for the drug requirement is established, then prescription is arranged within the constraint of keeping dosage as low as possible. Finally, dependent on the patient’s age, prophylactics may be prescribed with or without monitoring for osteoporosis.

Osteoarthritis Treatment Review: This review proceeds in four stages: assess and then review core treatments, review pain relief, then review non-pharmalogical treatments. In the first stage, written information, the exercise plan, and weight loss interventions are reviewed together (no restriction on the order of the activities). The second stage involves the review of four treatments (topical capsaicin, pain relief, intra-articular injections, and NSAIDs). The third stage assesses whether further pain relief is required, and indicates considerations if so.

If these pathways are followed concurrently, as when treating a patient diagnosed with both Osteoarthritis and COPD, then a conflict occurs. The British National Formulary [42] identifies that corticosteroids, which may be prescribed in the COPD pathway, and non-steroidal anti-inflammatory drugs (NSAIDs), which may be prescribed for Osteoarthritis, are in conflict. If
prescribed together they may lead to serious complicating symptoms. It is therefore of great importance that such interactions are discovered and avoided. Given that these example pathways are fragments of much larger care pathways extracted from the clinical guidelines, an automated method is crucial for doing so, and also for providing resolutions. Conflicts also exist between theophylline and roflumilast prescription in the COPD model, and potentially between the dependencies on ‘breathlessness’ in the two models.

2.2. Business Processes

Business processes describe activities carried out to fulfil a business function, and the relations between them [43], known as ‘control-flow’. Such functions are varied, including service, financial or customer management, software development, or in healthcare, treatment of patients. Defining processes facilitates understanding, and managing the complex interactions between activities and resources can help demonstrate adherence to regulations, or increase efficiency. Various representational mechanisms have been suggested for capturing process control flow. These range from formal languages such as Petri nets [44, 45, 46] or BPMN [34] which allow systematic analysis and comparison, to flowchart notations used to informally discuss business processes, such as the care guidelines for COPD [40] or Osteoarthritis [41].

Petri nets are widely used to model concurrent and distributed systems, and are rigorously defined, whereas BPMN although standard and very flexible, is not fully formalised [47]. So-called Workflow models are therefore now widely used for specifying business processes [46, 48, 49, 50]. These restrict the structure and behaviour of the models to a subset adequate for specifying business processes with well-defined behaviour and properties for analysis. Focusing on the analysis of the systems, Van der Aalst et al. [46, 48] present a Workflow modelling language in which models are constructed from blocks of Petri net models representing common workflow constructs.

2.3. Clinical Guidelines and Care Pathways

Formation of clinical guidelines is complex [51, 52, 53], requiring critical appraisal of evidence from many sources such as systematic reviews and clinical trials. Care pathways distilled from the guidelines, used in clinical decision support systems, present as large and complex networks of many activities, typically broken down into interacting sub-processes (e.g. [40, 41]). Features of care pathways include steps for assessment, treatment and review, as well as links to related pathways. Diverse options may be given for treatment and advice to help improve a condition, including for example lifestyle recommendations, exercise specifications or dietary advice. Care pathways also detail medication options, prescribing the clinically appropriate medication both for the presenting complaint and also for prevention of other conditions for which risk factors have been noted.

Another major element within the pathway relates to specialist services referrals, which could be within secondary- or tertiary-centred care. After being referred, the patient will be under a different section of the pathway, specific to that healthcare sector. For example, some medications can only be prescribed under specialist rather than generalist care. All of these decisions will be noted within the pathway. Furthermore, the decisions have the ability to trigger different sections of the pathway, creating the need to go back and re-investigate earlier options within the path.

The implementation challenges introduced by such complex processes, and increasing time pressures within the NHS have promoted the need to look outside the healthcare sector for solutions. Computer Scientists have investigated novel solutions to problems associated with healthcare pathways and clinical guidelines [54], for example using string metrics to identify how patient journeys differ from the prescribed care pathway [55]. This is particularly useful for interrogation of the patient journey to identify real world management. Natural Language Processing (NLP) has also been used to help with clinical decision support systems [56], extract information from clinical guidelines [57] and electronic health records [58], and to test improvement to health such as a reduction of opioid prescribing [59].

2.4. BPMN for Modelling Clinical Pathways

Many healthcare studies have employed BPMN to model clinical pathways (CPs) [4, 5, 6, 15, 16, 17]. Several authors have investigated the use of BPMN in healthcare [16, 17], and concluded that it is “sufficiently suitable for the planned modelling and imaging of CPs” [3], and its prevalence is increasing [16]. Key benefits of BPMN are stated as being graphically clear and appealing [3] and designed to facilitate communication between non-specialists [4, 5, 6]. BPMN has been criticised because its semantics are under-specified and models are not guaranteed to be interoperable between systems [60]. However it provides for extensions [16, 61], for example to use colour to enhance comprehensibility of complex clinical processes [5]. Formal semantics for subsets of BPMN [34] have been proposed including via transformation to Petri nets [47] or YAWL\(^\parallel\) [29, 30, 31].

Formalisms such as Petri nets [44, 45, 46] and YAWL [62] do enjoy fully specified semantics, including (for YAWL) interaction with data. Arguably this is at the expense of the graphical clarity and interpretability of BPMN. They have also been criticised [60] as too restrictive for modelling ‘real’ business processes. Computer-Interpretable Guidelines (CIGs) have also been developed as part of clinical decision support systems (DSS) [63], such as PROforma [64], Arden Syntax [65] and GLIF [66]. DSS provide a complete environment for process automation and decision support, including comprehensive supporting clinical information. The interaction with data of YAWL and PROforma et al. is more complex than required for the conflict detection discussed here, and closely entwined with the process engine and decision support behaviour. Domain-specific languages may also risk limited acceptance and interoperability issues [67].

Since we are concerned here with modelling clinical guidelines in an accessible and human-interpretable way rather than process automation, we base our notation on BPMN for the benefits of widespread acceptance, ease of interpretation and extension described above. We hope that in the future our methods may be extended to other data-aware or domain-specific modelling languages.

2.5. Workflow Graphs

We find the modelling language suggested by Vanhatalo et al. [36], based on Petri nets, the closest to the style of BPMN 2.0 or major software development tools such as Oracle JDeveloper\^\(\parallel\). Our model (Section 5.1) builds on their formalism, outlined next.

Definition 1

A **Workflow Graph** is a graph \(G = (N, E)\), with a set of nodes \(N\) connected by a set of edges \(E \subseteq N \times N\). Each node represents an activity or a control-flow construct from the set \{START, STOP, ACTIVITY, FORK, JOIN, DECISION, MERGE\}.

