Stakeholder utility measures for declarative processes and their use in process comparisons

Mark Dukes

***Abstract*—We present a method for calculating and analyzing stakeholder utilities of processes that arise in, but are not limited to, the social sciences. These areas include business process analysis, healthcare workflow analysis and policy process analysis. This method is quite general and applicable to any situation in which declarative-type constraints of a modal and/or temporal nature play a part.**

A declarative process is a process in which activities may freely happen while respecting a set of constraints. For such a process, anything may happen so long as it is not explicitly forbidden. Declarative processes have been used and studied as models of business and healthcare workflows by several authors. In considering a declarative process as a model of some system it is natural to consider how the process behaves with respect to stakeholders. We derive a measure for stakeholder utility that can be applied in a very general setting. This derivation is achieved by listing a collection a properties which we argue such a stakeholder utility function ought to satisfy, and then using these to show a very specific form must hold for such a utility. The utility measure depends on the set of unique traces of the declarative process, and calculating this set requires a combinatorial analysis of the declarative graph that represents the process.

This builds on previous work of the author [2] wherein the combinatorial diversity metrics for declarative processes were derived for use in policy process analysis. The collection of stakeholder utilities can themselves then be used to form a metric with which we can compare different declarative processes to one another. These are illustrated using several examples of declarative processes that already exist in the literature.

1. Introduction

We present a method for calculating and analyzing stakeholder utilities of processes that arise in, but are not limited to, the social sciences. These areas include business process analysis [3], healthcare workflow analysis [12], [11], [16] and policy process analysis [10], [1], [17], [18]. This method is quite general and applicable to any situation in which declarative-type constraints play a part. A declarative process D is a pair consisting of a set of activities, and a list of constraints detailing how these activities may happen in relation to one another. For such a process, anything may happen so long as it is not explicitly forbidden by the constraint set.

Declarative processes have been used as models of business and healthcare workflows by several authors [3], [4], [6]. The notion of a declarative process is an attractive one: simply declare the constraints on activities in a system and then let the system run or evolve according to these constraints. An execution of such a system is a (potentially infinite) listing of the activities in the order they occur, i.e. they satisfy all of the constraints that define the system. Such listings are called *traces*. Two immediate concerns arise: Is there a sensible way to quantify stakeholder satisfaction for such processes? Is there a sensible way to compare two processes with regard to stakeholder satisfaction? In this paper we will take a first look at answers to these questions. The work we present in this paper is new in that it does not necessarily build on any existing body of work in the literature. The closest work by other authors to what we are examining seems to be the topic of similarity measures for business process models [13], [14], [15], however none of the material in those papers is necessarily applicable to the modelling framework that we are considering.

In considering a declarative process as a model of some system it is natural to consider how the process behaves with respect to stakeholders. Our previous paper [2] introduced several metrics related to the combinatorial diversity of a declarative process for use in policy process analysis. The purpose of that paper was to derive a metric that satisfied various properties and represented, to an extent, how 'free' a given declarative process was to happen. In this paper we have a different aim in mind. We will consider stakeholders in the declarative process and utilities for these stakeholders, and derive a measure for stakeholder utility. This is done by focusing on a class of representatives for a declarative process (the set of unique traces) and determining the solution to a collection of properties which we argue such a stakeholder utility function should satisfy. Calculating the set of unique traces requires a combinatorial analysis of the declarative graph that represents the declarative process. We then use these stakeholder utilities in order to give a method for comparing declarative processes. These are illustrated using several examples of declarative processes that already exist in the literature.

In Section 2 we introduce a declarative process and define the set of unique traces for a declarative process. In Section 3 we will present an algorithm for calculating the set of unique traces and discuss how this can be optimized. In Section 4 we consider stakeholders in a declarative process and suppose that each stakeholder will have a preference for or against each of the unique traces of the declarative process. We state and explain some reasonable properties that a utility function for a stakeholder should satisfy, and solving the equations that these properties imply gives an expression for the stakeholder utility function. We calculate the stakeholder utility vector, the vector of all utility functions for a given process, for several examples and discuss the results. In Section 5 we use the stakeholder utility vector to give a method for comparing different declarative processes with respect to a prescribed set of stakeholder preferences. This method uses

School of Mathematics and Statistics, University College Dublin, Dublin 4, Ireland.

ℓ_2-norm minimization but, as we show, is robust to 'noise' in the system. We illustrate this method by comparing three declarative processes in the paper. In Section 6 we conclude with a discussion of what we have shown.

2. Declarative Processes and Unique Traces

Let us first introduce some standard notation and terminology related to declarative processes [3], [2]. Let Σ be a set of *activities* and let Σ^* be the set of all possible sequences that one can form whose entries are element of Σ. That is

$$\Sigma^* := \{\epsilon\} \cup \{(e_1, e_2, e_3, \ldots) \ : \ e_i \in \Sigma\},$$

where ϵ denotes the empty sequence. A *trace* is a sequence of activities $\sigma = (e_1, \ldots, e_n) \in \Sigma^*$. An *event* is an occurrence of an activity in a trace.

A declarative constraint is a constraint on the activities in a process. We require a language through which to express temporal and modal aspects of these activities, and the natural choice for this is linear temporal logic [9]. Linear temporal logic (LTL) is an extension of propositional logic $\mathcal{L}_P$ that includes temporal modal operators $\mathbf{X}$ or $\circ$ (neXt), $\mathbf{U}$ (Until), $\mathbf{F}$ or $\diamond$ (Finally), $\mathbf{G}$ or $\Box$ (Globally), $\mathbf{R}$ (Release), $\mathbf{W}$ (Weak until) and $\mathbf{M}$ (strong release). As an example, given two activities a and b in Σ, we may wish to specify that event b must happen as a response to event a. In LTL one would represent this by the LTL formula $\mathsf{G}(a \Rightarrow \mathsf{F}b)$, which can be read as "it is globally true that (a occurs implies b occurs at some point after a)". However, the semantics of the Declare framework [7], [6] are easier to grasp in this respect and uses $\mathsf{resp}(a, b)$ for $\mathsf{G}(a \Rightarrow \mathsf{F}b)$. A list of some popular Declare expressions along with their LTL equivalents is given in Figure 2. For readability, in this paper we will consider the constraints as expressed in Declare.

We say that a trace σ satisfies the constraint $\mathsf{resp}(a, b)$ if any occurrence of a in the trace will feature an occurrence of b to its right. To represent this we write $\sigma \models \mathsf{resp}(a, b)$. It may be the case that a and b are not events in σ, in which case σ certainly satisfies the constraint $\mathsf{resp}(a, b)$. If Const is a set of constraints, then we will write $\sigma \models \mathsf{Const}$ if $\sigma \models x$ for all $x \in \mathsf{Const}$.

Let us consider the trace $\sigma = (3, 3, 2, 4, 1, 4)$ with $\Sigma = \{1, 2, 3, 4, 5\}$. The trace σ satisfies the declarative constraint $\mathsf{resp}(2, 1)$, i.e. $\sigma \models \mathsf{resp}(2, 1)$ since event 1 happens after event 2 in σ. However, both $\sigma \models \mathsf{resp}(2, 3)$ and $\sigma \models \mathsf{resp}(2, 5)$ are false.

Definition 2.1. A *declarative process* is a process on a set of activities Σ that satisfies all conditions in a set Const of declarative constraints. We will represent this as a pair $D = (\Sigma, \mathsf{Const})$. The set of traces of the process is

$$\mathsf{Traces}(D) \ = \ \{\sigma \in \Sigma^* \ : \ \sigma \models \mathsf{Const}\}.$$

Restrictions on the beginning and ending of these processes may be incorporated into the constraint set using declarative constraints.

Example 2.2. Consider the declarative process $D = (\Sigma, \mathsf{Const})$ where $\Sigma = \{1, 2, 3, 4, 5\}$ and $\mathsf{Const} = \{\mathsf{resp}(1, 2), \mathsf{prec}(2, 3), \mathsf{prec}(3, 5), \mathsf{succ}(1, 4), \mathsf{notsucc}(4, 2)\}$. Examples of traces for this process include ϵ, (2), $(1, 2, 3, 5, 4)$, $(2, 2)$, $(2, 2, \ldots, 2, 3)$, and $(2, 2, \ldots)$. There are will be an infinite number of traces, so $|\mathsf{Traces}(D)| = \infty$.

How should one approach analysing declarative processes? In theory one could derive a measure from simply looking at the two constraint listings that define them. A difficulty with this approach is that there are many different types of relations that can link two activities. The interactions between these constraints are consequently far more involved than, say, those represented by directed edges in a graph and studying the resulting graph properties.

An alternative way to consider and analyse such systems is to study the set of traces, $\mathsf{Traces}(D)$, of the declarative process. A drawback to this is that such a set can be infinite as in Example 2.2. We might then consider finite versions of the process that contain only traces of finite length [8], however the drawback in this case is more serious in that in the application areas we envisage, the occurrence of an activity (at some time perhaps far in the future) is more critical to our analysis than, say, a million occurrences of two activities up to the point of trace truncation.

This consideration leads us to considering traces in which an activity of Σ occurs at most once in a trace. In our first paper [2] on this subject we were able to justify this consideration as 'first passage/time traces'. No choice of projection from a set of infinite objects to a set of finite objects comes without a drawback. However we feel that the considerations of the application area combined with the notion of first passage times make this the best set of representatives for our consideration. We note that this could be specified at the constraint level by including into the set of constraints a declarative constraint on every activity that it can not happen more than once. An equivalent way to consider this is to simply look at the subset of traces that contain at most once occurrence of every event.

Definition 2.3. Let $D = (\Sigma, \mathsf{Const})$ be a declarative process. Let $\mathsf{UniqueTraces}(D)$ be the set of those traces $\sigma \in \mathsf{Traces}(D)$ for which every activity in σ is unique.

Example 2.4. Consider the declarative process $D = (\Sigma, \mathsf{Const})$ given in Example 2.2 where $\Sigma = \{1, 2, 3, 4, 5\}$ and $\mathsf{Const} = \{\mathsf{resp}(1, 2), \mathsf{prec}(2, 3), \mathsf{prec}(3, 5), \mathsf{succ}(1, 4), \mathsf{notsucc}(4, 2)\}$. The set $|\mathsf{Traces}(D)| = \infty$ while

$$\begin{aligned}\mathsf{UniqueTraces}(D) = \{&\epsilon, (2), (2, 3), (1, 2, 4), (2, 3, 5),\\ &(1, 2, 3, 4), (1, 2, 4, 3), (1, 2, 3, 4, 5),\\ &(1, 2, 3, 5, 4), (1, 2, 4, 3, 5)\}.\end{aligned}$$

Example 2.5 (After Dinner)**.** The following is a description of the house rules for a child between the end of dinner and going to bed. After dinner is finished (and it must be finished) the table must be tidied. If they want to do a jigsaw that the table must have been tidied beforehand. The doing of a jigsaw means this jigsaw must be tidied away afterwards. The child can watch a bedtime television show only after finishing dinner. The child cannot get ready for bed before the jigsaw

Declare Constraint	Explanation	LTL expression
$\mathsf{participation}(a)$	Event a occurs at least once	$\mathsf{F}a$
$\mathsf{initial}(a)$	Event a is first to occur	a
$\mathsf{resp}(a,b)$	If event a occurs, then event b occurs after a	$\mathsf{G}(a \Rightarrow \mathsf{F}b)$
$\mathsf{chainresp}(a,b)$	If event a occurs, then event b occurs immediately after a	$\mathsf{G}(a \Rightarrow \mathsf{N}b)$
$\mathsf{prec}(a,b)$	Event b occurs only if preceded by event a	$(\neg b)\mathbf{W}a$
$\mathsf{succ}(a,b)$	Event a occurs iff it is followed by event b	$\mathsf{G}(a \Rightarrow \mathsf{F}b) \ \wedge ((\neg b)\mathbf{W}a)$
$\mathsf{not-coexist}(a,b)$	Events a and b cannot coexist b	$\neg(\mathsf{F}a \Leftrightarrow \mathsf{F}b)$

Figure 1. Some typical Declare constraints

has been tidied away (in the event it needs to be). The child cannot get ready for bed before watching the bedtime show. The child cannot get ready for bed before tidying the table.

