This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

Temporal Conformance Analysis and Explanation of Clinical Guidelines Execution: an Answer Set Programming approach

Matteo Spiotta, Paolo Terenziani, Daniele Theseider Dupré

Abstract—Clinical Guidelines (CGs) provide general evidence-based recommendations and physicians often have to resort also to their Basic Medical Knowledge (BMK) to cope with specific patients. In this paper we explore the interplay between CGs and BMK from the viewpoint of a-posteriori conformance analysis, intended as the adherence of a specific execution log to both the CG and the BMK. In this paper we consider also the temporal dimension: the guideline may include temporal constraints for the execution of actions, and its adaptation to a specific patient and context may add or modify conditions and temporal constraints for actions. We propose an approach for analyzing execution traces in Answer Set Programming with respect to a guideline and BMK, pointing out discrepancies – including temporal discrepancies – with respect to the different knowledge sources, and providing explanations regarding how the applications of the CG and the BMK have interacted, especially in case strictly adhering to both is not possible.

Index Terms—Knowledge Management, Knowledge modeling, Conformance analysis, Clinical Guidelines, Answer Set Programming, Temporal reasoning.

1 Introduction

A Clinical Guideline (CG) is "a systematically developed statement to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances" [1]. CGs are developed in order to capture medical evidence and to put it into practice, and deal with *typical* classes of patients, i.e., defining *default* processes for diagnosis, treatment and follow-up of patients, since the CG developers cannot define all possible executions of a CG on any specific patient in any clinical condition. When diagnosing and treating "atypical" patients, physicians have to resort to their Basic Medical Knowledge (BMK), i.e., the different forms of medical knowledge that physicians have acquired during their studies and clinical practice.

The interplay between CG and BMK recommendations can be quite complex. For instance, the BMK may recommend not to perform an action prescribed by the CG, but no general principle can be given for resolving such conflicts [2]. Such a complexity significantly increases in case the temporal dimension is taken into account: indeed, (i) temporal information is an intrinsic part of most CGs and BMK, and (ii) the interplay between CGs and BMK occurs in time. Regarding issue (i), actions may be temporally constrained with respect to preconditions or to other actions (e.g., in case of hip fracture, surgery is recommended within 36 hours after admission). Considering (ii), there are cases in which CG and BMK recommendations are contradictory, but in most cases the two recommendations should be "merged" along time. Typical cases are the treatment of exceptions [3]–[5], which, depending on the situation, may be treated as soon as they occur (thus suspending CG execution), delayed after the end of the CG, or executed concurrently with it. In all cases, some of the temporal constraints (in the CG and/or in the BMK) may be violated.

Unfortunately, the proper solution for managing the interplay between CGs and BMK is usually situation- and patient-dependent, and, in general, expecting that a model could provide such solutions is not realistic. Nevertheless, computer science can support physicians in the analysis of such an interplay, considering also patient data and contextual information. This is the challenging goal of the approach in this paper. In particular, we explore, with specific attention to the temporal dimension, the interplay between CGs and BMK from the viewpoint of *a posteriori* conformance analysis [6], intended as the adherence of an observed CG execution trace to both the CG and BMK. We do not provide an evaluation of how the interplay has been managed (i.e., whether the treatment was appropriate or not): we aim at identifying, in the trace, situations in which some recommendation (either in the CG or in the BMK) has not been followed, and at providing possible explanations for situations of non-conformance. In such a way, conformant treatments (to both the CG and the BMK) can be identified, as well as treatments that, while being non conformant to some individual piece of knowledge, have an explanation based on the interplay of knowledge. Therefore, the main advances with respect to the state of the art are:

- complementing conformance analysis with an explanation of how potentially contrasting knowledge sources (CG and BMK) have been applied to a specific case;
- the *temporal* analysis of conformance and explanation.

Different pieces of knowledge describe default be-

This research is original and has financial support of the Università del Piemonte Orientale.
The authors are with DISIT, Sezione di Informatica, Università del Piemonte Orientale.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

haviour, and explanations of actual behaviour should be searched in terms of exceptions to (possibly conflicting) knowledge sources; such exceptions include alterations to the "normal" timing of actions. Then, an important goal is that explanations are close to the way physicians reason on the problem. We then chose to model knowledge and reasoning in Answer Set Programming (ASP) [7]. In fact, ASP (see Appendix A) is a declarative problem solving approach which has been successfully applied to computationally difficult search problems, and, being nonmonotonic, is especially suitable for formalizing knowledge and reasoning in terms of default behaviour and exceptions in a modular, *elaboration tolerant* way [8], i.e., a way where an additive elaboration of the problem (including the addition of a class of exceptions) corresponds to the addition of formulae, rather than modification of previous ones.

We developed the approach, in particular, for the GLARE formalism [9], [10] for specifying CGs; it is briefly presented in section 2, together with the other inputs to the conformance framework; in particular, we describe the form of BMK rules we take into account in the analysis. In section 3 we provide a logical formalization of the CG execution in time, which is the basis for conformance analysis, and in section 4 we provide an integrated model describing how the recommendations in the CG and the BMK could interact. Our approach relies on medical terminology, based on SNOMED-CT [11], and terminological reasoning, in order to match specific actions and situations in the log to the general ones mentioned in the CG and BMK (see section 5). In section 6 we describe our general approach to conformance analysis and explanation, while in section 7 we describe our framework for conformance reasoning and how it is realized in ASP. We show how, once the input to the framework is represented in ASP, suitable rules can be used to let the ASP solver infer discrepancies between the execution trace in the log and the integrated CG/BMK execution model described in section 4. We describe the results of the framework on an example (section 8) and an evaluation of its scalability (section 9). Finally, we discuss related work.