\(\parallel\)Yet Another Workflow Language [62].
\^\(\parallel\)http://www.oracle.com/.
These relate to the core control-flow elements used in BPMN 2.0 (Fig. 2), with the exclusion of ‘inclusive’ gateways (a large circle \( \bigcirc \) in a diamond), which we define in our formalism (Section 5.1). Small circles indicate \textsc{start} and \textsc{stop} nodes; rectangles with rounded corners, \textsc{activity} nodes; diamonds containing a large \( X \), \textsc{decision} or \textsc{merge}; containing a large \( + \), \textsc{fork} or \textsc{join}.

\textbf{Notation 1}  
A Workflow Graph \( G \) is well-formed \cite{47} by definition:

- \( G \) has a unique \textsc{start} node \( i_G \) with a single output edge and no input edges, and a unique \textsc{stop} node \( o_G \) with a single input edge and no output edges;
- each node \( n \in N \) has a set of input and output edges \( I(n) \in E \) and \( O(n) \in E \);
- each \textsc{activity} node has a single input and a single output edge;
- each \textsc{fork} and \textsc{decision} node has a single input edge and two or more outgoing edges; and
- each \textsc{join} and \textsc{merge} node has two or more input edges and a single output edge.

An \textsc{activity} node models an atomic (indivisible) unit of work. A \textsc{decision} node models choice between alternate sequences of activity following the node (\textsc{merge} nodes model the end of the alternate paths). A \textsc{fork} node models sequences of activity which may happen in parallel following the node (\textsc{join} nodes model the end of the parallel paths).

\textbf{Definition 2}  
The \textit{behavioural semantics} of Workflow Graphs is described as a ‘token game’, similar to the semantics of Petri nets \cite{44, 45}. The flow of tokens through the graph indicates the progress of \textit{instances} of the process execution. See Vanhatalo et al. \cite{36} for full definitions.

\textbf{Notation 2}  
A \textit{state} \( s \) of a Workflow Graph \( G = (N, E) \) is a mapping \( s : E \rightarrow \mathbb{N} \) assigning tokens to edges \( E \). We write \( s(e) = k \) to indicate that in state \( s \) edge \( e \) carries \( k \in \mathbb{N} \) tokens. An execution of node \( n \in N \) results in changing the state of \( G \) from \( s \) to \( s' \), denoted \( s \xrightarrow{n} s' \).

Informally, the state \( s \) of Workflow Graph \( G \) controls which nodes can be executed, while the semantics describe the change in state resulting from execution of a node. Execution of an node in \( G \) results in the movement of tokens between the edges to capture the flow of actions. An \textit{instance} of the process (e.g. treatment review for a given patient) is started by executing the \textsc{start} node \( i_G \), when a single token is added to its output edge \( O(i_G) \). There is no change to the allocation of tokens to other edges in the graph. Multiple process instances may be executing concurrently, thus \( O(i_G) \) may carry more than one token. Executing the \textsc{stop} node \( o_G \) removes a token from its input edge, denoting completion of the instance. A node \( n \) of type \{\textsc{activity}, \textsc{fork}, \textsc{merge}\} can be executed if each input edge in \( I(n) \) has one or more tokens. One token is then removed from each edge in \( I(n) \), and one token is added to each edge in \( O(n) \). If \( n \) is a \textsc{decision} node then a token is removed from the single \( I(n) \), and added to just one output edge in \( O(n) \), chosen non-deterministically. If \( n \) is a \textsc{join} node, a single token is removed from one edge in \( I(n) \), chosen non-deterministically from all those bearing one or more tokens, and passed to the single output edge \( O(n) \).

\subsection{2.6. Coloured Petri Nets}  
Coloured Petri Nets (CPN) \cite{37} are commonly used to model concurrent systems and analyse their properties, particularly when data is involved. CPN extends the Petri net formalism with high-level programming language capabilities, enabling definition of data attached to the process and interaction between data and process behaviour. As such, they are appropriate for our need to analyse the interaction between multiple processes (care pathways), where process behaviour is strongly driven by data (patient characteristics).

\textbf{Definition 3}  
A \textbf{Coloured Petri Net} is a bi-partite graph specified by a tuple \( C = (P, T, A, \Sigma, V, S, G, E, l) \), where

- \( P \) is a finite set of \textit{places}, \( T \) a finite set of \textit{transitions} such that \( P \cap T = \emptyset \), and \( A \subseteq (P \times T) \cup (T \times P) \) a set of directed arcs. \( N = (P, T, A) \) is the Petri net structure of \( C \).
The behavioural semantics of CPNs are governed by the flow through the net of data-enriched coloured tokens, which are associated with variables in $V$, controlled by the interaction of $V$ with the types $S$ of places and guard expressions $G$ on transitions and $E$ on arcs.

We use a subset of CPN to formalise our data-enriched BPMN (BPMN+V). We introduce some CPN concepts as necessary in later sections, and refer the interested reader to Jensen and Kristensen [37] for full definitions of the syntax and semantics of CPN. Suitably restricted subsets of BPMN and Petri nets can be defined with equivalent semantics, and a model transformation between them has been defined by Dijkman et al. [47]. We base a similar transformation on this, between BPMN+V and CPN, outlined in Section 5.9. CPN have been used as the formal target model for the underlying semantics of other models of concurrency, such as Sequence Diagrams (e.g. [68]).

3. A SIMPLE DATA MODEL FOR BPMN

Data is a core asset of any organisation, and plays a key role in business processes, forming the basis of many decisions. In the care pathway in Fig. 1 the decision to prescribe prophylactics or not in a COPD patient is taken on the basis of the age of the patient. In BPMN the primary construct for modelling data within the process flow is the data object element [34, p205]. However, the semantics of data objects remain unspecified and even left to the interpretation of the modellers [33, 34]. Moreover, in BPMN the behaviour of data objects is decoupled from the control-flow. Data aspects of BPMN have received considerable attention recently [69, 70, 71]. Meyer et al. evaluate business process modelling languages with respect to data modelling and investigate modification of data by activities, events, gateways and control-flows [69]. They also present a set of algorithms for extracting such data models [70]. Sun et al. [71] present an elaborate formalism of the interaction between business processes and databases. They deal with crucial issues such as isolation, or when the execution of one process or instance must not interfere with that of another.

3.1. Data in Care Pathways

Each clinical guideline contains many tests, recommendations and actions covering all aspects of patient management for a specific condition. All involve data, whether pertaining to the patient characteristics, clinical aspects, or the environmental context. These data may guide the ‘flow’ of the patient through the care pathway (e.g. decisions taken on the basis of age or a blood test), and may themselves be modified by actions taken (e.g. necessity for further pain relief). For example, lifestyle advice may include data on minimum duration, frequency, and intensity of exercise, and depend upon patient age and specific symptoms; dietary advice includes examples; and medication recommendations depend on the presenting complaint but include consideration of cost and patient preference, as well as the context (generalist or specialist care) within which care is being provided. Data-driven decisions and data effects of activities may also trigger different sections of the pathway, or modify previous data attributes, causing earlier parts of the pathway to be revisited.