To model this as a declarative process, label the events as follows:

1. Finish dinner
2. Tidy table
3. Do jigsaw
4. Tidy away jigsaw
5. Watch the bedtime show
6. Get ready for bed

The above description translates into the following constraint set (see Figure 2 for an illustration of the constraints):

$$\begin{aligned}\mathsf{Const}_{AD1} = \{&\mathsf{participation}(1), \mathsf{resp}(1,2), \mathsf{prec}(1,5), \mathsf{prec}(2,3),\\ &\mathsf{succ}(3,4), \mathsf{notsucc}(6,4), \mathsf{notsucc}(6,5),\\ &\mathsf{notsucc}(6,2)\}.\end{aligned}$$

and to the declarative process $D_{AD1} = (\{1,2,3,4,5,6\}, \mathsf{Const}_{AD1})$. As a declarative process, it is easy to see that $\mathsf{Traces}(D_{AD1})$ will have an infinite size. As a process, it is clear from the description that each of the activities is intended to happen at most once. To analyse this declarative process, we are interested in $\mathsf{UniqueTraces}(D_{AD1})$, which is

$$\begin{aligned}&\mathsf{UniqueTraces}(D_{AD1}) =\\ &\{(1,2),(1,2,5),(1,5,2),(1,2,6),(1,2,3,4),(1,2,5,6),\\ &(1,5,2,6),(1,2,3,4,5),(1,2,3,5,4),(1,2,5,3,4),\\ &(1,5,2,3,4),(1,2,3,4,6),(1,2,3,4,5,6),(1,2,3,5,4,6),\\ &(1,2,5,3,4,6),(1,5,2,3,4,6)\}\end{aligned}$$

This set reveals that, through the rules the parents laid down, the child does not necessarily have to ever get ready for bed (as evidenced by eight traces that do not contain activity 6), and satisfies all the rules they must follow. If one now includes the rule that activity 6 must happen, i.e.

$$\mathsf{Const}_{AD2} = \mathsf{Const}_{AD1} \cup \{\mathsf{participation}(6)\}$$

and

$$D_{AD2} = (\{1,2,3,4,5,6\}, \mathsf{Const}_{AD2}),$$

then one finds those traces that the parents intended in the first place:

$$\begin{aligned}&\mathsf{UniqueTraces}(D_{AD2}) =\\ &\{(1,2,6),(1,2,5,6),(1,5,2,6),(1,2,3,4,6),\\ &(1,2,3,4,5,6),(1,2,3,5,4,6),(1,2,5,3,4,6),\\ &(1,5,2,3,4,6)\}.\end{aligned}$$

3. Enumerating and generating the unique trace representatives

In order to generate the set of unique traces of a declarative process, we have to consider all events that can occur in such a trace. This can be any subset X of the set of activities Σ. We must then consider all the different permutations of events in X to see if such a sequence satisfies the set of constraints. Algorithm 1 gives a simple procedure for doing this.

Algorithm 1 Generating the set of unique traces $\mathsf{UniqueTraces}(D)$

```
1: procedure UniqueTraces(Σ, Const)
2:     A ← ∅
3:     for X ⊆ Σ do
4:         for π ∈ Permutations(X) do
5:             if π ⊨ Const then
6:                 A ← A ∪ {π}
7:     return A
```

The above algorithm will of course be dependent upon the size of Σ and the number of of times 'satisfies' is called will be $2.718|\Sigma|!$ For example, if Σ has 7 activities then $2.718|\Sigma|! = 2.718 \times 7 \times 6 \times \cdots \times 1 = 13700$. This fact will lead to an exponential slowdown for every extra activity that is considered in Σ. Consequently, the runtime on an average PC will increase from minutes to double digit-hours as $|\Sigma|$ goes from 9 to 15.

We can remove needless checking by stripping a declarative process down to a smaller smaller core process. This is done by removing the equivalent of *leaves* from the constraint list (and by extension the activity set). For example, if we have a declarative process $D = (\Sigma, \mathsf{Const})$ and the only appearance of activity j, say, in Const is as $\mathsf{resp}(i,j)$, then we can construct $\mathsf{UniqueTraces}(D)$ from $\mathsf{UniqueTraces}(D')$ where $D' = (\Sigma\backslash\{j\}, \mathsf{Const}\backslash\{\mathsf{resp}(i,j)\})$ by using a simple resp leaf addition procedure as given in Algorithm 2.

This procedure is useful in that it allows us to consider decompose stakeholder satisfaction for the declarative process on a smaller process. If activity j features in $G_{i'}$ for some stakeholder $S_{i'}$ then it will be possible to re-specify $G_{i'}$ on the smaller process while conditioning on the position/absence-of activity i in any traces. It also allows us to gain some insights into the distribution of activities within the set of unique traces. This information is useful in tracking the evolution of the unique traces and could be utilized in later work to provide bounds on certain aspects of the declarative process.

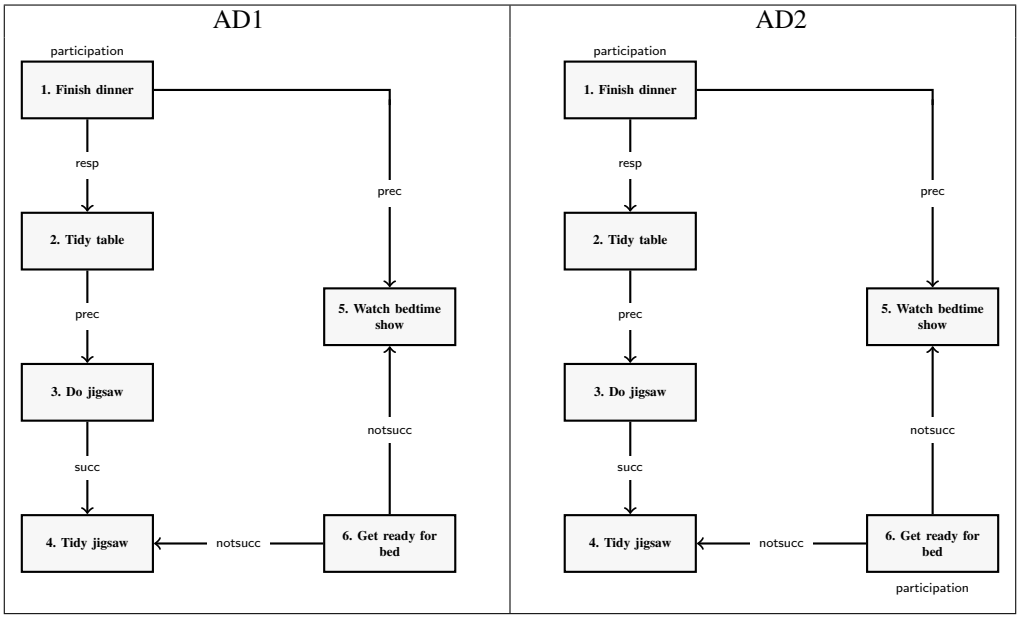

Figure 2.

Algorithm 2 Generating $\mathsf{UniqueTraces}(D)$ from $\mathsf{UniqueTraces}(D')$ where $j \notin \Sigma'$ and D contains the additional constraint $\mathsf{resp}(i,j)$

procedure TRLEAF($\mathsf{UniqueTraces}(\Sigma', \mathsf{Const}'), \mathsf{resp}(i,j)$)
  $A \leftarrow \emptyset$
  for $\sigma \in \mathsf{UniqueTraces}(D')$ **do**
    if $i \in \sigma$ **then**
      for $k \in \{\mathrm{index}(\sigma, i), \ldots, \mathrm{length}(\sigma)\}$ **do**
        $\mu \leftarrow \sigma$ with j inserted after the kth entry
        if $\mu \models \mathsf{Const}'$ **then**
          $A \leftarrow A \cup \{\mu\}$
    else
      $A \leftarrow A \cup \{\sigma\}$
      for $k \in \{\mathrm{index}(\sigma, i), \ldots, \mathrm{length}(\sigma)\}$ **do**
        $\mu \leftarrow \sigma$ with j inserted after the kth entry
        if $\mu \models \mathsf{Const}'$ **then**
          $A \leftarrow A \cup \{\mu\}$
  return A

Algorithms 3 and 4 deal with the declarative constraints prec and succ, respectively. Similar algorithms can be given for other declarative constrains and these allow for the calculation of the unique traces of declarative processes of significantly larger size. We have calculated unique traces for declarative processes on 23 activities using these algorithms but have not yet had cause to consider larger systems.

4. Measuring stake-holder utilities

To every declarative process $D = (\Sigma, \mathsf{Const})$ we may consider a set of stake-holders $S = S(D) := \{S_1, \ldots, S_m\}$. These stake-holders have an interest in aspects of the declarative process such as the execution order or existence of particular activities. As such there are some traces that are more desirable than others for these stakeholders.

Algorithm 3 Generating $\mathsf{UniqueTraces}(D)$ from $\mathsf{UniqueTraces}(D')$ where $j \notin \Sigma'$ and D contains the additional constraint $\mathsf{prec}(i,j)$

procedure TPLEAF($\mathsf{UniqueTraces}(\Sigma', \mathsf{Const}'), \mathsf{prec}(i,j)$)
  $A \leftarrow \emptyset$
  for $\sigma \in \mathsf{UniqueTraces}(D')$ **do**
    $A \leftarrow A \cup \{\sigma\}$
    if $i \in \sigma$ **then**
      for $k \in \{\mathrm{index}(\sigma, i), \ldots, \mathrm{length}(\sigma)\}$ **do**
        $\mu \leftarrow \sigma$ with j inserted after the kth entry
        if $\mu \models \mathsf{Const}'$ **then**
          $A \leftarrow A \cup \{\mu\}$
  return A

Algorithm 4 Generating $\mathsf{UniqueTraces}(D)$ from $\mathsf{UniqueTraces}(D')$ where $j \notin \Sigma'$ and D contains the additional constraint $\mathsf{succ}(i,j)$

procedure TSLEAF($\mathsf{UniqueTraces}(\Sigma', \mathsf{Const}'), \mathsf{prec}(i,j)$)
  $A \leftarrow \emptyset$
  for $\sigma \in \mathsf{UniqueTraces}(D')$ **do**
    if $i \in \sigma$ **then**
      for $k \in \{\mathrm{index}(\sigma, i), \ldots, \mathrm{length}(\sigma)\}$ **do**
        $\mu \leftarrow \sigma$ with j inserted after the kth entry
        if $\mu \models \mathsf{Const}'$ **then**
          $A \leftarrow A \cup \{\mu\}$
    else
      $A \leftarrow A \cup \{\sigma\}$
  return A

For example, one stake-holder might prefer an execution of the declarative process wherein a particular activity happens (be it at the end of the process or at any stage of the process). Another preference might be that an activity a happens iff activity b happens afterwards. It might be the case that a stake-holder is happy with any one of a collection of activities happening, or is insistent that a particular collection of activities must all happen (at least at some point).

In stating preferences for stake-holders, we are not considering these preferences to have any dynamic implications on how the process unfolds. In this sense it is best to assume stake-holders preferences are private during the execution of such a process. Our measure will compare the number of outcomes of such a process to the number of outcomes that are preferable to a given stakeholder. We are therefore not trying to determine the best outcomes of such a process, but instead consider how well a process behaves in relation to stake-holder preferences.