2 Input to the Framework

At least four different types of data/knowledge sources should be considered to analyze compliance: patient data, contextual data, CG model, and BMK.

We distinguish two types of **patient data** collected in a *log*. We consider *patient findings*, i.e., data which are usually collected in patients' EHR. In particular, we assume that all data required during patient treatment are available, and that all such pieces of information are temporally tagged. We also assume to have a complete trace of all the clinical *actions* executed on the patient, in which each occurrence of actions is temporally tagged. Specifically, we assume that the start and end time of all actions are recorded. We also assume to have in the log the relevant **contextual data** such as personnel and resources availability.

Our approach is not biased towards a specific **CG formalism**; however, it has been developed using the GLARE formalism [9], [10] as a concrete example, due to its specific attention to the temporal aspects. GLARE provides the possibility of specifying therapeutic and diagnostic decisions

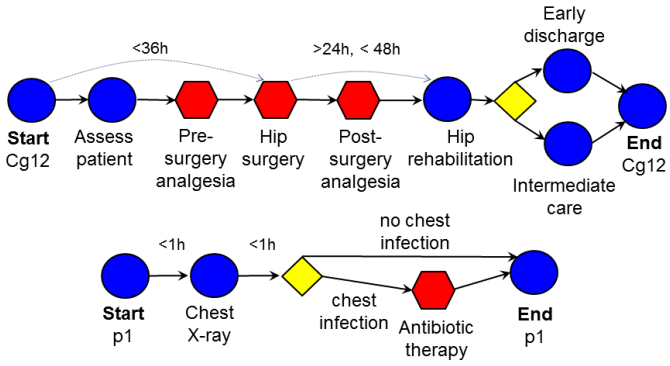

Fig. 1. Hip fracture CG (above) and chest infection plan (below). Circles are atomic actions, hexagonal nodes are composite actions, diamond nodes are decisions.

and of distinguishing between atomic and composite actions, i.e., actions that are in turn defined in the same formalism; we use the term *plan* for the description of a composite action, as opposed to the overall CG. The control flow of a CG or plan in GLARE can be specified with four different operators: *sequence*, *concurrency* (also called *group*), *alternative* (also called *decision*) and *repetition*. The *sequence* operator is for actions which have to be executed in sequential order. The *concurrency* operator between two or more (simple or composite) actions states that such actions can be executed in any order, possibly in parallel. The *alternative* operator links a decision action (diagnostic or therapeutic decision) with the possible successor actions: after the decision, one of the alternative paths has to be followed. The *repetition* operator represents iterations, stating that an action has to be repeated a given number of times, or until a given exit condition becomes true.

GLARE also provides the possibility of specifying *temporal constraints* between actions. Additionally, actions may have *preconditions*, and temporal constraints between the time when preconditions hold and the time when the related action must be executed can be specified. More details will be given in the next section.

The upper part of figure 1 is an adaptation, represented in GLARE, of the guideline for hip fracture by the British National Institute for Health and Care Excellence (NICE) [12], which will be used as example; *pre-surgery* and *post-surgery analgesia* are composite actions and are repetitions of the administration of analgesic drugs. *Hip surgery* is also a composite action, the actual surgical procedure depending on the type of fracture and other patient conditions. The CG contains information about recommended timing to ensure the effectiveness of the treatment, and to fit humanitarian criteria, while previously "hip fracture surgery was often disproportionately delayed in comparison with other operations" [12]. Since it recommends that surgery is performed "on the day of, or the day after, admission", we consider a "within 36 hours" recommendation (the delay in figure). Mobilization must start "the day after surgery", since for that case there is evidence of earlier restoration of mobility. Then, time constraints are also imposed on the starting time of hip rehabilitation (with respect to the end of surgery).

We assume that pieces of knowledge in the **BMK** are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

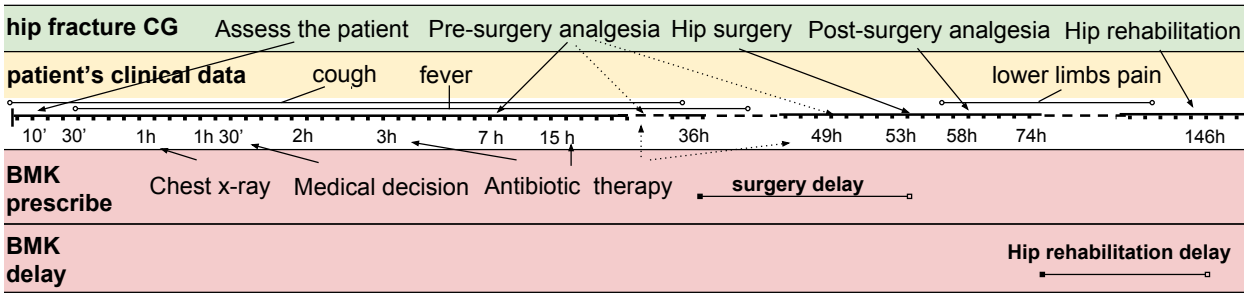

Fig. 2. Example case.

rules formed by:

- a *trigger*, i.e., a condition on the patient and context that makes the rule relevant, and either:
- a simple or composite action, which is suggested if the triggering condition holds; or:
- knowledge (formalizing counterindications) that suggests to avoid or delay some action, in one of the following forms:
  - *Avoid a*: states that action a should not be executed (we assume that such a statement is triggered by conditions that are not reversible);
  - *Delay a while c*: states that action a should not be performed as long as c holds (where we expect c to be reversible);
  - *Delay a for d*: suggests delaying action a for time d.