3.2. Modelling Care Pathway Data in BPMN

The Workflow Graph subset of BPMN we use for modelling the control-flow structure underlying care pathways, corresponds closely to Workflow nets [36, 47]. It is therefore natural to draw on
Coloured Petri Net (CPN) concepts to enrich our models with data. The formalism of coloured tokens carrying data attributes through the model, interacting with data-defined guards on elements of the net, is appropriate for modelling patients with given characteristics interacting with treatment activities in a particular clinical context. We assign to each token within a BPMN+V model a colour in the form of a vector of values, to each node a condition guarding execution of the node, and define implicit data which may affect the process execution.

**Notation 4**
Assume a fixed set of $d$ variables $X = \{x_1, \ldots, x_d\}$ of types $\mathcal{T}(x_i) \in \{T_1, \ldots, T_m\}$. We model the data associated with a process instance as a $d$-tuple of valuations $V = (v_1, \ldots, v_d)$, assigning values to $X$ as the business process executes and tokens flow through the net.

Data types include \{boolean, integer, rational, enumerated (categorical)\}. For example, in the COPD treatment process (Fig. 1) we might use boolean variables to record breathlessness and prescription of the various medications; enum. to record reasons for corticosteroid prescription and the age category of the patient; rational for plasma level; and integer for approximate weight.

**Remark 1**
Since we deal with well-defined models (Notation 1), the set of variables $X$ is fixed and will not change. Our model is therefore a simplification of CPN.

**Notation 5**
A condition is a first order logic formula $c(\cdot)$ over variables in $X$. We define pre and post conditions on nodes, as guards controlling when a node may be executed or may complete. For convenience we also define edge conditions equivalent to a pre condition on the following node:

**Notation 6**
A node condition on node $n$ is a pre-condition $\text{pre}(n)$ such that $\text{pre}(n) = c(\cdot)$, controlling when $n$ may be executed, or a post-condition $\text{post}(n)$ controlling when $n$ may be considered complete. We write $c(\cdot) \models V$ if the variable valuation $V$ satisfies $c(\cdot)$. For example, if $n$ is an activity $a$ representing a treatment only valid for patients over 55 years of age, and we have a variable $x_i$ indicating patient age in years, then $\text{pre}(a)$ would be $c(x_1, \ldots, x_d) \equiv (x_i > 55)$.

**Notation 7**
An edge condition is a convenience when considering nodes with multiple output edges (i.e. diverging gateways). $c(e_{\text{out}})$ corresponds to the pre-condition of the node following the edge, i.e. $c(e_{\text{out}}) = \text{pre}(n) \mid n \in N \land I(n) = e_{\text{out}}$.

**Notation 8**
A data modification is a statement $f(\cdot)$ over variables in $X$ describing the effect on data of the execution of an activity, e.g. $x := x + 1$ or $x := \text{False}$. We loosely define $D$ as the set of all valid such data modification functions.

**Notation 9**
Assume that a process interacts with a database $D$. We consider data in $D$ to be implicit to the process, involved in defining the control-flow, but unchanged by it. For example, a database of drug interactions [42] describes constraints on the context in which a drug may be prescribed.

In Section 5.1 we describe an extension of the semantic model of Vanhatalo et al. by describing the changes in the value of colours as the tokens flow through the (data-enhanced) BPMN+V model. We believe our method will limit the anomalies between data and semantics (see e.g. [33]) through introducing a clear semantics which marries data and control flow. Explicit modelling of the interaction between data and control-flow enables a transparent view on how process execution is constrained by, and affects, patient characteristics, and implicitly depends on external attributes such as medication interactions and resource scheduling.
4. PROBLEM DESCRIPTION

The problem we address is that of identifying conflicts between clinical care guidelines when they are followed concurrently in treating patients with multiple morbidities. Assume that we have two BPMN+V models $M_1, M_2$ representing two guidelines, and these interact with a database $D$. Suppose further that each model has a set of relevant variables $X_1, X_2$ some of which are shared between the models. We collect the variables into a set $X = \{x_1, \ldots, x_d\} = X_1 \cup X_2$. For example, assume these variables represent the medication taken when a patient is treated by two pathways (cf the example in Section 2.1). If the patient is on medication $m_i$, then the corresponding variable $x_i$ is set True. Knowledge of drug interactions (defined in $D$) indicates that some combinations of medication are not permitted. We write this as a constraint over the variables.

**Notation 10**

Suppose that the medication related to variable $x_i$ should never be taken with the the medication related to $x_j$, we write this as a constraint $C_r(x_1, \ldots, x_d) \equiv \neg(x_i \land x_j)$. We assume that we know all $k$ possible constraints on interactions between variables in $X$, $C = \{C_1, \ldots, C_k\}$. Although conflicts of this type between $M_1$ and $M_2$ could be identified by checking the values $V$ assigned to $X$ against the set of constraints $C$, it is more useful to pin-point which execution paths in the models are in conflict. We therefore pose the question in this form:

**Question**

Given two BPMN+V models $M_1, M_2$, variables $X$ and a set of constraints $C$, identify all pairs of execution paths which will modify the variables in $X$ so that at least one of the $C$ is violated.

5. APPROACH

Our approach to detecting execution paths in conflict between two or more clinical guidelines follows several steps, detailed in the following sections:

1. define a process modelling language BPMN+V to capture control flow for clinical guidelines, and its dependencies and effect on data;
2. translate models described as BPMN+V to CPN to facilitate formal analysis;
3. compose CPN models representing multiple clinical guidelines, adding constraints between the models; and
4. solve the composed model using CPN state space methods to identify points of conflict such as deadlocks.

**Remark 2**

The translation to CPN is not strictly necessary, since the BPMN+V notation is consistent and complete for our purposes, but it allows us to take advantage of the existing theory of CPN analysis.

**Remark 3**

This is the first introduction of our method. In the future we plan to replace the CPN analysis with translation to logical constraints and efficient analysis using SAT and SMT solvers.

5.1. Semantics For BPMN With Data

We extend the Workflow Graph formalism ([36] and Section 2.5) to allow for the flow of data through the graph, using a form of colour for tokens. Our notion of data is defined in Section 3.2. We refer to this formalism as BPMN+V.