A stake-holder preference will be represented by an LTL expression. We will denote the LTL expression that corresponds to a 'desirable/good' outcome for stakeholder S_i by G_i and define $G = (G_1, \ldots, G_m)$. The declarative system along with the stake-holders and their (private) preferences is represented by a triple $T = (D, S, G)$ that we will call a *declarative stakeholder system*. A more involved analysis might involve a partial ordering of preferred outcomes with scores assigned to each. In this paper we will restrict ourselves to a binary measure of whether a given trace is 'good' for a particular stake-holder. For stake-holder S_i, let us define

$$\mathsf{GoodTraces}_i(D) = \{\sigma \in \mathsf{UniqueTraces}(D) \ : \ \sigma \models G_i\}.$$

This is the set of 'good' outcomes of the declarative process for S_i.

We wish to associate a utility u_i to stake-holder S_i that represents their satisfaction with declarative stakeholder system $T = (D, S, G)$. We make the following reasoned assumptions on $u_i(T)$.

Assumption 1 We are not dealing with a degenerate case so $\mathsf{UniqueTraces}(D)$ is non-empty.

Assumption 2 The utility $u_i(T)$ will be a function of both the size of $\mathsf{GoodTraces}_i(D)$ and $\mathsf{UniqueTraces}(D)$.

Assumption 3 The utility should achieve its maximum value 1 when $\mathsf{GoodTraces}_i(D) = \mathsf{UniqueTraces}(D)$ and achieve its minimum value 0 when $\mathsf{GoodTraces}_i(D) = \emptyset$.

Assumption 4 The utility should be an increasing function of $|\mathsf{GoodTraces}_i(D)|$.

Assumption 5 The utility should possess a scaling property so that a doubling of $|\mathsf{UniqueTraces}|$ does not mean that a doubling of $|\mathsf{GoodTraces}_i(D)|$ is required to achieve the same utility.

Theorem 4.1. *Let $D = (\Sigma, \mathsf{Const})$ be a declarative process. Let $S = \{S_1, \ldots, S_n\}$ be a set of stake-holders and let $G = \{G_1, \ldots, G_n\}$ be the set of preferences for the stake-holders. Suppose that Assumptions 1–5 hold true. Then the stakeholder utility vector of the declarative stakeholder system $T = (D, S, G)$ is $u(T) = (u_1(T), \ldots, u_n(T))$ where*

$$u_i(T) = \frac{\ln(1 + |\mathsf{GoodTraces}_i(D)|)}{\ln(1 + |\mathsf{UniqueTraces}(D)|)}.$$

Proof. Let us write $a = |\mathsf{GoodTraces}_i(D)|$ and $b = |\mathsf{UniqueTraces}(D)|$ so that, by Assumptions 2 and 3, $u_i(T) = f(a, b)$ with $f(0, b) = 0$ and $f(b, b) = 1$. Note that by Assumption 1 we have $b > 0$. Assumption 4 tells us the utility should increase with the size of $\mathsf{GoodTraces}_i(D)$, so the function $f(x, b)$ should be an increasing function of x.

However, if one sets $f(x, b)$ to simply be a function $g(x/b)$ that is increasing, then we rule out being able to incorporate important aspects with relation to how such a utility should behave with respect to different scalings. In order to accommodate Assumption 5 let us assume $f(x, b) = g(x)/g(b)$ for some function $g(x)$. With regard to the properties of f outlined above, these imply that $g(x)$ must satisfy the following:

$$g(0) = 0 \text{ and } g(x) \text{ is an increasing function of } x.$$

The function g is not a direct measure of utility, but represents the weight attached to the number of desirable traces for stake-holder S_i. Consider instances of $\mathsf{GoodTraces}_i(D)$ that have 0, 1, 2, and 100, valid traces (this is similar to what was considered in [2]). An empty $\mathsf{GoodTraces}_i(D)$ indicates no desirable executions of the process for user S_i. If $\mathsf{GoodTraces}_i(D)$ consists of a single trace then it is better (for S_i) than the previous case of no traces. If $\mathsf{GoodTraces}_i(D)$ consists of 2 traces then it is certainly better (for S_i) than a process that only has one trace. However, we would consider a set of preferred traces having 101 traces to be better, but only marginally, to a process that has 100 traces.

The simplest function that represents this situation is one that is inversely proportional to its argument, i.e. satisfies the differential equation $g'(x) = k_1/(x + k_2)$. In order for the general solution to this, $g(x) = k_1 \ln(x + k_2) + c$ for constants k, c, to represent our situation we must have $k_1 > 0$. If there is no trace in the set of desirable outcomes, then we will have $g(0) = k_1 \ln(k_2) + c$. In order for this to equal 0, we must have $k_2 = 1$ and $c = 0$ and this implies $g(x) = k_1 \ln(x + 1)$.

This now gives us the required expression for the utility function for stake-holder S_i:

$$u_i(T) = \frac{g(a)}{g(b)} = \frac{\ln(a+1)}{\ln(b+1)} = \frac{\ln(1 + |\mathsf{GoodTraces}_i(D)|)}{\ln(1 + |\mathsf{UniqueTraces}(D)|)}. \tag{1}$$

□

Example 4.2 (After Dinner cont'd)**.** Consider the two declarative processes D_{AD1} and D_{AD2} from Example 2.5. Suppose that the two stake-holders are S_1, the child, and S_2, the parents and $S = \{S_1, S_2\}$. We will consider some different forms for G_1 and G_2 and calculate their utility in order to see how the declarative processes compare for the stake-holders. Note that $|\mathsf{UniqueTraces}(D_{AD1})| = 16$ and $|\mathsf{UniqueTraces}(D_{AD2})| = 8$.

Before doing this, let us reiterate a point made at the beginning of this section. In this study, once a preference for one stakeholder is stated, it is natural to assume that that stake-holder will engage in some activities to force a preferential outcome. This certainly might be the case and, in the case of two stake-holders having somewhat complementary preferences, it might be considered as a competition. Our

purpose is not to study how effective such stake-holders are in forcing the outcome of a process. (A stake-holder might not be involved in the execution of a process or have any impact on the activities of that process.) Instead, our goal is to analyse how 'good' a declarative process is in relation to stated stake-holder preferences.

(i) On a given evening, the child has a desire to watch the bedtime show after dinner. The parents are interested in the child getting ready for bed. To assign LTL expressions to these events, we have

$$G_1 = \mathsf{participation}(5) \text{ and } G_2 = \mathsf{participation}(6).$$

Given these expressions, we find that there are 12 traces in $\mathsf{UniqueTraces}(D_{AD1})$ that contain activity 5, so $|\mathsf{GoodTraces}_1(D_{AD1})| = 12$. There are 8 activities in $\mathsf{UniqueTraces}(D_{AD1})$ that contain activity 6, so $|\mathsf{GoodTraces}_2(D_{AD1})| = 6$. These values allow us to calculate utilities for the declarative process $AD1$:

$$u_1(D_{AD1}, S, G) = \frac{1+\ln(12)}{1+\ln(16)} = 0.92374$$

and

$$u_2(D_{AD1}, S, G) = \frac{1+\ln(6)}{1+\ln(16)} = 0.74001.$$

Likewise, for the second declarative process D_{AD2} we find that $|\mathsf{GoodTraces}_1(D_{AD2})| = 6$ and $|\mathsf{GoodTraces}_2(D_{AD1})| = 8$, from which we calculate the utilities:

$$u_1(D_{AD2}, S, G) = \frac{1+\ln(6)}{1+\ln(8)} = 0.90658$$

and

$$u_2(D_{AD2}, S, G) = \frac{1+\ln(8)}{1+\ln(8)} = 1.$$

The first stake-holders utility is better with process AD1 whereas the opposite is true for the second stake-holder.

(ii) On a given evening, the child has a particular wish to do their jigsaw and then watch the bedtime show before tidying the jigsaw. The parents, for some reason or another, would rather that the child not watch television after dinner and before going to bed. The LTL expressions corresponding to these propositions are

$$\begin{aligned} G'_1 =& \mathsf{participation}(3) \wedge \mathsf{participation}(5) \wedge \mathsf{succ}(3,5) \\ & \wedge \mathsf{succ}(5,4) \\ G'_2 =& \neg \mathsf{participation}(5). \end{aligned}$$

There are 2 traces in $\mathsf{UniqueTraces}(D_{AD1})$ that satisfy G'_1 and 4 traces in $\mathsf{UniqueTraces}(D_{AD1})$ that satisfy G'_2. There is a single trace in $\mathsf{UniqueTraces}(D_{AD2})$ that satisfies G'_1 and 2 traces in $\mathsf{UniqueTraces}(D_{AD2})$ that satisfy G'_2. The utilities for the two different stake-holders for both declarative processes are now:

$$\begin{aligned} u_1(D_{AD1}, S, G') &= \frac{1+\ln(2)}{1+\ln(16)} = 0.44880 \\ u_2(D_{AD1}, S, G') &= \frac{1+\ln(4)}{1+\ln(16)} = 0.63253 \\ u_1(D_{AD2}, S, G') &= \frac{1+\ln(1)}{1+\ln(16)} = 0.26507 \\ u_2(D_{AD2}, S, G') &= \frac{1+\ln(2)}{1+\ln(16)} = 0.44880. \end{aligned}$$

For this choice of G', we see that both users would therefore have a preference for process AD1 over AD2.

Discussion: In this example we have considered the two declarative processes D_{AD1} and D_{AD2}. We have seen how we can compare these two processes with respect to two different preferences for the two different stake-holders. There is no reason to limit the number of stake-holders to two. Our method shows that in one of these cases, the utilities calculated indicate a clear preference for one declarative process over another. For the other set of preferences, that is not the case.

Example 4.3 (Patient Handler 1)**.** Let us consider an example of a declarative process from [6] that is illustrated in Figure 3. This is a process for handling a patient at the first aid department in a hospital with a suspected arm fracture and comprises eight activities. The patient is initially examined by a medical professional (activity 1) and the init constraint on this activity means it is the first activity that occurs in any execution of this process. Activity 5 'medication' shares no constraints with any other processes, however the init constraint on activity 1 forbids activity 5 from occurring first. Two constraints warrant further explanation:

- The 1of4 constraint indicates that at least one of the four constraints 3,4,6,8 must happen. This constraint is not conditional on other constraints or activities, and so rules out the situation whereby activity 1 happens (patient examined) followed by them being given medication (activity 5) for what is a sore arm. We are not sure why this possibility was ruled out in the original model but will not alter it since it adds to the diversity of the underlying process. Furthermore, since we are dealing with representatives of traces, one could argue that this is represented by the trace $(1, 5, 8)$ whereby the patient simply chooses not to wear a sling.
- The 'optional response' from activity 3 to activity 7 is different to the normal response in that if activity 3 happens then 7 may or may not occur afterwards. However if activity 3 does not occur then activity 7 certainly cannot occur. Of course neither 3 nor 7 need occur and this constraint is still satisfied.

The unique traces of the declarative process D_{PH1} are summarized in Figure 6. Let us now consider some stakeholders in this process. Recall that stakeholders do not have to have an active role in a process but may have some clear preferences regarding observed executions of the process.