Knowledge involving suggested actions in our approach (**"do"** knowledge for short) is similar to *exceptions* in [3] and *guideline-independent exceptions* in [4]. Both "do" knowledge and the one that suggests avoiding or delaying an action (**"do not"** knowledge for short) refine rules proposed in [13], taking better into account the temporal dimension. The BMK may account both for actions not in the CG and for cancellation or a different timing of actions prescribed by the CG (see section 4).

The clinical case used to present our approach is the treatment of a patient which has been hospitalized to treat a hip fracture. At hospitalization time, the patient had also cough and high temperature. The following are examples of BMK rules (R1 and R2 will actually be used in our example):

R1 For patients with high body temperature and cough, the presence of a chest infection has to be investigated through a chest x-ray, and, if present, treated with an antibiotic therapy[1] (see the lower part of figure 1).

R2 The execution of physiotherapy rehabilitation of the hip has to be delayed if the patient suffers from pain in lower limbs.

R3 Actions *Calcium level measurement* and *Glucose level measurement* (which are routinely performed in all patients admitted to the ward of Italian hospitals) are always allowed, regardless of the disease.

R4 The execution of any action may be interrupted, if a problem threatening the patient's life suddenly arises. One such problem is acute heart failure; a treatment for it could be a Diuretic Therapy.

1. Actually, the recommendations in [12] explicitly mention acute chest infection as one of the conditions to be checked and, if necessary, treated, to avoid delaying surgery too much; but they also mention, without detailing them, less common concerns which may require delaying surgery: we consider chest infection as if it was one of these.

Figure 2 describes the relevant part of an example case. Actions in the first row (hip fracture CG) are directly recommended by the CG. However, several non-compliant actions (with respect to the CG) appear (fourth row, "BMK prescribe"). Actions *Chest x-ray*, *Medical decision*, *Antibiotic therapy* (occurring in the plan in the lower part of figure 1) are explained by rule R1. The fact that another problem is being treated should also explain the delay of *Hip surgery* beyond the 36 hours recommended by the CG, even though we do not expect to have an explicit model of the condition which allows surgery to proceed. Rule R2 is triggered because the day after surgery the patient has pain in the lower limbs; it explains the delay of *Hip Rehabilitation* (fifth row).

3 Execution Model

In this section we describe the model of the correct execution in time of actions (both CG and BMK actions). This model is the basis for identifying non-conformant executions.

Possible states and transitions of an atomic action are as in figure 3. The control flow of the CG execution, or triggers in the BMK, may indicate that a given action has to be considered for execution (is a candidate). A candidate action could become started or discarded; if started, it could either be completed or aborted.

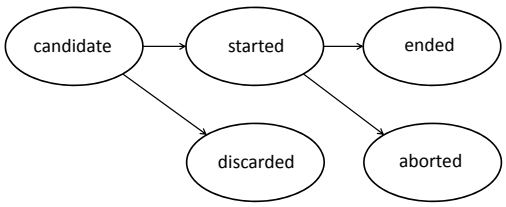

Fig. 3. States for a work action

No formal semantics has been given for the full language of GLARE, and providing it is out of the scope of this work; however, conformance needs to be defined with respect to a detailed model of execution. Therefore, in the following we describe in first order logic the semantics of the state

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

transitions of a single action, assuming they are influenced by a single source of knowledge (either the CG or the BMK), i.e., the default behavior according to such knowledge. In particular, we provide necessary or sufficient conditions for transitions to occur. We use first order logic as a specification language since it is more widely known than ASP, but, for the reasons discussed in the introduction, we limit its application to modeling the default behaviour. In section 4 we describe (informally) how the execution according to single knowledge sources can be altered when more than one source applies, and in section 7 we sketch how the overall model is represented in ASP.

We start discussing the case of an atomic action, and later we generalize to an action group (set of concurrent actions).

The control flow that makes an action a candidate, at the time t_{ca} when the previous action ends, imposes constraints on the time t_{str} when a could start.

Such constraints are provided with respect to the start/end times t_i of previous (atomic or composite) actions in the control flow, and are in the form of *bounds on differences* (b.o.d., [14]):

$$m_i \leq t_{str} - t_i \leq n_i \tag{I}$$

In addition, *preconditions* on a impose constraints of the form:

$$t + m \leq t_{str} \leq t' + n \tag{II}$$

where t and t' are the start or end time of a precondition, or, more precisely, of an episode of a precondition, i.e., a time interval for which the precondition holds. Expressions $t' + n$ in constraints (II) do not, in general, provide deadlines because the precondition could become true again[2].

There could also be constraints on the duration of action, i.e., a b.o.d. constraint $m \leq t_{end} - t_{str} \leq n$.

The *conformant* execution after an action becomes candidate (i.e., it is reached by the control flow), can be characterized as follows:

1) The action should start at a time such that the workflow constraints, and all preconditions, with their temporal constraints, enable the action, if one such time exists.
2) Otherwise, when the first deadline is reached, the action is discarded.
3) Started actions should be completed by the time allowed by the duration constraint, if no failure arises during their execution.
4) Otherwise the action is aborted at the time when the failure arises.