**Definition 4**

A Workflow Graph extended with data (BPMN+V) is a tuple $G = (N, l, E, X, \text{pre}, \text{post}, \text{mod})$:

- $N$ is a finite set of nodes (BPMN elements);
- $l : N \rightarrow \{\text{START, END, ACTIVITY, EXCLUSIVE, INCLUSIVE, PARALLEL}\}$ is a relation assigning each node a fixed type;
• $E \subseteq N \times N$ is a finite set of edges (sequence flows) connecting nodes;
• $X = \{X_1, \ldots, X_d\}$ is a finite set of $d$ variables associated with the process;
• $\text{pre} : N \rightarrow C$ is a set of guards defining node pre-conditions, as defined in Notation 5;
• $\text{post} : N \rightarrow C$ is a set of guards defining node post-conditions; and
• $\text{mod} : N \rightarrow D$ is a set of data modifications enacted by ACTIVITIES.

The execution semantics of BPMN+V are described by the flow of tokens through the model.

Notation 11
Let $m : E \rightarrow \{T_1, T_2, \ldots\}$ define a marking defining the state of the process, mapping each edge to a set of coloured token IDs. $m(e)$ defines the number of tokens on edge $e \in E$ in state $m$. Each token $T_i$ is a pair $(t_i, V_i)$, with $\tau(T_i) = t_i$ an ID unique to process instance $i$, $V(T_i) = V_i$ a valuation of the variables $X$ as defined in Notation 4. Executing node $n \in N$ changes the state from $m$ to $m'$, denoted $m \xrightarrow{n} m'$, as $n$ consumes and produces tokens according to the semantics, and may modify the values $V_j$. Execution of a sequence of nodes is indicated by $m \xrightarrow{e} m'$.

Each instance of a process carries a single coloured token, which may be split by diverging PARALLEL nodes to describe parallel sequences of activity, and merged by converging PARALLEL nodes. The colour is synonymous with the assigned data values. The valuations affect process control flow by interacting with guards (pre and post) on nodes, and may be changed by executing ACTIVITY node mod statements, as the business process executes and tokens flow through the net.

Notation 12
In the following, unless otherwise indicated, $T$ denotes a token carrying a $d$-tuple of valuations (colour) $V$ before execution of the node under consideration. $T'$ denotes the same token following execution, with possibly modified values $V'$.

5.2. Start Event

A Start event is a node $n \in N$ that captures the initialisation of a process instance, by creating a coloured token $T'$ on its unique output edge. For example, a patient may be registered on a programme of treatment. Any BPMN+V model has exactly one Start event. Assume $V$ captures initial values of all data attributes associated with the process instance to be created, then $n$ can execute (thus the process instance can start) if $\text{pre}(n) = V$. The effect of executing $m \xrightarrow{n} m'$ is:

1. $V' = V$, and
2. $m'(e) = \begin{cases} m(e) \cup \{T'\} & \text{if } e \in O(n), \\ m(e) & \text{otherwise}. \end{cases}$

Note that $m'(e) = m(e) \cup \{T'\} \Rightarrow |m'(e)| = |m(e)| + 1.$

5.3. End Event

An End event $n$ brings the process to a conclusion, e.g. the patient no longer exhibits symptoms of the morbidity under consideration. Any BPMN+V model has exactly one End event. $n$ can execute if $\exists T = (t, V) \in m(I(n)) \mid \text{pre}(n) = V$. The effect of executing $m \xrightarrow{n} m'$ is:

1. $V' = V$, and
2. $m'(e) = \begin{cases} m(e) \setminus \{T\} & \text{if } e \in I(n), \\ m(e) & \text{otherwise}. \end{cases}$

Note that $m'(e) = m(e) \setminus \{T\} \Rightarrow |m'(e)| = |m(e)| - 1.$

5.4. Sequence Flow

Sequence Flow is represented by an edge $e$ between two nodes, expressing flow of execution and carrying coloured tokens. Edge $e = (a, b)$ between nodes $a, b \in N$ is an output edge of its preceding node, $e \in O(a)$, and an input edge of its following node, $e \in I(b)$. 

Copyright © 2017 John Wiley & Sons, Ltd.

5.5. **Activity Node**

An Activity node $a$ represents an atomic (indivisible) unit of work to be executed. In a well-formed BPMN+V model an Activity has exactly one input sequence flow $e_{in} \in E$, i.e. $I(a) = \{e_{in}\}$, and exactly one output sequence flow $e_{out} \in E$, i.e. $O(a) = \{e_{out}\}$. On execution, $a$ consumes a token $T = (t, V)$ from $e_{in}$, and returns $T' = (t, V')$ on $e_{out}$, with possibly modified assignment $V'$.  

**Activity** $a$ is executed if: $\exists T = (t, V) \in m(e_{in}) \mid pre(a) \models V$. 

Execution of $a$ changes the state such that $m \xrightarrow{a} m'$, where

1. $post(a) \models V'$, and
2. \[ m'(e) = \begin{cases} m(e) \setminus \{T\} & \text{if } e = e_{in}, \\ m(e) \cup \{T'\} & \text{if } e = e_{out}, \\ m(e) & \text{otherwise}. \end{cases} \]

5.6. **Exclusive Gateway**

An Exclusive Gateway $b$ models a decision point in a process. It can be either diverging: modelling a decision to be followed by one of several sequences of subsequent activity, or converging: modelling the rejoining of the alternative sequences created by the corresponding previous diverging Exclusive gateway.

**Diverging:** In a well-formed BPMN+V model a diverging Exclusive gateway $b$ has one input sequence flow $e_{in} \in E$, such that $I(b) = \{e_{in}\}$, and two or more output sequence flows $O(b) \subset \{E \setminus e_{in}\}$. $b$ consumes a token $T$ from $e_{in}$ and returns it unmodified to one of its output edges.

The gateway $b$ executes if: $\exists T = (t, V) \in m(e_{in}) \mid pre(b) \models V$.

Then $b$ is executed, $m \xrightarrow{b} m'$, and there exists a unique $e_{out} \in O(b)$ such that:

1. $V' = V$,
2. $c(e_{out}) \models V'$, and
3. \[ m'(e) = \begin{cases} m(e) \setminus \{T\} & \text{if } e = e_{in}, \\ m(e) \cup \{T'\} & \text{if } e = e_{out}, \\ m(e) & \text{otherwise}. \end{cases} \]

A single output sequence flow can be followed after a diverging Exclusive Gateway $b$, subject to satisfaction of its edge condition (the pre-condition on the next node which will be executed, cf Notation 7). If the condition of more than one sequence flow after an Exclusive Gateway is satisfied, $|\{e_{out} | e_{out} \in O(b) \land c(e_{out}) \models V'\}| > 1$, then one output flow is chosen non-deterministically.