Stakeholder S_1 The first stakeholder is the patient who is presenting to the emergency department. This patient has

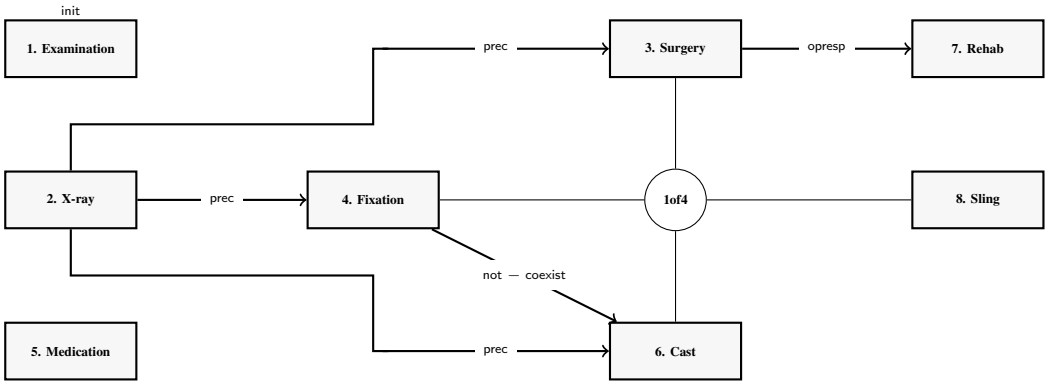

Figure 3. Illustration of the declarative process *Patient Handler* (that we will refer to as PH1) from [6]

a phobia of surgery and is averse to medication. The LTL proposition that models the stakeholder's preferences is

$$G_1 = \neg\mathsf{participation}(3) \ \wedge \ \neg\mathsf{participation}(5).$$

Stakeholder S_2 The second stakeholder is the X-ray department who are overwhelmed by the number of X-rays that are needed. The LTL proposition that models the stakeholder's preferences is

$$G_2 = \neg\mathsf{participation}(2).$$

Stakeholder S_3 The third stakeholder is the surgery department who have a very small team and do not have the resources to dedicate to removing casts before surgery. The LTL proposition that models the stakeholder's preferences is

$$\begin{aligned} G_3 =& \neg(\mathsf{participation}(3) \wedge \mathsf{participation}(6)) \\ & \vee \ (\mathsf{participation}(3) \wedge \mathsf{participation}(6) \wedge \mathsf{succ}(3,6)). \end{aligned}$$

Stakeholder S_4 The fourth stakeholder is the hospital itself. Resources are often scare and a patient who needs multiple resources can be costly (both in work and time) for the hospital. An avoidance of the overuse of multiple costly resources is preferred. An instance of this is a patient who is initially given a sling, their arm does not improve, and so an X-ray indicates a fixation is the best option. This fixation does not solve the problem and surgery is required, followed by rehabilitation. The hospital has noticed that in the past there have been several such 'expensive' instances among patients who have revisited multiple times. This process execution is represented by the unique trace $\sigma = (1, 8, 2, 4, 3, 7)$. The LTL proposition that models the stakeholder's preferences is

$$\begin{aligned} G_4 =& \neg(\mathsf{participation}(1) \wedge \mathsf{succ}(1,8) \wedge \mathsf{succ}(8,2) \\ & \wedge \mathsf{succ}(2,4) \wedge \mathsf{succ}(4,3) \wedge \mathsf{succ}(3,7)). \end{aligned}$$

Stakeholder S_5 The fifth stakeholder is the pharmaceutical industry. In this instance it benefits when patients are prescribed medication or their medications are used during surgery. The LTL proposition that models the stakeholders preferences for this process is

$$G_5 = \mathsf{participation}(3) \vee \mathsf{participation}(5).$$

Examining the traces in $\mathsf{UniqueTraces}(D_{PH1})$ with respect to the five stakeholders we find

$$\begin{aligned} &(\mathsf{GoodTraces}_1(D_{PH1}), \ldots, \mathsf{GoodTraces}_5(D_{PH1})) \\ &\qquad = (11, 3, 389, 452, 448). \end{aligned}$$

This gives the following collection of utilities for the stakeholders in $T_{PH1} = (D_{PH1}, S, G)$:

$$\begin{aligned} u_1(T_{PH1}) &= \frac{1+\ln(11)}{1+\ln(459)} = 0.47663 \\ u_2(T_{PH1}) &= \frac{1+\ln(3)}{1+\ln(459)} = 0.29437 \\ u_3(T_{PH1}) &= \frac{1+\ln(389)}{1+\ln(459)} = 0.97679 \\ u_4(T_{PH1}) &= \frac{1+\ln(452)}{1+\ln(459)} = 0.99784 \\ u_5(T_{PH1}) &= \frac{1+\ln(448)}{1+\ln(459)} = 0.99660. \end{aligned}$$

This gives the stakeholder utility vector $u(T_{PH1}) = (0.47663, 0.29437, 0.97679, 0.99784, 0.99660)$ for the declarative stakeholder system $T_{PH1} = (D_{PH1}, S, G)$.

Example 4.4 (Patient Handler 2)**.** In this example we will consider a modified patient handler process that we call Patient Handler 2 and is motivated by an example given in Mertens et al. [4]. The process we look at is a simplified version of the one given in [4] since in that paper the authors introduced a more general declarative framework that captured aspects of a healthcare process that was not captured by the original declarative process given in [6]. The concerns of [4] are quite different to ours and our purpose in looking at that declarative process is to have a second process to compare the first process PH1 to.

With these points in mind we will describe two simplified versions of Patient Handler 2 that we will refer to as PH2a and PH2b. The difference between these two is that PH2b contains an additional activity (activity 11 in Figure 4) that does not feature in Patient Handler 1 but which we find interesting to include for comparison purposes.

In this modification of Patient Handler 1, we have attempted to preserve the labelling of similar activities. In Patient Handler 1 there was one activity (activity 5) for the patient being

given medication. Patient Handler 2 considers a collection of different medications, and those medications that correspond to the old activity 5 have been labelled 51, 52, and 53 in order to provide a comparison between the two processes. Moreover, there are now activities for prescribing anti-inflammatory and anti-coagulation drugs after surgery. These did not feature in PH1. Information about the unique traces of the two processes D_{PH2a} and D_{PH2b} can be found in Figures 7 and 8, respectively.

Let us consider stakeholders in these processes precisely as with did in PH1. As activity 11 (physiotherapy) does not impact on any of the preferences for the stakeholders S_1, ..., S_6, we will assume that the expressions for stakeholder preferences for PH2a and PH2b are the same.

Stakeholder S_1 The patient who has a phobia of surgery and is averse to medication. The LTL proposition that models the stakeholder's preferences is now

$$\begin{aligned} G_1 = \neg(&\mathsf{participation}(3) \vee \mathsf{participation}(51) \\ &\vee \mathsf{participation}(52) \vee \mathsf{participation}(53) \\ &\vee \mathsf{participation}(9) \vee \mathsf{participation}(10)). \end{aligned}$$

Stakeholder S_2 The overwhelmed X-ray department. The LTL proposition that models the stakeholder's preferences is the same as before

$$G_2 = \neg\mathsf{participation}(2).$$

Stakeholder S_3 The under-staffed surgery department. The LTL proposition that models the stakeholder's preferences is the same as before

$$\begin{aligned} G_3 = &\neg(\mathsf{participation}(3) \wedge \mathsf{participation}(6)) \\ &\vee (\mathsf{participation}(3) \wedge \mathsf{participation}(6) \wedge \mathsf{succ}(3,6)). \end{aligned}$$

Stakeholder S_4 The hospital that would like to cut down on a particularly common overuse of its resources. The LTL proposition that models the stakeholder's preferences is the same:

$$\begin{aligned} G_4 = \neg(&\mathsf{participation}(1) \wedge \mathsf{succ}(1,8) \wedge \mathsf{succ}(8,2) \\ &\wedge \mathsf{succ}(2,4) \wedge \mathsf{succ}(4,3) \wedge \mathsf{succ}(3,7)). \end{aligned}$$

Stakeholder S_5 The pharmaceutical industry that benefits when patients are prescribed or given medications. The LTL proposition that models the stakeholder's preferences is

$$\begin{aligned} G_5 = &\mathsf{participation}(3) \vee \mathsf{participation}(51) \\ &\vee \mathsf{participation}(52) \vee \mathsf{participation}(53) \\ &\vee \mathsf{participation}(9) \vee \mathsf{participation}(10). \end{aligned}$$

Examining the traces in $\mathsf{UniqueTraces}(D_{PH2a})$ and $\mathsf{UniqueTraces}(D_{PH2b})$ with respect to the five stakeholders we find

$$\begin{aligned} &(\mathsf{GoodTraces}_1(D_{PH2a}), \ldots, \mathsf{GoodTraces}_5(D_{PH2a})) \\ &=(324, 1457048, 16316590, 16285678, 16316266) \end{aligned}$$

and

$$\begin{aligned} &(\mathsf{GoodTraces}_1(D_{PH2b}), \ldots, \mathsf{GoodTraces}_5(D_{PH2b})) \\ &=(1952, 16316590, 199143708, 198749700, 199141756). \end{aligned}$$

The collection of utilities for the declarative stakeholder system $T_{PH2a} = (D_{PH2a}, S, G)$ is:

$$\begin{aligned} u_1(T_{PH2a}) &= \frac{1+\ln(324)}{1+\ln(16316590)} = 0.38510 \\ u_2(T_{PH2a}) &= \frac{1+\ln(1457048)}{1+\ln(16316590)} = 0.86280 \\ u_3(T_{PH2a}) &= \frac{1+\ln(16316590)}{1+\ln(16316590)} = 1.00000 \\ u_4(T_{PH2a}) &= \frac{1+\ln(16285678)}{1+\ln(16316590)} = 0.99989 \\ u_5(T_{PH2a}) &= \frac{1+\ln(16316266)}{1+\ln(16316590)} = 0.99999. \end{aligned}$$

This gives the stakeholder utility vector $u(T_{PH2a}) = (0.38510, 0.86280, 1.00000, 0.99989, 0.99999)$. The collection of utilities for the declarative stakeholder system $T_{PH2a} = (D_{PH2a}, S, G)$ is:

$$\begin{aligned} u_1(T_{PH2b}) &= \frac{1+\ln(1952)}{1+\ln(199143708)} = 0.42649 \\ u_2(T_{PH2b}) &= \frac{1+\ln(16316590)}{1+\ln(199143708)} = 0.87559 \\ u_3(T_{PH2b}) &= \frac{1+\ln(199143708)}{1+\ln(199143708)} = 1.00000 \\ u_4(T_{PH2b}) &= \frac{1+\ln(198749700)}{1+\ln(199143708)} = 0.99990 \\ u_5(T_{PH2b}) &= \frac{1+\ln(199141756)}{1+\ln(199143708)} = 0.99999 \end{aligned}$$

This gives stakeholder utility vector $u(T_{PH2b}) = (0.42649, 0.87559, 1.00000, 0.99990, 0.99999)$.