We represent the conditions above in logical formulas, where:

- $precond(a, \bar{t})$ means that episodes for each precondition of a hold, and the start and end times of such episodes are the tuple $\bar{t}$ of variables;
- $C_{wf}(a, t_{ca}, t, m, d)$ represents the temporal constraint on the start time t of a imposed by the workflow, i.e. the constraints on the instance of a which becomes candidate at t_{ca}, that should start at time t between m and d. The predicate C_{wf} can be defined in terms of the (variable, in the process model) times t_i and the constants m_i, n_i in the constraints (I); for a given execution, the times t_i are known and the condition can be simplified to $m \leq t \leq d$ where $m = \max_i(t_i + m_i), d = \min_i(t_i + n_i)$
- $C_{prec}(a, t, \bar{t})$ represents the constraints between episodes of preconditions (whose start and end times are in the tuple $\bar{t}$ of variables) and the time t when a could start;
- $C_{dur}(a, t_{st}, t_{end})$ represents the constraints on the action duration, i.e. constraining the start and end times t_{st}, t_{end};
- $trans, trans'$ represent times of transitions and bind different transitions for the same action; in particular, $trans(a, cand, t_{ca})$ means that a becomes candidate at t_{ca}, while $trans'(a, t_{ca}, str/disc/compl/abrt, t)$ means that the instance of a that became candidate at t_{ca} becomes started/discarded/completed/aborted at t;
- similarly, $failure(a, t_{ca}, t)$ means that for the instance of a that became candidate at t_{ca}, there was a failure at t.

The formulas below are the basis for checking compliance of a log of events with the guideline; in fact, a non-compliance corresponds to a case when one of such formulas is false, and types of non-compliance correspond to different ways such formulas can be false.

We start describing the case of an action which is not part of a group, which is simpler; for a group, the time when an action could or should start depends on other actions in the group.

The following two formulas correspond to item 1 above and provide necessary and sufficient conditions for the action to start:

$$\forall t_{ca} trans(a, cand, t_{ca}) \wedge [\exists t, \bar{t}\ precond(a, \bar{t}) \wedge C_{prec}(a, t, \bar{t}) \wedge C_{wf}(a, t_{ca}, t, m, d)] \rightarrow \exists! t_{str}\ trans'(a, t_{ca}, str, t_{str}) \tag{1a}$$

$$\forall t_{str} t_{ca} trans'(a, t_{ca}, str, t_{str}) \rightarrow C_{wf}(a, t_{ca}, t_{str}, m, d) \wedge trans(a, cand, t_{ca}) \wedge \exists \bar{t}\ precond(a, \bar{t}) \wedge C_{prec}(a, t_{str}, \bar{t}) \tag{1b}$$

Formula (1a) states that if there are times when a candidate action is allowed to start, it starts. Formula (1b) states that if the action starts, it does so at a time when it is allowed to start. The following formula corresponds to item 2 above:

$$\forall t_{ca}\ trans(a, cand, t_{ca}) \rightarrow [\nexists t, \bar{t}\ precond(a, \bar{t}) \wedge C_{prec}(a, t, \bar{t}) \wedge C_{wf}(a, t_{ca}, t, m, d) \leftrightarrow trans'(a, t_{ca}, disc, d)] \tag{2}$$

In fact, it states that the action is discarded (at time $t_{ca} + n$, i.e., at the deadline) if and only if there is no suitable time to start it, that is, one such that there are preconditions with appropriate timing and the workflow constraints allow the execution of the action.

Item 2 and formula (2) correspond to a strict interpretation of the recommendations. If some of the conditions for starting the action cannot be met, there may be valid

2. In some cases there is a deadline, e.g. for a diagnostic action that should be performed within a given time after the *first* episode of a symptom. Such deadlines can be dealt with as the other ones, but, for simplicity, we ignore them in the following.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

medical alternatives to discarding the action: either relaxing a precondition or a deadline, i.e., performing the action late, or even performing the action at a time when a precondition recommends not to. It is therefore appropriate to point out, in conformance analysis, the occurrence of case 2, and, for a non-conformance case where a deadline or a precondition is violated, to point out whether performing the action meeting *all* the recommendations was possible or not.

Formula (3) below corresponds to items 3 and 4 above. It states that the action is completed in the time required by the process specification if there are no failures in the execution (failures, and possibly their reasons, can be found in the log); otherwise, it is aborted.

$$\begin{aligned}\forall t_{ca} t_s\ trans'(a, t_{ca}, str, t_s)\ \rightarrow \\ [(\nexists t_f\ failure(a, t_{ca}, t_f) \leftrightarrow \exists t_c\ trans'(a, compl, t_c) \wedge \\ C_{dur}(a, t_s, t_c)) \wedge \\ (\forall t_f\ failure(a, t_{ca}, t_f) \leftrightarrow trans'(a, t_{ca}, abrt, t_f))] \end{aligned} \quad (3)$$

3.1 Composed actions

The formulae (1a-3) apply for atomic work actions that appear at the most abstract level of the CG and for work actions that are part of plans or repetitions. In all these cases the action is temporally constrained only with respect to the end time of the previous action; in case of a plan, the constraint on the time for starting the plan is transferred to the first action of the plan (which action should be the first is specified at design time). Similarly, for a repetition, the constraint for starting the repetition is transferred to the first instance of the repetition.

For a group of concurrent actions, the description of the conformant execution is slightly more complex. The concurrent operator, represented in GLARE with the *group* composite action, prescribes the execution of a set of actions, possibly with a set of b.o.d. constraints between their start and end times. Consider, for example, defining when actions should be discarded. In the simplest scenario, any action in the group can be executed as first action; but, in general, the temporal constraints may imply that only some of the actions can be the first one to occur. The deadline imposed on the start point of the group allows to discard the actions that can be the first ones of the group if for all such actions there are preconditions which do not allow their execution before the deadline.