**Converging:** In a well-formed BPMN+V model, a converging Exclusive gateway $b$ corresponds exactly to a preceding diverging Exclusive gateway $b'$ with $p$ output edges. Therefore $b$ has $p$ input sequence flows $|I(b) \subset \{E \setminus e_{out}\}| = p$. For a given process instance identified by token $T$ with ID $\tau(T) = t$, a single edge $e_{in} \in I(b)$ can be active (carrying a token with ID $t$), i.e.

$$\forall e_{in} \in I(b), T = (t, V) \in m(e_{in}) \Rightarrow \forall e_i \in I(b) \land e_i \neq e_{in} \,(T_i \in m(e_i) \Rightarrow \tau(T_i) \neq t).$$

$b$ has one output sequence flow $e_{out} \in E$, such that $O(b) = \{e_{out}\}$. $b$ consumes a coloured token from $e_{in}$ and returns it unmodified to $e_{out}$.

The gateway $b$ therefore executes if

$$\exists e_{in} \in I(b) \land T = (t, V) \in m(e_{in}) \land \forall e_i \in I(b) \land e_i \neq e_{in} \land \forall T_i \in m(e_i) \tau(T_i) \neq t.$$ 

effecting $m \xrightarrow{b} m'$ such that

1. $V' = V$,
2. $m'(e) = \begin{cases} m(e) \setminus \{T\} & \text{if } e = e_{in}, \\ m(e) \cup \{T'\} & \text{if } e = e_{out}, \\ m(e) & \text{otherwise}. \end{cases} \]
5.7. PARALLEL Gateway

A PARALLEL Gateway $b$ indicates that all output sequence flows will be activated simultaneously.

**Diverging:** In a well-formed BPMN+V model, a diverging PARALLEL gateway $b$ has one input sequence flow $e_{in} \in E$, such that $I(b) = \{e_{in}\}$, and two or more output sequence flows $O(b) \subseteq \{E \setminus e_{in}\}$. $b$ consumes a token from $e_{in}$ and duplicates it unmodified, to each output edge $e \in O(b)$.

The gateway $b$ executes if: $\exists T = (t, V) \in m(e_{in}) \mid prec(b) \models V$ and $\forall e \in O(b), c(e) \models V'$.

Then $m \xrightarrow{b} m'$ such that:

1. $T' = T$, and
2. $m'(e) = \begin{cases} m(e) \setminus \{T\} & \text{if } e = e_{in}, \\ m(e) \cup \{T'\} & \forall e \in O(b), \\ m(e) & \text{otherwise}. \end{cases}$

**Converging:** In a well-formed BPMN+V model, a converging PARALLEL gateway corresponds exactly to a preceding diverging PARALLEL gateway $b'$ with $p$ output sequence flows. Therefore $b$ has $p$ input sequence flows, $|I(b) \subseteq \{E \setminus e_{out}\}| = p$, and one output sequence flow $e_{out} \in E$, such that $O(b) = \{e_{out}\}$.

Since each input edge of $b$ concludes a particular concurrently executing sequence of activities following $b'$, the data assignments $V_i$ carried by tokens $T_i = (t, V_i)$, arriving at $b$ on edges $e_j \in I(b)$ for a given process instance, may differ. For $b$ to execute, these assignments must be compatible. They must then be synchronised to a single assignment $V'$ on $e_{out}$.

**Notation 13**

Compatible data assignments $V_1, \ldots, V_d$ on tokens arriving at $b$ are defined with respect to the assignment $Y$ on tokens leaving $b'$. Let the data assignments to tokens leaving $b'$, arriving at $b$, and leaving $b$, be denoted respectively:

- each $e_{out} \in O(b')$ carries token $S = (t, Y)$ and $Y = (y_1, \ldots, y_d)$,
- $e_j \in I(b)$ carries token $T = (t, V_j)$ and $V_j = (v^j_1, \ldots, v^j_d)$, $0 < j \leq p$, and
- $e_{out}$ carries token $T' = (t, V')$ and $V' = (v'_1, \ldots, v'_d)$.

Then $Y, V_1, \ldots, V_p, V'$ satisfy one of the following three criteria; $\forall 0 < i \leq d$,

1. $\forall 0 < j \leq p, v^j_i = y_i$; assignment to variable $x_i$ is not changed by any parallel path following $b'$; we set $v'_i = y_i$;
2. $\exists 0 < j \leq p, v^j_i \neq y_i \land \forall 0 < k \leq p, k \neq j, v^k_i = y_i$, assignment to $x_i$ is changed on one parallel path only: we set $v'_i = v^j_i$; or
3. $\exists 0 < j \leq p, v^j_i \neq y_i \land \exists 0 < k \leq p, k \neq j, v^k_i \neq y_i$, assignment to $x_i$ is changed on more than one parallel path, the data cannot be synchronised, and the gateway cannot execute.

In the case 1. and 2. the differing data assignments on each input are compatible with each other and with $Y$, denoted $\text{compat}(V_1, \ldots, V_p, Y)$, and can be synchronised by setting the elements of $V'$ as stated. We can relax the condition of equality of assignments to a suitable definition of approximate equality such as numeric values within some threshold. In case 3. the data have been changed incompatibly and cannot be synchronised: the gateway cannot execute.

Therefore $b$ can execute when each input sequence flow $e_{in} \in I(b)$ has a token with the same ID and the data assignments are compatible, i.e.

$\forall e_{in} \in I(b), T_i = (t, V_i) \in m(e_{in}) \wedge \forall e_{out} \in O(b') S = (t, Y) \in m(e_{out}) \wedge \text{compat}(V_1, \ldots, V_p, Y)$.

$b$ consumes a token from each $e \in I(b), m \xrightarrow{b} m'$, and creates a single token on $e_{out}$, such that:

1. $V' = (v'_1, \ldots, v'_d)$ s.t. $v'_i$ are assigned according to compatibility cases 1. and 2. above, and
2. $m'(e) = \begin{cases} m(e) \setminus \{T_i\} & \text{if } e \in I(b) \land T_i = (t, V_i) \in m(e), \\ m(e) \cup \{T'\} & \text{if } e = e_{out}, \\ m(e) & \text{otherwise}. \end{cases}$
5.8. Inclusive Gateway

An Inclusive Gateway $b$ models a decision in a process and can trigger more than one outgoing sequence flow. An Inclusive gateway can be either converging or diverging. It is a combination of EXCLUSIVE and PARALLEL gateways; a subset of outgoing sequence flows can be activated following a diverging gateway, paths that are activated take place in parallel, and all and only those paths that are activated must be synchronised at the subsequent converging gateway.

Diverging: In a well-formed BPMN+V model, a diverging Inclusive gateway $b$ has one input sequence flow $e_{in} \in E$ s.t. $I(b) = \{e_{in}\}$, and two or more output sequence flows $O(b) \subset \{E \setminus e_{in}\}$. $b$ consumes a token from $e_{in}$ and duplicates it unmodified to a subset of its output edges $E_O \subset O(b)$.