5. Comparing processes using stakeholder utility vectors

The stakeholder utility vector of a declarative stakeholder system is a vector of values between 0 and 1 in which the ith entry represents the utility to user i. With the notation we have been using, we have the

$$u(T) = (u_1(T), \ldots, u_m(T)).$$

The optimal outcome for all stakeholders would be for $u(T) = (1, 1, \ldots, 1)$. Given a collection of declarative processes $D_1, \ldots, D_t$, in order to determine which declarative process is 'optimal' with respect to all stakeholders, one can simply determine the stakeholder utility vector that is closest to $(1, 1, \ldots, 1)$ in Euclidean m-space. This is done by using the Euclidean norm, also known as the ℓ_2-norm, of a vector in m-space:

$$\|(v_1, \ldots, v_m)\|_2 \ := \ \sqrt{v_1^2 + \ldots + v_m^2}.$$

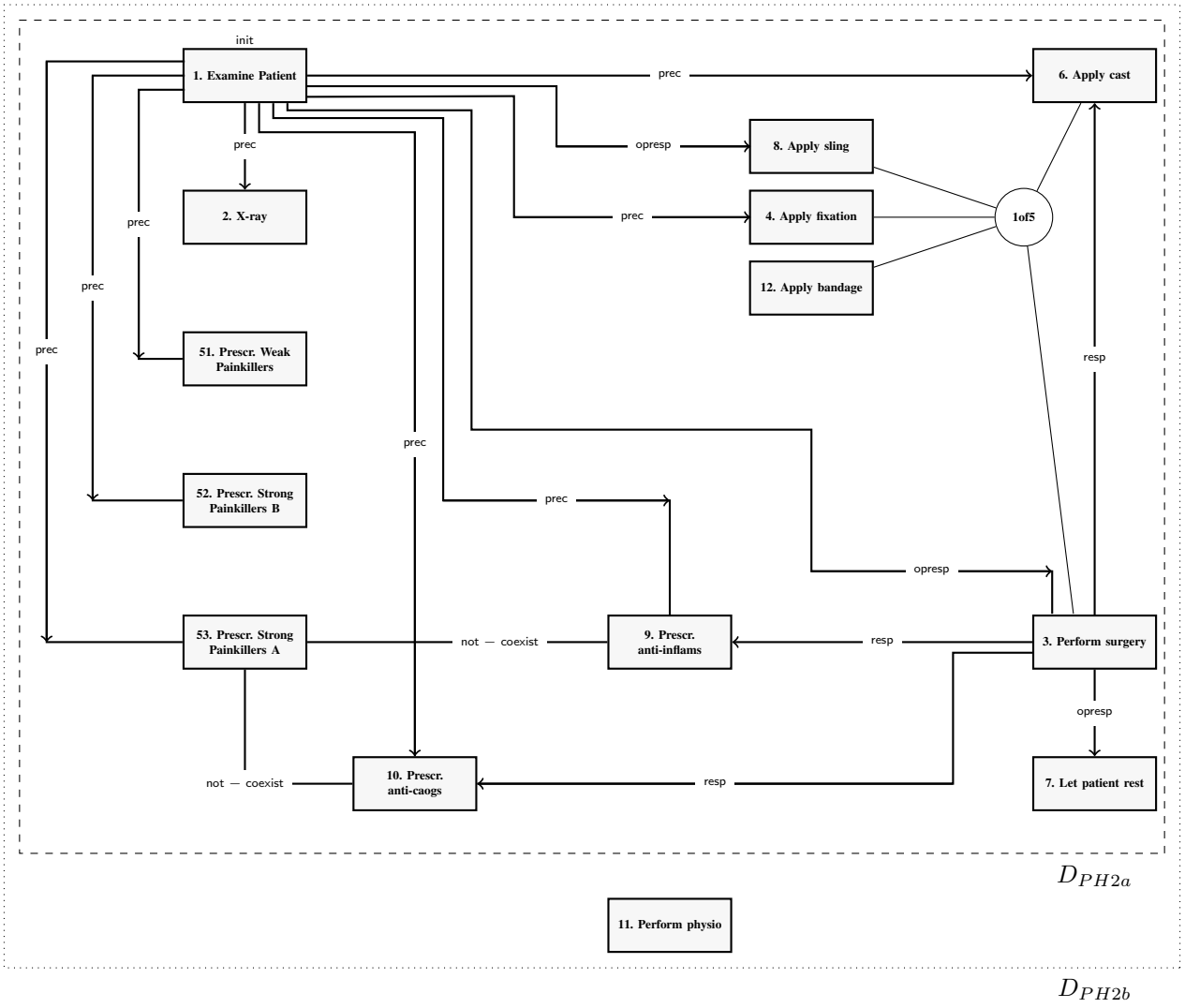

Figure 4. Both versions of Patient Handler 2 are illustrated in the diagram. The constraints for D_{PH2a} are contained within the dashed rectangle, and the constraints for D_{PH2b} are contained within the dotted rectangle.

Using this we determine which declarative stakeholder system $T_i = (D_i, S, G)$ minimizes

$$\min_{1 \le i \le n} \|u(T_i) - (1, 1, \ldots, 1)\|_2 .$$

Let us define $H(T_i) := \|u(T_i) - (1, 1, \ldots, 1)\|_2$.

Example 5.1. Consider Examples 4.3 and 4.4. We have

$$\begin{aligned} &H(T_{PH1}) = \\ &\|(0.4766, 0.2944, 0.9768, 0.9978, 0.9966) - (1,1,1,1,1)\|_2 \\ &\qquad = 0.87885 \\ &H(T_{PH2a}) = \\ &\|(0.3851, 0.8628, 1.0000, 0.9999, 0.9999) - (1,1,1,1,1)\|_2 \\ &\qquad = 0.63002 \\ &H(T_{PH2b}) = \\ &\|(0.4265, 0.8756, 1.0000, 0.9999, 0.9999) - (1,1,1,1,1)\|_2 \\ &\qquad = 0.58685. \end{aligned}$$

The minimum of these is $H(T_{PH2b})$ and so the declarative stakeholder system $PH2b$ is the optimal choice from the set $\{PH1, PH2a, PH2b\}$.

The method of ℓ_2-norm minimization is known to be sensitive to a moderate change in one of the values. In our setting this might amount to a single stakeholder's utility changing dramatically. In order to make our method more robust to such changes, we consider the optimal declarative processes for all possible subsets of stakeholders and make an informed decision from this based on what we observe.

Given a subset $X = \{i_1, \ldots, i_k\}$ of stakeholders $S = (S_1, \ldots, S_m)$, let us consider the reduced stakeholder utility vector for those entries given in X:

$$u^{(X)} := (u_{i_1}(T), \ldots, u_{i_k}(T))$$

For every such vector $u^{(X)}$ we will consider the declarative process that minimizes the ℓ_2-norm from it to the best case utility $(1, 1, \ldots, 1) \in \mathbb{R}^{|X|}$. The result will be a list of $2^{|S|} - 1$ declarative processes. Let $H^{(X)}(T) :=$

$\left\|u^{(X)}(T) - (1, 1, \ldots, 1)\right\|_2$. We then use the information given in this list to determine the optimal choice of declarative process for all stakeholders.

Example 5.2. Let us consider PH1, PH2a and PH2b in Example 5.1. The table in Figure 5 records the reduced stakeholder utility vectors for all possible subsets of stakeholders. For each we record in the rightmost column the optimal choice of process.

- Of the $5 + 1 = 6$ subsets X of S having size $|X| \geq |S| - 1 = 4$, we see that the optimal choice is PH2b since it appears as the answer in 5 of the 6 cases.
- Of the $10 + 5 + 1 = 16$ subsets X of S having size $|X| \geq |S|/2$ we see that the optimal choice is PH2b since it appears as the answer in 12 of the 16 cases.
- Of the $2^5 - 1 = 31$ non-empty subsets X of S we see that the optimal choice of strategy is PH2b since it appears as the answer in 20 of the 31 cases.

Each of these agrees with what we found in Example 5.1 when the subset of interest X is the set of all stakeholders S. There appears to be no clear reason to consider that either of the other processes could be optimal for this particular collection of stakeholders.

The previous example highlights the reasoning and analysis that allows us to conclude that a particular declarative process is the optimal one for particular set of stakeholders. The data in Table 5 might have been different and so to address this let us make the following comments in relation to what one should do in that event:

A. Choosing between optimal answers for $X = S$ and $X \subset S$

Let us suppose that the optimal choice in Figure 5 was not necessarily the same as the optimal choice for the other subsets of S that we considered. How should one go about deciding on an answer that can be considered robust? There are several things to consider in this setting. In the list below the word 'optimal' signifies it is optimal with respect to frequency. Suppose that

- Process T_{all} is optimal for $X = S$.
- Process $T_{\mathsf{almostall}}$ is optimal among

$$\{X \ : \ X \subseteq S \text{ with } X \neq \emptyset \text{ and } |X| \geq |S| - 1\}.$$

- Process $T_{\mathsf{morethanhalf}}$ is optimal among

$$\{X \ : \ X \subseteq S \text{ with } X \neq \emptyset \text{ and } |X| \geq |S|/2\}.$$

- Process T_{any} is optimal among

$$\{X \ : \ X \subseteq S \text{ with } X \neq \emptyset\}.$$

With these notions defined we can make the following observations.

1) If $\mathsf{all} = \mathsf{almostall}$ then, in the absence of any other information, the optimal choice of declarative process is clearly D_{all}.
2) If $\mathsf{all} \neq \mathsf{almostall}$ then it would seem that the optimal choice of declarative process needs to be considered. If our motivation is to decide on a process that is strictly optimal for **all** stakeholders then D_{all} is that process. However process $D_{\mathsf{almostall}}$ could be considered as an alternative in the event that (a) the difference between $H(T_{\mathsf{all}})$ and $H(T_{\mathsf{almostall}})$ is very small and/or (b) if there is uncertainty about whether one of the stakeholders should be considered an active stakeholder at the time. If $\mathsf{almostall} = \mathsf{morethanhalf}$ then a stronger case could be made for choosing $D_{\mathsf{almostall}}$ over D_{all}, however if $\mathsf{almostall} \neq \mathsf{morethanhalf}$ then we cannot make that case.
3) The process D_{any} might seem an odd one to consider. However it can come into play in the event that we do not know what collection of (declared) stakeholders are active stakeholders in some execution of a process.

We end this section with a brief summary of the method:

To compare the declarative stakeholder systems $T_1 = (D_1, S, G)$**, ...,** $T_n = (D_n, S, G)$**:**

Step 1 We remind ourselves that due regard must be given to stakeholder preferences in light of different declarative systems (cf. expression G_1 for stakeholder S_1 in Patient Handler 2 to the G_1 in Patient Handler 1).

Step 2 Use Theorem 4.1 to calculate the stakeholder utility vectors $u(T_i) = (u_1(T_i), \ldots, u_m(T_i))$ for all n different declarative stakeholder systems.

Step 3 For each of the $2^m - 1$ non-empty subsets X of $\{S_1, \ldots, S_m\}$, Determine the $j \in \{1, \ldots, n\}$ that minimizes $H^{(X)}(T_j)$ and denote this index by $\mathsf{Optimal}(X)$.

Step 4 Given the collection of optimal answers $\{\mathsf{Optimal}(X) \ : \ S \supseteq X \neq \emptyset\}$, analyse the four different answers $(T_{\mathsf{all}}, T_{\mathsf{almostall}}, T_{\mathsf{morethanhalf}}, T_{\mathsf{any}})$ one gets from Section 1 in order to determine the optimal process for this collection of declarative stakeholder systems.

6. Discussion

In this paper we have presented a method for quantifying stakeholder utility for declarative processes. This method allows us to decide which of a collection of declarative processes is optimal in light of stakeholder preferences. As far as we are aware, there is currently no known method in the literature for performing this type of quantification and comparison.