In order to generalize formulae (1a,1b,2) to groups of actions, we introduce some predicates in order to define whether an action a in a group g can start at a time t; in particular we have to impose the following:

- the action preconditions must hold (as in the non-group case);
- the CG group constraints must hold, with respect to the start/end events of other actions in the group; the times for such events are fixed, in case they occurred before t, or else they are constrained to be after t;
- if no other action in the group started before t, starting a means starting the group, then the CG workflow constraints for the group must hold for the start time of a.

At a given time t, another action b in a group might be started or not, and we use $cst(b, t_{b_s}, t)$ ("constraint on start time") to mean that t_{b_s} is either the time when b started, if it started before t; or it is larger than t:

$$\begin{aligned} cst(b, t_{b_s}, t) \equiv trans'(b, t_{ca}, str, t_{b_s}) \wedge t_{b_s} \leq t \vee \\ (\nexists t' trans'(b, t_{ca}, str, t') \wedge t' \leq t \wedge t_{b_s} > t) \end{aligned}$$

therefore such a predicate represents what is known, at t, about the start time t_{b_s} of b. Similarly, a "constraint on end time" $cet(b, t_{b_c}, t)$ means that t_{b_c} is either the time when b ended, if it did before t; or it is larger than t.

For an action a in a group of n actions, we define as follows $gtc(a, \hat{t}, t)$, where $\hat{t}$ is a tuple of $2n-2$ variables, used, as the start/end times of the other $n-1$ actions in the group g, in the cst and cet formulas for such other actions:

$$gtc(a, \hat{t}, t) \equiv \bigwedge_{b \in g, b \neq a} cst(b, t_{b_s}, t) \wedge cet(b, t_{b_c}, t)$$

Then, for a time t, $gtc(a, \hat{t}, t)$ means that $\hat{t}$ are the start/end times of other actions in the group, if such events already occurred; or they are later than t. In conjunction with the b.o.d. constraints that relate the start/end times of the actions in the group, we obtain a condition for a to start at t given the start/end events that have already occurred. The GC group constraints relate $2n$ variables: t, t' for the start and end time of a, and the $\hat{t}$ variables as above. We consider the constraints $C_g(\hat{t}, t)$ on $\hat{t}$ and t only (which result from the minimal network [14] of the b.o.d. constraints).

Considering also preconditions (there should be episodes of preconditions, and C_{prec} should hold), we define:

$$\begin{aligned} cond(a, \hat{t}, \bar{t}, t) \equiv gtc(a, \hat{t}, t) \wedge C_g(\hat{t}, t) \wedge precond(a, \bar{t}) \wedge \\ \wedge C_{prec}(a, t, \bar{t}) \end{aligned}$$

Then, $cond(a, \hat{t}, \bar{t}, t)$ means that a can start at t considering its preconditions and the constraints with respect to the other start/end events in the group (the ones that already occurred, and the other ones, which are constrained to occur later than t).

Additionally, if no other action of the group started before t, we have to impose on t the CG workflow constraint on the start time of the group g, denoted as $C_{wfg}(g, t_{ca}, t)$. We define:

$gns(a, t, t_{ca}) \equiv$
$\nexists b\ b \in g \wedge a \in g \wedge trans'(b, t_{ca}, str, t') \wedge t' < t$

so that $gns(a, t, t_{ca})$ means that, for the group including a that is candidate at t_{ca}, no action started before t.

The existence of $t, \hat{t}, \bar{t}$ such that $cond(a, \hat{t}, \bar{t}, t)$ and, if $gns(a, t, t_{ca})$, also $C_{wfg}(g, t_{ca}, t)$, is a sufficient condition for a to start (at some time t_{str}, further constrained by necessary conditions):

$$\begin{aligned} \forall t_{ca} trans(a, cand, t_{ca}) \wedge [\exists t, \hat{t}, \bar{t}\ cond(a, \hat{t}, \bar{t}, t) \wedge \\ (gns(a, t, t_{ca}) \rightarrow C_{wfg}(g, t_{ca}, t))] \\ \rightarrow \exists! t_{str}\ trans'(a, t_{ca}, str, t_{str}) \end{aligned} \quad (1ag)$$

$cond$, and C_{wfg} in case of gns, are also necessary conditions for a to start at a given t_{str}:

$$\begin{aligned} \forall t_{str} t_{ca} trans'(a, t_{ca}, str, t_{str}) \rightarrow trans(a, cand, t_{ca}) \wedge \\ \wedge \exists \hat{t}, \bar{t}\ cond(a, \hat{t}, \bar{t}, t_{str}) \\ \wedge (gns(a, t_{str}, t_{ca}) \rightarrow C_{wfg}(g, t_{ca}, t_{str})) \end{aligned} \quad (1bg)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

It can be seen that in case a is not in a group, such formulas reduce to the ones (1a,1b) given before.

We now consider the conditions under which an action in a group cannot be executed according to all the recommendations provided in the guideline, and then it could, in principle, be discarded. Note that a deadline for starting one such action a, i.e., a time after which a cannot be executed respecting all the conditions, can be given by the CG workflow constraints for starting the group, if no other action in the group has started, or by a b.o.d. constraint with respect the start/end of another action in the group.