The gateway $b$ executes if: $\exists T = (t, V) \in m(e_{in}) \mid \text{pre}(b) \models V$. Then $m \xrightarrow{b} m'$, such that:

1. $V' = V$, and
2. $m'(e) = \begin{cases} m(e) \setminus \{T\} & \text{if } e \in I(b), \\ m(e) \cup \{T'\} & \text{if } e \in O(b) \land c(e) \models V', \\ m(e) & \text{otherwise.} \end{cases}$

Converging: If $b$ is converging, then $I(b) \subset E$ and $O(b) = \{e_{out}\}$. In a well-formed BPMN+V model, the subset $E_I \subset I(b)$ of input sequence flows to a converging Inclusive gateway, receiving tokens for a given process instance ID $t$, must correspond with the subset $E_O' \subset O(b')$ of outgoing sequence flows from the preceding diverging Inclusive gateway on which tokens with ID $t$ were created. We write $E_t \equiv E_O' \iff |E_t| = |E_O'|$ and

\[
\begin{align*}
\forall e_i \in E_I & \exists e_o \in E_O', T_o \in m(e_o) \land \tau(T_o) = t \land m \xrightarrow{e_i} m' \land T_i \in m'(e_i) \land \tau(T_i) = t, \\
\forall e_o \in E_O' & \exists e_i \in E_I, T_i \in m'(e_i) \land \tau(T_i) = t \land m \xrightarrow{e_o} m' \land T_o \in m(e_o) \land \tau(T_o) = t.
\end{align*}
\]

The gateway $b$ executes if $E_I \equiv E_O'$, and data assignments carried by tokens on $E_I$ are compatible (Notation 13). Then $b$ consumes a token from each $e \in E_I$, $m \xrightarrow{b} m'$, and a single token is created on $e_{out}$, such that:

1. $V' = (\nu'_1, \ldots, \nu'_d)$ s.t. $\nu'_i$ are assigned according to compatibility cases 1. and 2. above, and
2. $m'(e) = \begin{cases} m(e) \setminus \{T_i\} & \text{if } e \in E_I \land T_i = (t, V_i) \in m(e), \\ m(e) \cup \{T'\} & \text{if } e = e_{out}, \\ m(e) & \text{otherwise.} \end{cases}$

5.9. Transformation From BPMN+V to CPN

BPMN+V supports only a subset of the semantics of CPN, similar to the restriction of BPMN and Petri nets supported by Workflow Graphs. The transformation from BPMN to Petri nets is complicated by the relatively informal semantics of BPMN and features such as Inclusive gateways, events and message passing, none of which are directly supported by CPN. However, many Inclusive gateway structures can be modelled by combinations of EXCLUSIVE and PARALLEL gateways (e.g. [72, 73]). Events and messages are currently not part of our BPMN+V specification. We therefore use a subset of the transformations described by Dijkman et al. [47] for Place-Transition (non-Coloured) Petri nets, to transform the BPMN control structures defined in the previous sections to CPN structures. We extend them to map BPMN+V conditions and data modification statements onto CPN transition guards and arc inscriptions, as summarised below.

- **START** and **End** events are mapped to a start and end place respectively.
- **Activity** nodes map to CPN transitions with pre- and post-places as necessitated by the surrounding nodes. Associated pre-conditions are mapped to transition guards, post-conditions and data modifications to inscriptions on the outgoing arcs.
- **Parallel** and **Exclusive** gateways map to parallel and exclusive-or (XOR) splits whose semantics are governed by the appropriate use of CPN places (Fig. 2).
Inclusive gateways are modelled using a combination of parallel and exclusive splits, allowing each sequence of activities following the split to be either executed or bypassed.

5.10. Composition of CPN Models

In this work, we employ a simple parallel composition. This is achieved by preceding the start places of the two models by a ‘silent’ CPN transition and a new start place, and following the end places with a second ‘silent’ transition and new end place (Fig. 3). This allows the two models to be followed simultaneously, simulating a patient following two care pathways. This makes the simplifying assumption that a patient starts both pathways at the same time, whereas most likely they would be following one or more pathways when a new one is started. However this approach ensures maximum overlap between the models and thus for all conflicts to be detected. The method could be extended to allow models to be connected at any point, to apply to a specific situation.

5.11. Conflict Detection

To support the evaluation of the BPMN+V model, we use state-space analysis [37] techniques to identify conflicts introduced by the composition of the CPN models. The main purpose is to demonstrate the suitability of BPMN+V to describe care pathways and allow conflicts to be detected.

To simplify the evaluation, we restrict our attention to three types of conflict, described next in relation to the artificial care pathway fragments illustrated in Table I.

Data assignment. A type of conflict, or inconsistency within a model, can be created by interaction between the data valuation $V$ associated with a process instance, and the condition on a single activity $a$. For example, Table I model (1) will be blocked if the patient is already prescribed with NSAIDs since activity “Prescribe NSAIDs” cannot execute. A resolution might be to introduce a bypass, as in models (7) and (8).

Single variable. If a pair of activities $a, b$ in either a single or pair of models check and modify the same variable $x$ in similar ways, they may conflict. Models (3) and (4) in Table I for example check and set ‘breathless’ in opposing ways. Similarly, duplicating prescription of medication [74, 75, 76, 77] or tests [78, 79, 80] could both adversely affect patient safety and efficiency of the healthcare practice.

Pairs of variables. If activity $a$ has a pre-condition on variable $x_1$ and modifies $x_2$, while activity $b$ has a pre-condition on $x_2$ and modifies $x_1$, they may be in conflict. For example, two drugs which may not be prescribed together [42]: cf models (5) and (6) in Table I.

Conflict detection proceeds in two phases. We first analyse the individual care pathway models using the steps described below, to determine inconsistencies preventing either model from executing under certain data valuations. The models are then composed and the same analysis applied, excluding tokens with valuations which failed the single models.

1. Identify the $d$ variables $X$ involved in on pre- and post-conditions, and the values $V$ to be checked or assigned, e.g. condition $x < 1$ involves variable $x$ and value 1.
2. Create a ‘covering set’ of $2^d$ coloured tokens for which the model needs to be checked. For each $x_i \in X$, construct the pair of valuations $\nu_i \in V$ that will satisfy and fail each pre- and post-condition. E.g. tokens must include both True and False values for conditions $x$ or $\neg x$; and with values to satisfy both $y > 1$ and $y \leq 1$, for an integer condition $y > 1$.
Table I. Artificially-designed illustrative care pathway fragments to demonstrate conflicts within and between models (Section 6.1 and results in Table II).