Our method relies on constructing a set of representative traces for a declarative process followed by considering a function on subsets of these representatives that are deemed good with respect to stakeholder preferences. The assumptions we have used in this analysis are elementary and in subsequent work more general versions might be considered. For example, attributing values to traces for particular stakeholders that are different to the indicator function on traces used to determine membership of GoodTraces. Our technique will faithfully model declarative systems for which unique traces are good representatives, and we consider this to be the case with processes whose activities happen at most once during any execution. Many real life processes that one finds in

Subset $X = \{S_{i_1}, \ldots, S_{i_k}\}$ of stakeholders	$\big(H^{(X)}(T_{PH1}),\ H^{(X)}(T_{PH2a}),\ H^{(X)}(T_{PH2b})\big)$	Process $j \in \{PH1, PH2a, PH2b\}$ that minimizes $H(T_j)$
$\{S_1\}$	(0.52337000000, 0.61490000000, 0.57351000000)	PH1
$\{S_2\}$	(0.70563000000, 0.13720000000, 0.12441000000)	PH2b
$\{S_3\}$	(0.02321000000, 0.00000000000, 0.00000000000)	PH2a
$\{S_4\}$	(0.00216000000, 0.00011000000, 0.00010000000)	PH2b
$\{S_5\}$	(0.00340000000, 0.00001000000, 0.00001000000)	PH2a
$\{S_1, S_2\}$	(0.87853847599, 0.63002051554, 0.58684884613)	PH2b
$\{S_1, S_3\}$	(0.52388439660, 0.61490000000, 0.57351000000)	PH1
$\{S_1, S_4\}$	(0.52337445725, 0.61490000984, 0.57351000872)	PH1
$\{S_1, S_5\}$	(0.52338104370, 0.61490000008, 0.57351000009)	PH1
$\{S_2, S_3\}$	(0.70601161534, 0.13720000000, 0.12441000000)	PH2b
$\{S_2, S_4\}$	(0.70563330597, 0.13720004410, 0.12441004019)	PH2b
$\{S_2, S_5\}$	(0.70563819121, 0.13720000036, 0.12441000040)	PH2b
$\{S_3, S_4\}$	(0.02331029172, 0.00011000000, 0.00010000000)	PH2b
$\{S_3, S_5\}$	(0.02345770875, 0.00001000000, 0.00001000000)	PH2a
$\{S_4, S_5\}$	(0.00402810129, 0.00011045361, 0.00010049875)	PH2b
$\{S_1, S_2, S_3\}$	(0.87884501358, 0.63002051554, 0.58684884613)	PH2b
$\{S_1, S_2, S_4\}$	(0.87854113131, 0.63002052514, 0.58684885465)	PH2b
$\{S_1, S_2, S_5\}$	(0.87854505508, 0.63002051562, 0.58684884621)	PH2b
$\{S_1, S_3, S_4\}$	(0.52388884947, 0.61490000984, 0.57351000872)	PH1
$\{S_1, S_3, S_5\}$	(0.52389542945, 0.61490000008, 0.57351000009)	PH1
$\{S_1, S_4, S_5\}$	(0.52338550085, 0.61490000992, 0.57351000880)	PH1
$\{S_2, S_3, S_4\}$	(0.70601491953, 0.13720004410, 0.12441004019)	PH2b
$\{S_2, S_3, S_5\}$	(0.70601980213, 0.13720000036, 0.12441000040)	PH2b
$\{S_2, S_4, S_5\}$	(0.70564149715, 0.13720004446, 0.12441004059)	PH2b
$\{S_3, S_4, S_5\}$	(0.02355694589, 0.00011045361, 0.000100498756)	PH2b
$\{S_1, S_2, S_3, S_4\}$	(0.87884766797, 0.63002052514, 0.58684885465)	PH2b
$\{S_1, S_2, S_3, S_5\}$	(0.87885159037, 0.63002051562, 0.58684884621)	PH2b
$\{S_1, S_2, S_4, S_5\}$	(0.87854771037, 0.63002052522, 0.58684885473)	PH2b
$\{S_1, S_3, S_4, S_5\}$	(0.52389988223, 0.61490000992, 0.57351000880)	PH1
$\{S_2, S_3, S_4, S_5\}$	(0.70602310628, 0.13720004446, 0.12441004059)	PH2b
$\{S_1, S_2, S_3, S_4, S_5\}$	(0.87885424474, 0.63002052522, 0.58684885473)	PH2b

Figure 5. The table for Example 5.2.

the application areas of business process models, healthcare workflow analysis, and policy process analysis can be seen to be such systems. Again, in subsequent work it may well be worth considering allowing activities to happen, say, at most twice but such an allowance will naturally lead to further complexity in the calculations that are necessary.

A challenge in this analysis is constructing the set of unique traces of a declarative process. This challenging enumeration problem is a computationally demanding task since the more activities in a declarative process the more time and space this will take. However, this can be overcome in part by a combinatorial analysis of the declarative process graph as was evidenced by the discussion of the *core* of a *declarative process* in Section 3 along with the algorithmic considerations. The paper lays the foundation for further investigations into this area of declarative system analysis.

REFERENCES

[1] A. A. Casey. A Declarative Model of the Policy Process. PhD thesis, University College Dublin, in preparation.

[2] M. Dukes and A. A. Casey. Combinatorial diversity metrics for declarative processes: an application to policy process analysis. *International Journal of General Systems*, to appear, 2021. arXiv:2008.10401.

[3] W. van der Aalst. *Process Mining: Data Science in Action*. Second edition. Springer, 2016. doi: 10.1007/978-3-662-49851-4.

[4] S. Mertens, F. Gailly, and G. Poels. Enhancing Declarative Process Models with DMN Decision Logic. In: *Gaaloul K., Schmidt R., Nurcan S., Guerreiro S., Ma Q. (eds) Enterprise, Business-Process and Information Systems Modeling. BPMDS 2015, EMMSAD 2015. Lecture Notes in Business Information Processing*, vol 214. Springer, 2015.

[5] T. Hildebrandt and R. Mukkamala. Declarative Event-Based Workflow as Distributed Dynamic Condition Response Graphs. PLACES 2010. *Electronic Proceedings in Theoretical Computer Science* **69**:59–73, 2010.

[6] W. van Der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - Research and Development* **23**, no. 2, 99-113, 2009.

[7] M. Pesic, H. Schonenberg, and W. van der Aalst. DECLARE: Full support for loosely-structured processes. In *EDOC. IEEE Computer Society*, 287–300. doi: 10.1109/EDOC.2007.25, 2007.

[8] J. Li, L. Zhang, G. Pu, M. Y. Vardi, and J. He. LTLf satisfiability checking. arXiv:1403.1666, 2014.

[9] D.M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-dimensional modal logics: theory and applications*. Elsevier, 2003.

[10] M. Howlett, M. Ramesh, and A. Perl. Studying Public Policy: Principles and Processes. Fourth Edition, Oxford University Press, 2020.

[11] T. Hildebrandt, R. Mukkamala R,and T. Slaats. Declarative Modelling and Safe Distribution of Healthcare Workflows. In: *Liu Z., Wassyng A. (eds) Foundations of Health Informatics Engineering and Systems. FHIES 2011. Lecture Notes in Computer Science*, vol. 7151. Springer, Berlin, Heidelberg, 2012.

[12] F. Chesani, P. Mello, M. Montali, and S. Storari. Testing Careflow Process Execution Conformance by Translating a Graphical Language to Computational Logic. In: *Bellazzi, R., Abu-Hanna, A., Hunter, J. (eds.) AIME 2007. Lecture Notes in Computer Science (LNAI)*, vol. 4594, pp. 479–488. Springer, Heidelberg, 2007.

[13] R. Dijkman, M. Dumas, B. van Dongen, R. Käärikb, and J. Mendling. Similarity of business process models: Metrics and evaluation. *Information Systems* **36** (2011), no. 2, 498–516.

[14] A. Schoknecht, T. Thaler, P. Fettke, A. Oberweis, and R. Laue. Similarity of Business Process Models – A State-of-the-Art Analysis. ACM Computing Surveys **50** (2017), no. 4. Article No. 52.

[15] C. S. Wahyuni, K. R. Sungkono, and R. Sarno. Novel Parallel Business Process Similarity Methods Based on Weighted-Tree Declarative Pattern Models. *International Journal of Intelligent Engineering and Systems* **12**, no.6, doi: 10.22266/ijies2019.1231.23, 2019.

[16] S. Mertens, F. Gailly, D. Van Sassenbroeck, and G. Poels. Integrated Declarative Process and Decision Discovery of the Emergency Care Process. *Information Systems Frontiers* doi: 10.1007/s10796-020-10078-5, 2020.

[17] B. A. Furtado, M. A. Fuentes, and C.J.Tessone. Public Policy Modeling and Applications: state-of-the-art and perspectives. *Complexity* (special issue), 2019.

[18] B. A. Furtado, P. A. M. Sakowski, and M.H. Tóvolli. *Modeling Complex Systems for Public Policies*. IPEA, 2015. ISBN: 9788578112493.

UniqueTraces(D_{PH1})