Discarding a is justified when we reach a deadline with respect to the group and/or workflow constraints, i.e., a time t_d such that they are satisfied at t_d, but not after t_d; but there is no time up to t_d when also the requirements on preconditions hold:

$$\begin{aligned}\forall t_{ca} t_d\ trans(a, cand, t_{ca}) \rightarrow \{[\exists \hat{t}\ gtc(a, \hat{t}, t_d) \wedge C_g(\hat{t}, t_d) \wedge \\ (gns(a, t_d, t_{ca}) \rightarrow C_{wfg}(g, t_{ca}, t_d))] \wedge \\ [\nexists t, \hat{t}\ t > t_d \wedge gtc(a, \hat{t}, t) \wedge C_g(\hat{t}, t) \wedge \\ (gns(a, t, t_{ca}) \rightarrow C_{wfg}(g, t_{ca}, t))] \wedge \\ [\nexists t, \hat{t}, \bar{t}\ t \leq t_d \wedge cond(a, \hat{t}, \bar{t}, t_d) \wedge \\ (gns(a, t_d, t_{ca}) \rightarrow C_{wfg}(g, t_{ca}, t_d))] \\ \leftrightarrow trans'(a, t_{ca}, disc, t_d)\}\end{aligned} \tag{2g}$$

Also in this case, it can be shown that the formula reduces to (2) when the action is not in a group.

4 Interaction of CG and BMK

In this section we identify different modalities of interaction between the CG and the BMK. We point out several situations where CG and BMK recommendations could be merged, or, in case they are contradictory, one could override the other.

When a (possibly composite) action a in the *"do"* BMK knowledge is triggered, several **modalities** of execution are possible (similarly to [4]):

- execution of a and the CG proceeds concurrently, according to their own constraints (**concur** modality);
- a and the CG are executed concurrently, but the temporal constraints are not enforced (since they are presumably given for the case where there is no concurrent treatment for another problem) (**concur_no_tc**); as special cases, a is delayed after the end of the CG execution (**after**), or a is executed first, and then the CG execution proceeds (**before**);
- the CG execution continues, and the BMK suggestion is ignored (**ignore**);
- the execution of the CG is aborted and a is executed (**abort**).

A special case occurs for the concurrent modality, in case the same action b is candidate for both CG execution and the execution of the BMK action. In such a case, b is executed only once, respecting the temporal constraints from both the CG and BMK (**cg_bmk_constr**), or from either of them (**cg_constr, bmk_constr**).

When an action a is executable or started and a *"do not"* BMK rule regarding a is triggered, the options are as follows, depending on what is triggered:

- *Avoid a*: the action a is discarded, if candidate, or aborted, if started (**avoid**); or the BMK rule is ignored (**ignore_avoid**);
- *Delay a while c*: either the BMK rule is ignored (**ignore_delay**), or $\neg c$ is used as an additional precondition for action a (**add_delay**), or it replaces preconditions for a (**delay**).
- *Delay a for d*: either the BMK rule is ignored (**ignore_delay**), or it adds the constraint $t_{now} + d \leq t_{act}$ (**add_delay**), or replaces with it the constraints on t_{act} (**delay**).

This accounts for a wide range of modifications to the set of executions allowed by a CG, taking into account knowledge, the BMK, that is, under one respect, more *general* than the CG, not being related to the class of problems addressed by the CG; under another respect, more *specific* than the CG, justifying its adaptation to a specific class of patients, which is not explicitly considered in the CG definition. It allows one of the knowledge sources to prevail over the other one (which is then ignored), or for its recommendations to be given temporal priority on the other ones. The result can only approximate the set of medically correct adaptations of a CG to a case; it is a way of making conformance analysis more flexible, without assuming exhaustivity of CGs.

5 Medical Terminology

We introduce a terminological layer to relate facts stored in the log with the knowledge sources. In fact, the actual execution could involve actions and patient conditions that could be more specific than the ones mentioned in the CG and the BMK. In particular, we have used the well developed Systematized Nomenclature of Medicine - Clinical Terms (SNOMED CT, [11]), extended with *post-coordinated* concepts (concepts defined or constrained in terms of the ones provided in advance) to specify in the BMK the triggering conditions, and use terminological reasoning to detect possible instances of such conditions in the log. For the restricted ontology language OWL 2 EL used for SNOMED CT, resoning can be integrated in ASP using the approach in [15], or, for the purposes of conformance analysis,performed in a preprocessing phase, using an OWL 2 EL reasoner. An example of definition of additional concept used in the formulation of rule R4 is:

$$LifeThreat \equiv Disease \sqcap \exists Severity.LifeThreateningSeverity$$

6 Conformance analysis

Conformance analysis exploits the execution model in section 3 in order to detect non-conformances with respect to the CG and BMK, and attempts at explaining them in terms of the execution modalities in section 4.

The analysis takes as input the description of the problem constituted by:

- the log, containing timed information on the executed actions, and the context and patient data known to be true during the execution;

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

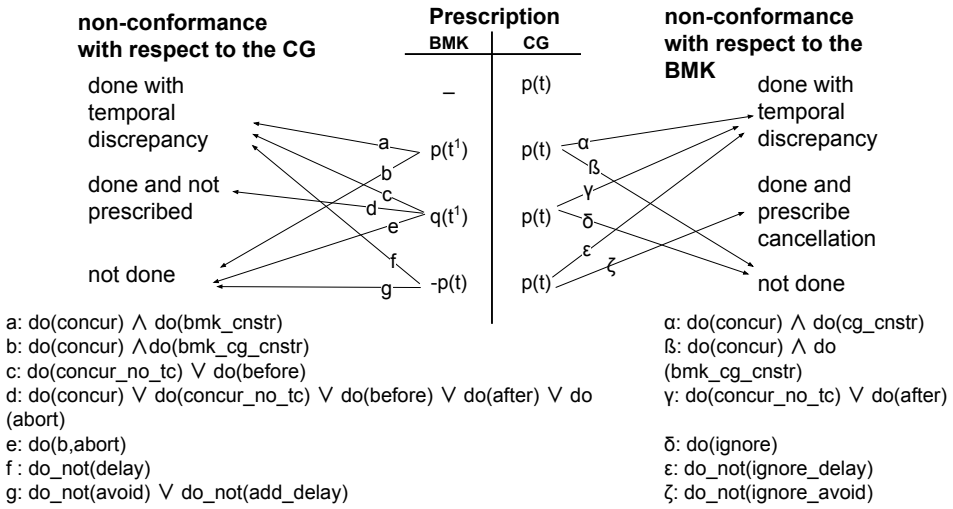

Fig. 4. Non-conformances with single knowledge sources, and their explanations.