<table>
<thead>
<tr>
<th>Example Model(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Single model exhibiting a problem with certain data settings. Non-steroidal anti-inflammatory drugs (NSAIDs) are prescribed but should not be over-prescribed. The associated variable is set to True on prescription, but checked first to avoid over-prescription.</td>
</tr>
<tr>
<td>(2)</td>
<td>Single model in which two activities are in conflict. Both NSAIDs and corticosteroids are prescribed (in parallel). These drugs should not be prescribed together nor individually over-prescribed, hence the guards. (Note that whereas in this illustration, these guards have been added manually, in reality they would be automatically discovered, e.g. from the BNF database [42].)</td>
</tr>
<tr>
<td>(3) &amp; (4)</td>
<td>Two care pathway fragments which exhibit conflict when combined. The first model includes prescription of an exercise plan, which should be avoided if the patient exhibits breathlessness. The second model fragment sets a variable indicating breathlessness, which will be in conflict.</td>
</tr>
<tr>
<td>(5) &amp; (6)</td>
<td>Two pathway fragments which combine the drug conflict of example model (2) with between-model conflict illustrated in example models (3) and (4). The guards and data modifications of the two prescription activities prevent the model from executing.</td>
</tr>
<tr>
<td>(7) &amp; (8)</td>
<td>An example of one way in which example models (5) and (6) might be changed to avoid conflict. The prescription activities can now be bypassed. However, the conflict detection technique presented will not discover the potential for conflict between these activities.</td>
</tr>
</tbody>
</table>
3. Construct the reachability graph $R_i$ (1 ≤ i < 2^d) of the CPN transformation of the model, corresponding to each token.

4. Identify dead markings in all $R_i$, and the corresponding blocked ACTIVITIES, conditions which failed, and the token data assignment and condition variables which are in conflict.

5. A pair of dead markings linked by common variables indicates that the corresponding ACTIVITIES conflict. The dead markings result from different paths through the model. For example, in Table I model (4), one path through leads to “Prescribe NSAIDs” before “Prescribe corticosteroids”, resulting in the latter being blocked. An alternate path leads to the two activities in the reverse order, with “Prescribe NSAIDs” being blocked. This will result in two dead markings connected by the boolean variables ‘NSAIDS’ and ‘corticosteroids’.

6. A single dead marking indicates a conflict due to initial data settings, such as an attempt to prescribe an already-prescribed medication.

We evaluate this approach in the next section. Section 7 outlines plans for more efficient analysis and determination of medically-appropriate conflicts.

6. EVALUATION

In this section we describe a three stage empirical evaluation of the modelling language and approach described in Section 5. We first apply the approach to artificially-designed care pathway fragments exhibiting conflicts, to validate that BPMN+V is sufficient and complete for modelling care pathways, and that the approach for finding conflicts between them is effective. We next investigate the efficiency and scalability of the method for conflict detection, using large numbers of randomly-generated synthetic models of various size and complexity, with randomly added conflicts. Finally we apply our method to the case study (Section 2.1), to show that the known conflicts between the medical guidelines for COPD and Osteoarthritis is successfully identified.

To facilitate the evaluation we use BPMN annotations with structured text to specify the data aspects of BPMN+V, as illustrated in Fig. 2. Variables are specified as an annotation to the START event, ACTIVITY pre-conditions and data modifications are specified by annotations beginning ‘guard:’ and ‘data:’ respectively. We do not specify any post-conditions. These restrictions are not fundamental and in future work we will develop approaches more friendly to the modeller.

6.1. Artificial Care Pathway Fragments

In Table I we illustrate artificially-designed care pathway fragments to validate the BPMN+V modelling language and the state space method for conflict detection. These are not intended to be complete standalone pathways but rather to illustrate realistic conflict situations due to (e.g.) potential over-prescription [74, 75, 76, 77]. The annotations for data dependencies and modification have been added manually, whereas in the future such constraints will be automatically discovered using references such as the British National Formulary [42]. As described in Table I, these models exhibit various data-related inconsistencies and conflicts between models, explained in Section 5.11.

The results of the conflict detection method are shown in Table II. The ‘Activity’ column indicates the ACTIVITY a found to be blocked or in conflict. ‘Data’ reports the valuation which caused the conflict, ‘Initial Data’ the valuation of the initial token. ‘Conflict Model’, ‘Conflict Activity’ and ‘Conflict Data’ list any activities and variables found to be in conflict with a. These being empty indicates that the conflict is caused by the initial token valuation.

For model (1), we correctly identify that the model will be blocked for a patient who is already prescribed with NSAIDs. Similarly, Model (2) is blocked when either NSAIDs or corticosteroids are already prescribed. These activities are also in conflict. Since they occur in parallel, either may occur first and set the variable indicating prescription of the relevant medication, which conflicts with the guard on the other activity. A similar situation occurs between ‘roflumilast’ and ‘theophylline’ in the

**Code and results can be found at https://bitbucket.org/uobmitcon/jsep2017.**
Table II. Results of applying the state-space based conflict detection method to artificial models (Table I and Section 6.1).

<table>
<thead>
<tr>
<th>Model Activity</th>
<th>Data</th>
<th>Initial Data</th>
<th>Conflict Model</th>
<th>Conflict Activity</th>
<th>Conflict Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Prescribe NSAIDs</td>
<td>NSAIDS=True</td>
<td>NSAIDS=True</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Prescribe NSAIDs</td>
<td>NSAIDS=True</td>
<td>NSAIDS=True</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Prescribe corticosteroids</td>
<td>CS=True</td>
<td>CS=True</td>
<td>2 Prescribe NSAIDs,</td>
<td>Prescribe corticosteroids</td>
<td>NSAIDS</td>
</tr>
<tr>
<td>2 Prescribe corticosteroids</td>
<td>NSAIDS=True</td>
<td>NSAIDS=True</td>
<td>2 Prescribe NSAIDs,</td>
<td>Prescribe corticosteroids</td>
<td>NSAIDS</td>
</tr>
<tr>
<td>2 Prescribe NSAIDs</td>
<td>CS=True</td>
<td>CS=False</td>
<td>2 Prescribe corticosteroids</td>
<td>NSAIDS</td>
<td></td>
</tr>
<tr>
<td>2 Prescribe NSAIDs</td>
<td>NSAIDS=True</td>
<td>NSAIDS=False</td>
<td>2 Prescribe NSAIDs</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>3 Design and agree exercise plan</td>
<td>breathless=True</td>
<td>breathless=True</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Design and agree exercise plan</td>
<td>breathless=True</td>
<td>breathless=False</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 Prescribe NSAIDs</td>
<td>NSAIDS=True</td>
<td>NSAIDS=True</td>
<td>6 Prescribe corticosteroids</td>
<td>NSAIDS</td>
<td></td>
</tr>
<tr>
<td>5 Prescribe NSAIDs</td>
<td>CS=True</td>
<td>CS=True</td>
<td>6 Prescribe corticosteroids</td>
<td>NSAIDS</td>
<td></td>
</tr>
<tr>
<td>6 Prescribe corticosteroids</td>
<td>NSAIDS=True</td>
<td>NSAIDS=True</td>
<td>5 Prescribe NSAIDs</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>6 Prescribe corticosteroids</td>
<td>NSAIDS=True</td>
<td>NSAIDS=False</td>
<td>5 Prescribe NSAIDs</td>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>7 &amp; 8</td>
<td>No conflicts detected</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

case study COPD example (Fig. 1). Model (3) is correctly identified as blocked when ‘breathless’ is True (exercise should not be prescribed) but although the conflict with the setting of this variable in model (4) has been detected, it could not be fully identified. Further work is needed to determine if this is a problem with the implementation of the method, or a feature of this artificial example. Finally, models (5) and (6) exhibit the same medication conflict as model (2), illustrating that these conflicts are detected when two care pathways are combined. Models (7) and (8) illustrate that a recommendation to resolve these conflicts might be to introduce a bypass for these tasks. This could be implemented as a series of steps to check medication and prescribe only if appropriate, mitigating the danger of unsafe prescription. For these modified models, no conflicts are detected, raising the question of whether an extended method is desirable to highlight such potential conflicts.