length	traces						number
2	(1, 8)						1
3	(1, 2, 3)	(1, 2, 6)	(1, 5, 8)	(1, 8, 2)	(1, 8, 5)	(1, 2, 8)	7
	(1, 2, 4)						
4	(1, 2, 3, 4)	(1, 2, 4, 3)	(1, 2, 5, 6)	(1, 2, 8, 3)	(1, 5, 2, 4)	(1, 8, 2, 4)	29
	(1, 2, 3, 5)	(1, 2, 4, 5)	(1, 2, 5, 8)	(1, 2, 8, 4)	(1, 5, 2, 6)	(1, 8, 2, 5)	
	(1, 2, 3, 6)	(1, 2, 4, 8)	(1, 2, 6, 3)	(1, 2, 8, 5)	(1, 5, 2, 8)	(1, 8, 2, 6)	
	(1, 2, 3, 7)	(1, 2, 5, 3)	(1, 2, 6, 5)	(1, 2, 8, 6)	(1, 5, 8, 2)	(1, 8, 5, 2)	
	(1, 2, 3, 8)	(1, 2, 5, 4)	(1, 2, 6, 8)	(1, 5, 2, 3)	(1, 8, 2, 3)		
5	(1, 2, 3, 4, 5)	(1, 2, 3, 8, 4)	(1, 2, 5, 3, 8)	(1, 2, 6, 8, 5)	(1, 5, 2, 3, 7)	(1, 8, 2, 3, 6)	82
	(1, 2, 3, 4, 7)	(1, 2, 3, 8, 5)	(1, 2, 5, 4, 3)	(1, 2, 8, 3, 4)	(1, 5, 2, 3, 8)	(1, 8, 2, 3, 7)	
	(1, 2, 3, 4, 8)	(1, 2, 3, 8, 6)	(1, 2, 5, 4, 8)	(1, 2, 8, 3, 5)	(1, 5, 2, 4, 3)	(1, 8, 2, 4, 3)	
	(1, 2, 3, 5, 4)	(1, 2, 3, 8, 7)	(1, 2, 5, 6, 3)	(1, 2, 8, 3, 6)	(1, 5, 2, 4, 8)	(1, 8, 2, 4, 5)	
	(1, 2, 3, 5, 6)	(1, 2, 4, 3, 5)	(1, 2, 5, 6, 8)	(1, 2, 8, 3, 7)	(1, 5, 2, 6, 3)	(1, 8, 2, 5, 3)	
	(1, 2, 3, 5, 7)	(1, 2, 4, 3, 7)	(1, 2, 5, 8, 3)	(1, 2, 8, 4, 3)	(1, 5, 2, 6, 8)	(1, 8, 2, 5, 4)	
	(1, 2, 3, 5, 8)	(1, 2, 4, 3, 8)	(1, 2, 5, 8, 4)	(1, 2, 8, 4, 5)	(1, 5, 2, 8, 3)	(1, 8, 2, 5, 6)	
	(1, 2, 3, 6, 5)	(1, 2, 4, 5, 3)	(1, 2, 5, 8, 6)	(1, 2, 8, 5, 3)	(1, 5, 2, 8, 4)	(1, 8, 2, 6, 3)	
	(1, 2, 3, 6, 7)	(1, 2, 4, 5, 8)	(1, 2, 6, 3, 5)	(1, 2, 8, 5, 4)	(1, 5, 2, 8, 6)	(1, 8, 2, 6, 5)	
	(1, 2, 3, 6, 8)	(1, 2, 4, 8, 3)	(1, 2, 6, 3, 7)	(1, 2, 8, 5, 6)	(1, 5, 8, 2, 3)	(1, 8, 5, 2, 3)	
	(1, 2, 3, 7, 4)	(1, 2, 4, 8, 5)	(1, 2, 6, 3, 8)	(1, 2, 8, 6, 3)	(1, 5, 8, 2, 4)	(1, 8, 5, 2, 4)	
	(1, 2, 3, 7, 5)	(1, 2, 5, 3, 4)	(1, 2, 6, 5, 3)	(1, 2, 8, 6, 5)	(1, 5, 8, 2, 6)	(1, 8, 5, 2, 6)	
	(1, 2, 3, 7, 6)	(1, 2, 5, 3, 6)	(1, 2, 6, 5, 8)	(1, 5, 2, 3, 4)	(1, 8, 2, 3, 4)		
	(1, 2, 3, 7, 8)	(1, 2, 5, 3, 7)	(1, 2, 6, 8, 3)	(1, 5, 2, 3, 6)	(1, 8, 2, 3, 5)		
6	(1, 2, 3, 4, 5, 7)	(1, 2, 3, 7, 6, 5)	(1, 2, 5, 3, 4, 7)	(1, 2, 6, 5, 3, 7)	(1, 5, 2, 3, 4, 7)	(1, 8, 2, 3, 4, 7)	160
	(1, 2, 3, 4, 5, 8)	(1, 2, 3, 7, 6, 8)	(1, 2, 5, 3, 4, 8)	(1, 2, 6, 5, 3, 8)	(1, 5, 2, 3, 4, 8)	(1, 8, 2, 3, 5, 4)	
	(1, 2, 3, 4, 7, 5)	(1, 2, 3, 7, 8, 4)	(1, 2, 5, 3, 6, 7)	(1, 2, 6, 5, 8, 3)	(1, 5, 2, 3, 6, 7)	(1, 8, 2, 3, 5, 6)	
	(1, 2, 3, 4, 7, 8)	(1, 2, 3, 7, 8, 5)	(1, 2, 5, 3, 6, 8)	(1, 2, 6, 8, 3, 5)	(1, 5, 2, 3, 6, 8)	(1, 8, 2, 3, 5, 7)	
	(1, 2, 3, 4, 8, 5)	(1, 2, 3, 7, 8, 6)	(1, 2, 5, 3, 7, 4)	(1, 2, 6, 8, 3, 7)	(1, 5, 2, 3, 7, 4)	(1, 8, 2, 3, 6, 5)	
	(1, 2, 3, 4, 8, 7)	(1, 2, 3, 8, 4, 5)	(1, 2, 5, 3, 7, 6)	(1, 2, 6, 8, 5, 3)	(1, 5, 2, 3, 7, 6)	(1, 8, 2, 3, 6, 7)	
	(1, 2, 3, 5, 4, 7)	(1, 2, 3, 8, 4, 7)	(1, 2, 5, 3, 7, 8)	(1, 2, 8, 3, 4, 5)	(1, 5, 2, 3, 7, 8)	(1, 8, 2, 3, 7, 4)	
	(1, 2, 3, 5, 4, 8)	(1, 2, 3, 8, 5, 4)	(1, 2, 5, 3, 8, 4)	(1, 2, 8, 3, 4, 7)	(1, 5, 2, 3, 8, 4)	(1, 8, 2, 3, 7, 5)	
	(1, 2, 3, 5, 6, 7)	(1, 2, 3, 8, 5, 6)	(1, 2, 5, 3, 8, 6)	(1, 2, 8, 3, 5, 4)	(1, 5, 2, 3, 8, 6)	(1, 8, 2, 3, 7, 6)	
	(1, 2, 3, 5, 6, 8)	(1, 2, 3, 8, 5, 7)	(1, 2, 5, 3, 8, 7)	(1, 2, 8, 3, 5, 6)	(1, 5, 2, 3, 8, 7)	(1, 8, 2, 4, 3, 5)	
	(1, 2, 3, 5, 7, 4)	(1, 2, 3, 8, 6, 5)	(1, 2, 5, 4, 3, 7)	(1, 2, 8, 3, 5, 7)	(1, 5, 2, 4, 3, 7)	(1, 8, 2, 4, 3, 7)	
	(1, 2, 3, 5, 7, 6)	(1, 2, 3, 8, 6, 7)	(1, 2, 5, 4, 3, 8)	(1, 2, 8, 3, 6, 5)	(1, 5, 2, 4, 3, 8)	(1, 8, 2, 4, 5, 3)	
	(1, 2, 3, 5, 7, 8)	(1, 2, 3, 8, 7, 4)	(1, 2, 5, 4, 8, 3)	(1, 2, 8, 3, 6, 7)	(1, 5, 2, 4, 8, 3)	(1, 8, 2, 5, 3, 4)	
	(1, 2, 3, 5, 8, 4)	(1, 2, 3, 8, 7, 5)	(1, 2, 5, 6, 3, 7)	(1, 2, 8, 3, 7, 4)	(1, 5, 2, 6, 3, 7)	(1, 8, 2, 5, 3, 6)	
	(1, 2, 3, 5, 8, 6)	(1, 2, 3, 8, 7, 6)	(1, 2, 5, 6, 3, 8)	(1, 2, 8, 3, 7, 5)	(1, 5, 2, 6, 3, 8)	(1, 8, 2, 5, 3, 7)	
	(1, 2, 3, 5, 8, 7)	(1, 2, 4, 3, 5, 7)	(1, 2, 5, 6, 8, 3)	(1, 2, 8, 3, 7, 6)	(1, 5, 2, 6, 8, 3)	(1, 8, 2, 5, 4, 3)	
	(1, 2, 3, 6, 5, 7)	(1, 2, 4, 3, 5, 8)	(1, 2, 5, 8, 3, 4)	(1, 2, 8, 4, 3, 7)	(1, 5, 2, 8, 3, 6)	(1, 8, 2, 6, 3, 5)	
	(1, 2, 3, 6, 5, 8)	(1, 2, 4, 3, 7, 5)	(1, 2, 5, 8, 3, 6)	(1, 2, 8, 4, 3, 5)	(1, 5, 2, 8, 3, 7)	(1, 8, 2, 6, 3, 7)	
	(1, 2, 3, 6, 7, 8)	(1, 2, 4, 3, 7, 8)	(1, 2, 6, 3, 5, 7)	(1, 2, 8, 4, 5, 3)	(1, 5, 8, 2, 3, 6)	(1, 8, 5, 2, 3, 6)	
	(1, 2, 3, 6, 8, 5)	(1, 2, 4, 3, 8, 7)	(1, 2, 6, 3, 5, 8)	(1, 2, 8, 5, 3, 6)	(1, 5, 8, 2, 6, 3)	(1, 8, 5, 2, 3, 4)	
	(1, 2, 3, 6, 8, 7)	(1, 2, 4, 5, 3, 7)	(1, 2, 6, 3, 5, 7)	(1, 2, 8, 5, 3, 7)	(1, 5, 8, 2, 3, 4)	(1, 8, 5, 2, 3, 6)	
	(1, 2, 3, 7, 4, 5)	(1, 2, 4, 5, 3, 8)	(1, 2, 6, 3, 5, 8)	(1, 2, 8, 5, 4, 3)	(1, 5, 8, 2, 3, 6)	(1, 8, 5, 2, 3, 7)	
	(1, 2, 3, 7, 4, 8)	(1, 2, 4, 5, 8, 3)	(1, 2, 6, 3, 7, 5)	(1, 2, 8, 5, 6, 3)	(1, 5, 8, 2, 3, 7)	(1, 8, 5, 2, 4, 3)	
	(1, 2, 3, 7, 5, 4)	(1, 2, 4, 8, 3, 5)	(1, 2, 6, 3, 7, 8)	(1, 2, 8, 6, 3, 5)	(1, 5, 8, 2, 4, 3)	(1, 8, 5, 2, 6, 3)	
	(1, 2, 3, 7, 5, 6)	(1, 2, 4, 8, 3, 7)	(1, 2, 6, 3, 8, 5)	(1, 2, 8, 6, 3, 7)	(1, 5, 8, 2, 6, 3)		
	(1, 2, 3, 7, 5, 8)	(1, 2, 4, 8, 5, 3)	(1, 2, 6, 3, 8, 7)	(1, 2, 8, 6, 5, 3)	(1, 8, 2, 3, 4, 5)		
7	(1, 2, 3, 4, 5, 7, 8)	(1, 2, 3, 7, 6, 5, 8)	(1, 2, 5, 3, 4, 7, 8)	(1, 2, 6, 5, 3, 7, 8)	(1, 5, 2, 3, 4, 7, 8)	(1, 8, 2, 3, 4, 5, 7)	180
	(1, 2, 3, 4, 5, 8, 7)	(1, 2, 3, 7, 6, 8, 5)	(1, 2, 5, 3, 4, 8, 7)	(1, 2, 6, 5, 3, 8, 7)	(1, 5, 2, 3, 4, 8, 7)	(1, 8, 2, 3, 4, 7, 5)	
	(1, 2, 3, 4, 7, 5, 8)	(1, 2, 3, 7, 8, 4, 5)	(1, 2, 5, 3, 6, 7, 8)	(1, 2, 6, 5, 8, 3, 7)	(1, 5, 2, 3, 6, 7, 8)	(1, 8, 2, 3, 5, 4, 7)	
	(1, 2, 3, 4, 7, 8, 5)	(1, 2, 3, 7, 8, 5, 4)	(1, 2, 5, 3, 6, 8, 7)	(1, 2, 6, 8, 3, 5, 7)	(1, 5, 2, 3, 6, 8, 7)	(1, 8, 2, 3, 5, 6, 7)	
	(1, 2, 3, 4, 8, 5, 7)	(1, 2, 3, 7, 8, 5, 6)	(1, 2, 5, 3, 7, 4, 8)	(1, 2, 6, 8, 3, 7, 5)	(1, 5, 2, 3, 7, 4, 8)	(1, 8, 2, 3, 5, 7, 4)	
	(1, 2, 3, 4, 8, 7, 5)	(1, 2, 3, 7, 8, 6, 5)	(1, 2, 5, 3, 7, 6, 8)	(1, 2, 6, 8, 5, 3, 7)	(1, 5, 2, 3, 7, 6, 8)	(1, 8, 2, 3, 5, 7, 6)	
	(1, 2, 3, 5, 4, 7, 8)	(1, 2, 3, 8, 4, 5, 7)	(1, 2, 5, 3, 7, 8, 4)	(1, 2, 8, 3, 4, 5, 7)	(1, 5, 2, 3, 7, 8, 4)	(1, 8, 2, 3, 6, 5, 7)	
	(1, 2, 3, 5, 4, 8, 7)	(1, 2, 3, 8, 4, 7, 5)	(1, 2, 5, 3, 7, 8, 6)	(1, 2, 8, 3, 4, 7, 5)	(1, 5, 2, 3, 7, 8, 6)	(1, 8, 2, 3, 6, 7, 5)	
	(1, 2, 3, 5, 6, 7, 8)	(1, 2, 3, 8, 5, 4, 7)	(1, 2, 5, 3, 8, 4, 7)	(1, 2, 8, 3, 5, 4, 7)	(1, 5, 2, 3, 8, 4, 7)	(1, 8, 2, 3, 7, 4, 5)	
	(1, 2, 3, 5, 6, 8, 7)	(1, 2, 3, 8, 5, 6, 7)	(1, 2, 5, 3, 8, 6, 7)	(1, 2, 8, 3, 5, 6, 7)	(1, 5, 2, 3, 8, 6, 7)	(1, 8, 2, 3, 7, 5, 4)	
	(1, 2, 3, 5, 7, 4, 8)	(1, 2, 3, 8, 5, 7, 4)	(1, 2, 5, 3, 8, 7, 4)	(1, 2, 8, 3, 5, 7, 4)	(1, 5, 2, 3, 8, 7, 4)	(1, 8, 2, 3, 7, 5, 6)	
	(1, 2, 3, 5, 7, 6, 8)	(1, 2, 3, 8, 5, 7, 6)	(1, 2, 5, 3, 8, 7, 6)	(1, 2, 8, 3, 5, 7, 6)	(1, 5, 2, 3, 8, 7, 6)	(1, 8, 2, 3, 7, 6, 5)	
	(1, 2, 3, 5, 7, 8, 4)	(1, 2, 3, 8, 6, 5, 7)	(1, 2, 5, 4, 3, 7, 8)	(1, 2, 8, 3, 6, 5, 7)	(1, 5, 2, 4, 3, 7, 8)	(1, 8, 2, 4, 3, 5, 7)	
	(1, 2, 3, 5, 7, 8, 6)	(1, 2, 3, 8, 6, 7, 5)	(1, 2, 5, 4, 3, 8, 7)	(1, 2, 8, 3, 6, 7, 5)	(1, 5, 2, 4, 3, 8, 7)	(1, 8, 2, 4, 3, 7, 5)	
	(1, 2, 3, 5, 8, 4, 7)	(1, 2, 3, 8, 7, 4, 5)	(1, 2, 5, 4, 8, 3, 7)	(1, 2, 8, 3, 7, 4, 5)	(1, 5, 2, 4, 8, 3, 7)	(1, 8, 2, 4, 5, 3, 7)	
	(1, 2, 3, 5, 8, 6, 7)	(1, 2, 3, 8, 7, 5, 4)	(1, 2, 5, 6, 3, 7, 8)	(1, 2, 8, 3, 7, 5, 4)	(1, 5, 2, 6, 3, 7, 8)	(1, 8, 2, 5, 3, 4, 7)	
	(1, 2, 3, 5, 8, 7, 4)	(1, 2, 3, 8, 7, 5, 6)	(1, 2, 5, 6, 3, 8, 7)	(1, 2, 8, 3, 7, 5, 6)	(1, 5, 2, 6, 3, 8, 7)	(1, 8, 2, 5, 3, 6, 7)	
	(1, 2, 3, 5, 8, 7, 6)	(1, 2, 3, 8, 7, 6, 5)	(1, 2, 5, 6, 8, 3, 7)	(1, 2, 8, 3, 7, 6, 5)	(1, 5, 2, 6, 8, 3, 7)	(1, 8, 2, 5, 3, 7, 4)	
	(1, 2, 3, 6, 5, 7, 8)	(1, 2, 4, 3, 5, 7, 8)	(1, 2, 5, 8, 3, 4, 7)	(1, 2, 8, 4, 3, 5, 7)	(1, 5, 2, 8, 3, 4, 7)	(1, 8, 2, 5, 3, 7, 6)	
	(1, 2, 3, 6, 5, 8, 7)	(1, 2, 4, 3, 5, 8, 7)	(1, 2, 5, 8, 3, 6, 7)	(1, 2, 8, 4, 3, 7, 5)	(1, 5, 2, 8, 3, 6, 7)	(1, 8, 2, 5, 4, 3, 7)	
	(1, 2, 3, 6, 7, 5, 8)	(1, 2, 4, 3, 7, 5, 8)	(1, 2, 5, 8, 3, 7, 4)	(1, 2, 8, 4, 5, 3, 7)	(1, 5, 2, 8, 3, 7, 4)	(1, 8, 2, 5, 6, 3, 7)	
	(1, 2, 3, 6, 7, 8, 5)	(1, 2, 4, 3, 7, 8, 5)	(1, 2, 5, 8, 3, 7, 6)	(1, 2, 8, 5, 3, 4, 7)	(1, 5, 2, 8, 3, 7, 6)	(1, 8, 2, 6, 3, 5, 7)	
	(1, 2, 3, 6, 8, 5, 7)	(1, 2, 4, 3, 8, 5, 7)	(1, 2, 5, 8, 4, 3, 7)	(1, 2, 8, 5, 3, 6, 7)	(1, 5, 2, 8, 4, 3, 7)	(1, 8, 2, 6, 3, 7, 5)	
	(1, 2, 3, 6, 8, 7, 5)	(1, 2, 4, 3, 8, 7, 5)	(1, 2, 5, 8, 6, 3, 7)	(1, 2, 8, 5, 3, 7, 4)	(1, 5, 2, 8, 6, 3, 7)	(1, 8, 2, 6, 5, 3, 7)	
	(1, 2, 3, 7, 4, 5, 8)	(1, 2, 4, 5, 3, 7, 8)	(1, 2, 6, 3, 5, 7, 8)	(1, 2, 8, 5, 3, 7, 6)	(1, 5, 8, 2, 3, 4, 7)	(1, 8, 5, 2, 3, 4, 7)	
	(1, 2, 3, 7, 4, 8, 5)	(1, 2, 4, 5, 3, 8, 7)	(1, 2, 6, 3, 5, 8, 7)	(1, 2, 8, 5, 4, 3, 7)	(1, 5, 8, 2, 3, 6, 7)	(1, 8, 5, 2, 3, 6, 7)	
	(1, 2, 3, 7, 5, 4, 8)	(1, 2, 4, 5, 8, 3, 7)	(1, 2, 6, 3, 7, 5, 8)	(1, 2, 8, 5, 6, 3, 7)	(1, 5, 8, 2, 3, 7, 4)	(1, 8, 5, 2, 3, 7, 4)	
	(1, 2, 3, 7, 5, 6, 8)	(1, 2, 4, 8, 3, 5, 7)	(1, 2, 6, 3, 7, 8, 5)	(1, 2, 8, 6, 3, 5, 7)	(1, 5, 8, 2, 3, 7, 6)	(1, 8, 5, 2, 3, 7, 6)	
	(1, 2, 3, 7, 5, 8, 4)	(1, 2, 4, 8, 3, 7, 5)	(1, 2, 6, 3, 8, 5, 7)	(1, 2, 8, 6, 3, 7, 5)	(1, 5, 8, 2, 4, 3, 7)	(1, 8, 5, 2, 4, 3, 7)	
	(1, 2, 3, 7, 5, 8, 6)	(1, 2, 4, 8, 5, 3, 7)	(1, 2, 6, 3, 8, 7, 5)	(1, 2, 8, 6, 5, 3, 7)	(1, 5, 8, 2, 6, 3, 7)	(1, 8, 5, 2, 6, 3, 7)	