- the specification of the CG that should have been followed for the patient, i.e., the CG for the main health problem being treated;
- the BMK specification that could suggest variations on the CG execution;
- the terminology, an extended version of SNOMED CT containing all terms used in the log trace, the BMK and CG specifications.

In order to perform conformance analysis, formulae in section 3 are the basis for analyzing conformance of a trace with respect to CG or BMK taken in isolation. In fact, it should be detected whether some of the formulae are false for some action. E.g. (1b) is violated if a should be started, but it is not. In the negation of formula (1a) we could distinguish several cases: the action is started, but either it was not candidate, or some of its precondition is never true, or for the times when the action changes state, there are no episodes of the preconditions such that the constraints hold.

The set of formulae could be further elaborated in order to interpret traces based on the interaction of CG and BMK: the actual preconditions and temporal constraints to enforce depend on the knowledge source(s) being considered and the interaction modality being hypotesized; e.g., in (1a), the executed action may fail to be a CG candidate, but be a BMK candidate. We do not, however, describe in detail such an interaction in first-order logic; rather, for the reasons discussed in the introduction, we will describe in the following how the approach is represented in ASP.

It may be the case that, at least for the execution model resulting from some CG and BMK interaction modality, the events in the log are consistent with the model, i.e., only candidate actions are executed, and their timing is consistent with the constraints. If, for a given modality, there are discrepancies with the log, they must be pointed out. However, in case a BMK rule is triggered, the alternative modalites in section 4 introduce several potential explanations, also because some the modalities are (logically) stronger than others. For example, in a "do" rule, the *concur*, *after* and *before* modalities are strictly stronger than *concur_no_tc*; similarly, *add_delay* is stronger than *delay* and *ignore_delay*. If a strong modality is consistent with the log, also a weaker one is; for a different log, a weak modality may be consistent, while a stronger modality is not, i.e., it implies some discrepancies. In general, especially in case several BMK rules are triggered, this gives rise to several potential explanations, each one with zero or more discrepancies. We introduce then a notion of preference among potential explanations, where, as a primary criterion, we prefer reconstructions with a minimum number of discrepancies with the log. Secondarily, i.e., among potential explanations with the same number of discrepancies, we prefer the ones that are conformant with more knowledge. Then:

- *concur* is preferred over *after* and *before* (even though it is not logically stronger) since *concur* means executing the CG and the BMK plan respecting all their constraints;
- *after* and *before* are preferred over *concur_no_tc*, since the former impose to respect the internal quantitative constraints of each plan, while the latter only imposes that the control flow of each plan is followed; *concur_no_tc* would be preferred over *ignore* and *abort*, but if the first one has no discrepancies, the other ones will have some, and will then already be less preferred due to the main criterion;
- The *add_delay* modality is preferred over *delay* and *ignore_delay*.

These preferences are not intended as medically preferred choices, but rather as a way to avoid presenting explanations that unnecessarily assume a deviation from recommendations in the overall (CG+BMK) knowledge. The modality is part of the explanation; this is useful especially in cases where being conformant with all knowledge is not possible, i.e., two pieces of knowledge provide contrasting recommendations, and it is not specified which one should prevail. Consider the case where, according to a BMK rule applied with the *before* modality, a CG action was actually delayed, overriding the CG constraints: in the explanation there is evidence of this choice in the *before* modality. On

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

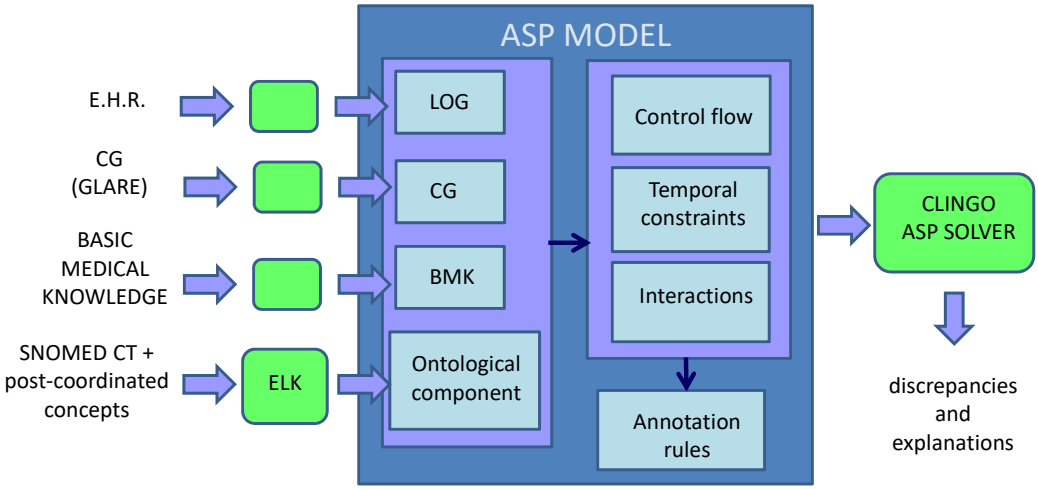

Fig. 5. General framework

the other hand, for a non-conformant action, in case a conformant execution was also possible, this is pointed out.