6.2. Efficiency and Scalability

To investigate the performance of the method for conflict detection we generated pairs of random models of increasing size and complexity, similarly to the method described for Process Log Generation [81]. Varying numbers of conflicts were inserted into one or both models. Models were generated by starting with a START–ACTIVITY–END sequence (e.g. model (1), Table I) and repeatedly randomly choosing an ACTIVITY to expand, up to a given number of structures in the target model. The chosen ACTIVITY was randomly replaced with either a sequence of two ACTIVITIES, an EXCLUSIVE diverging gateway followed by two ACTIVITIES and an EXCLUSIVE merge, or a PARALLEL gateway followed by two ACTIVITIES and a PARALLEL merge. We experimented with various probabilities for selection of each type of structure.

Fig. 4 reports statistics collected over 30 models of each type. Figs. 4(a) and 4(b) show that while the time for processing increases somewhat with the number of activities in any model, it is particularly sensitive to the amount of parallelism in the model. Fig. 4(b) shows that for composed models with a large amount of parallel activities in the original models, the time for conflict detection increases approximately exponentially with the number of structures. This is not unexpected, since the more activities are in parallel, the faster the size of the state space increases. (We do not consider models with cycles, so the state space remains finite.)

The COPD and OA examples (Fig. 1) each contain 14 activities. These are formalised from NICE pathways [40, 41] presented as hierarchies of 5-10 flowcharts of approximately 10 activities each, suggesting full models could have at least 50 activities. Studies have shown that 16% of adult
patients have more than one chronic condition [82, 83], increasing to 65% of over 65s with two or more [84]. Although performance efficiency is thus a potentially serious problem for the state space method described here, especially since the composed models are inherently parallel, these results were obtained using a basic state space calculation algorithm. As discussed in Section 7, we plan to use more efficient logical analysis methods. The suitability of the BPMN+V modelling language is not affected by these results.

Although many conflicts may exist between models – Dumbreck et al. [77] found that between 89 and 133 drug-drug interactions were possible between the guidelines for three conditions in combination with up to 11 other co-morbidities – Figs. 4(c) and 4(d) show that the method is less affected by the numbers of conflicts introduced into the model.

6.3. Case Study

In Table III we report the conflicts automatically identified for the case study introduced in Section 2.1. The top and centre sections report for the OA and COPD models individually; the lower section for the composed model. In this study we used integer variables for medications instead of boolean, to indicate the number of times a medication was prescribed.
For OA and COPD the data valuations which would block the model were correctly identified, namely the parallel activity “Agree exercise plan” in OA is blocked if the patient reports breathlessness, and NSAIDs cannot be prescribed if already being taken. Similarly for COPD, the activities to prescribe theophylline and roflumilast block execution of the model if the patient is already taking these medications. Theophylline and roflumilast may not be taken together, and this conflict within the COPD model was also discovered correctly.

To obtain a meaningful analysis of the composed model we removed the ‘breathless’ constraint from OA, since if True it blocks the OA part of the model, while if False, the COPD model is bypassed. We also added bypasses for the theophylline and roflumilast activities for similar reasons. A natural extension of this work would be automatically recommend model changes to resolve conflicts; these modifications could be regarded as implementing such recommendations.

As shown in Section 6.2 our state space exploration became impractical for large numbers of states. The reachability graph for the OA and COPD composed model contains over 11,000 states for some data assignments. Therefore we manually transferred the composed CPN model to CPN/Tools \[85, 86\] for state space analysis, and the dead markings back to our tool for analysis. Such an interface could be automated to allow efficient automatic state space exploration. The conflict between NSAIDs and corticosteroid prescription was successfully discovered.

### 7. CONCLUSION AND FUTURE WORK

We have presented BPMN+V, a data-enriched subset of the Business Process Model and Notation [1] (BPMN) suitable for modelling clinical guidelines. We defined a semantics for BPMN+V, based on Workflow Graphs [36] and Coloured Petri Nets [37], which allows the effect of data upon the guideline, and of the guideline upon the data, to be formally described. For instance, we can specify how the valuations assigned to data attributes, such as the medications prescribed to a patient, control the execution of activities, and how executing an activity (such as prescribing a medication), modifies that data. We then evaluated this model using a state space analysis approach to detecting the execution paths in two BPMN models which violate a given set of constraints.

The evaluation applied the method to artificial models, and to a real life case study using parts of the clinical guidelines for treatment of Chronic Obstructive Pulmonary Disease (COPD) and Osteoarthritis (OA). Using models designed in BPMN+V, the known conflicts were discovered successfully using the state space method. Performance analysis however showed the method to be impractical for larger models, especially those containing a high degree of parallelism (as would often be the case in a non-prescriptive guideline). This may be addressed by integrating state of the art state space analysis algorithms such as ASAP \[87\] or as implemented in CPN/Tools \[86\].
In future work we plan rather to avoid state space methods, by translating BPMN+V models into logical constraints (see e.g. [88]) to allow efficient analysis using SAT and SMT solvers such as Alloy [38] or Z3-SMT [39]. This will allow much larger models to be efficiently analysed and formally proved correct and complete. Whereas in the evaluations in Section 6 we designed the constraints into the example models, in the future relevant constraints will be automatically discovered and added to the models, using references such as the British National Formulary [42] for medication conflicts. In consultation with clinical experts, we will define a comprehensive set of potential conflicts, and methods to detect and recommend changes to mitigate them. These conflicts will include medication and lifestyle, as discussed here, but also scheduling problems relating to availability of resources such as appointments, medical personnel, and locations. Finally we plan to build the methods into a tool suitable for use by clinicians, necessitating development of a suitable user interface and natural language methods for inferring data and constraints from text.

ACKNOWLEDGEMENT

This research was funded by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant number EP/M014401/1.

REFERENCES