Figure 6. The unique traces for the Patient Handler 1 example D_{PH1}

UniqueTraces(D_{PH2a}) summary

length	traces					number
2	$(1,4)$	$(1,6)$	$(1,8)$	$(1,12)$		4
3	$(1,4,6)$	$(1,8,6)$	$(1,8,10)$	$(1,2,4)$	$(1,51,8)$	60
	$(1,6,4)$	$(1,6,8)$	$(1,10,8)$	$(1,4,2)$	$(1,8,51)$	
	$(1,8,4)$	$(1,9,6)$	$(1,8,12)$	$(1,2,6)$	$(1,51,12)$	
	$(1,4,8)$	$(1,6,9)$	$(1,12,8)$	$(1,6,2)$	$(1,12,51)$	
	$(1,9,4)$	$(1,10,6)$	$(1,8,53)$	$(1,2,8)$	$(1,52,4)$	
	$(1,4,9)$	$(1,6,10)$	$(1,53,8)$	$(1,8,2)$	$(1,4,52)$	
	$(1,10,4)$	$(1,12,6)$	$(1,9,12)$	$(1,2,12)$	$(1,52,6)$	
	$(1,4,10)$	$(1,6,12)$	$(1,12,9)$	$(1,12,2)$	$(1,6,52)$	
	$(1,12,4)$	$(1,53,6)$	$(1,10,12)$	$(1,51,4)$	$(1,52,8)$	
	$(1,4,12)$	$(1,6,53)$	$(1,12,10)$	$(1,4,51)$	$(1,8,52)$	
	$(1,4,53)$	$(1,8,9)$	$(1,12,53)$	$(1,51,6)$	$(1,52,12)$	
	$(1,53,4)$	$(1,9,8)$	$(1,53,12)$	$(1,6,51)$	$(1,12,52)$	
4	$(1,8,4,6),\ \ldots$					552
5	$(1,3,9,10,6),\ \ldots$					3726
6	$(1,3,9,10,4,6),\ \ldots$					19404
7	$(1,3,4,6,7,9,10),\ \ldots$					79164
8	$(1,3,4,6,7,8,9,10),\ \ldots$					257040
9	$(1,3,4,6,7,8,9,10,12),\ \ldots$					715680
10	$(1,2,3,4,6,7,8,9,19,12),\ \ldots$					1995840
11	$(1,51,2,3,4,6,7,8,9,10,12),\ \ldots$					5261760
12	$(1,52,51,2,3,4,6,7,8,9,10,12),\ \ldots$					7983360

Figure 7. Summary of the unique traces for Patient Handler D_{PH2a}

UniqueTraces(D_{PH2b}) summary

length	traces					number
2	$(1,4)$	$(1,6)$	$(1,8)$	$(1,12)$		4
3	$(1,11,4)$	$(1,10,4)$	$(1,53,6)$	$(1,12,53)$	$(1,51,8)$	68
	$(1,4,11)$	$(1,4,10)$	$(1,6,53)$	$(1,53,12)$	$(1,8,51)$	
	$(1,11,6)$	$(1,12,4)$	$(1,8,9)$	$(1,2,4)$	$(1,51,12)$	
	$(1,6,11)$	$(1,4,12)$	$(1,9,8)$	$(1,4,2)$	$(1,12,51)$	
	$(1,11,8)$	$(1,4,53)$	$(1,8,10)$	$(1,2,6)$	$(1,52,4)$	
	$(1,8,11)$	$(1,53,4)$	$(1,10,8)$	$(1,6,2)$	$(1,4,52)$	
	$(1,11,12)$	$(1,8,6)$	$(1,8,12)$	$(1,2,8)$	$(1,52,6)$	
	$(1,12,11)$	$(1,6,8)$	$(1,12,8)$	$(1,8,2)$	$(1,6,52)$	
	$(1,4,6)$	$(1,9,6)$	$(1,8,53)$	$(1,2,12)$	$(1,52,8)$	
	$(1,6,4)$	$(1,6,9)$	$(1,53,8)$	$(1,12,2)$	$(1,8,52)$	
	$(1,8,4)$	$(1,10,6)$	$(1,9,12)$	$(1,51,4)$	$(1,52,12)$	
	$(1,4,8)$	$(1,6,10)$	$(1,12,9)$	$(1,4,51)$	$(1,12,52)$	
	$(1,9,4)$	$(1,12,6)$	$(1,10,12)$	$(1,51,6)$		
	$(1,4,9)$	$(1,6,12)$	$(1,12,10)$	$(1,6,51)$		
4	$(1,11,4,6),\ (1,4,11,6),\ \ldots$					732
5	$(1,11,8,4,6),\ (1,8,11,4,6),\ \ldots$					5934
6	$(1,11,3,9,10,6),\ (1,3,11,9,10,6),\ \ldots$					38034
7	$(1,11,3,9,10,4,6),\ (1,3,11,9,10,4,6),\ \ldots$					195588
8	$(1,11,3,4,6,7,9,10),\ (1,3,11,4,6,7,9,10),\ \ldots$					811188
9	$(1,11,3,4,6,7,8,9,10),\ (1,3,11,4,6,7,8,9,10),\ \ldots$					2772000
10	$(1,11,3,4,6,7,8,9,10,12),\ (1,3,11,4,6,7,8,9,10,12),\ \ldots$					8436960
11	$(1,11,2,3,4,6,7,8,9,19,12),\ (1,2,11,3,4,6,7,8,9,19,12),\ \ldots$					25220160
12	$(1,11,51,2,3,4,6,7,8,9,10,12),\ (1,51,11,2,3,4,6,7,8,9,10,12),\ \ldots$					65862720
13	$(1,11,52,51,2,3,4,6,7,8,9,10,12),\ (1,52,11,51,2,3,4,6,7,8,9,10,12),\ \ldots$					95800320

Figure 8. Summary of the unique traces for Patient Handler D_{PH2b}