The following are semi-formal[3] **definitions** for conformance analysis:

- A **process state** is a time when, according to the log, some contextual or patient data changes truth value, or an action changes its state.
- A **scenario** (i.e., a potential explanation) for a given log is a choice of execution modality for each instance of BMK rules whose trigger holds for a process state.
- For a scenario, a **discrepancy** is the fact that a clause in items (1-4) in section 3, modified according to the chosen execution modalities (see section 4) does not hold for an action instance which mentioned in the log, or can be inferred to be candidate, given the log.
- Given two scenarios S_1 and S_2, S_1 is **preferred** over S_2 if it has fewer discrepancies with respect to S_2; or, they have the same number of discrepancies, and for each triggered BMK rule instance, the modality in S_2 is not preferred (according to the partial order given before) to the corresponding modality in S_1, while for at least one triggered rule instance, the modality in S_1 is preferred over the corresponding modality in S_2.
- An **explanation** is a scenario S such that no scenario is preferred to it. A **conformant explanation** is an explanation with no discrepancies.

Figure 4 summarizes an analysis which contributes to demonstrating coverage of our approach. It represents cases where a non-conformance with one knowledge source may be explained by the other source. Different lines in the central table provide different cases as regards current prescriptions from the CG and the BMK; $p(t)$ means that action p is candidate to start at time t, while $-p(t)$ means that p should not be done at t. Arrows connect a line to a type of non-conformance with respect to the CG or the BMK, and labels below provide a shorthand description of what will be part of the answer set in that case. For example, if the CG and BMK propose the same action as candidate (2nd line), it may be the case (a) that the action is performed without conforming to the CG constraints (the exception runs in concurrent modality and only the bmk constraints are enforced), or (α) it is performed without conforming to the BMK constraints, or (b and β) it is not performed because all constraints are enforced, but they are never all true.

3. Given that a complete formal definition of the semantics (in terms of possible traces) of the combination of CG and BMK has not been given.

7 Conformance framework and its ASP realization

An ASP solver can be used to perform the conformance analysis described so far. While a SAT solver finds models of a propositional representation, an ASP solver (in particular, we used Clingo [16]) finds stable models of an ASP representation, which is nonmonotonic and allows rules with variables, which are substituted in a grounding phase with a finite number of constants (in our case, for example, variables for time instants take values in a finite domain of time points). Annotation rules are used to identify cases of non-conformance with at least one source of knowledge, which include cases where formulae in section 3 are false. The nonmonotonic nature of ASP is useful to model exceptions to guideline execution according to the BMK. Optimization statements in ASP are used to represent preferences.

The general structure of the conformance analysis framework is shown in Figure 5. In this paper, terminological reasoning is performed in advance using the ELK reasoner [17] for OWL 2 EL, evaluating for each individual in the log whether it is an instance of concepts occurring in the CG and BMK specifications; the result is encoded in ASP in order to be used in the analysis (see Figure 5 where, for readability,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

the fact that the EHR, CG and BMK are also input to the preprocessor is not shown).

An ASP representation can be generated automatically (we do not provide details) from the other three knowledge and data sources of the framework.

The following main components can be identified in the framework and have a correspondent in the ASP model (see Figure 5) which is input to the Clingo ASP solver to provide discrepancies and explanations:

- the **inputs** for the analysis: the log trace, the CG and BMK model, which may use terms occurring in the terminology;
- the **ontological component** which, based on the terminology (SNOMED-CT with additional concepts), provides or contains the terminological inferences used to interpret input data;
- the **control flow component** which provides the set of allowed actions in each state (considering both the CG and the BMK);
- the **interaction component** which generates the different modalities of execution for the fired BMK rules (i.e., "concur","before", etc);
- the **temporal constraints component** which determines the allowed timing of actions with respect to other actions and action preconditions;
- the **annotation rules component** which detects non compliance, and selects BMK execution modalities which minimize non-conformances.

The ASP model reconstructs the behaviours allowed by the execution model. The sequence of actions in the log, together with patient and context data, makes it possible to identify the paths in the control flows (of the CG and the BMK plans) followed during execution. When a triggering condition of a BMK rule is satisfied by context and patient data, a candidate interpretation (answer set) is generated for each possible modality of rule execution. Preconditions and temporal constraints are evaluated taking into account the actions specification, with variations imposed by the BMK rules in the specific execution context. The annotation rules use this information to detect behaviours that differ from the expected one in each candidate interpretation.

The encoding of **inputs** is straightforward. The actions stored in the log are both the ones that have started and completed correctly, and the ones that have terminated with failure (aborted); state changes of actions are represented in ASP using the predicate `act(ActionName,ActionState,S)`, meaning that `ActionName` takes state `ActionState` at time `S`, where `ActionName` is an instance of a concept of the ontology, `ActionState` is an action state (i.e., candidate, or started, etc.), and `S` is a process state.

Clinical data about the patient and context data are represented by ground facts of the form `holds(var(Name,Value),Ts,Te)` meaning that datum `Name` has value `Value` in the interval `[Ts,Te]`. Data in the log can bind properties to measures (e.g. systolic blood pressure = 124) or represent some other kind of information (e.g., the patient has a hip fracture). The sequence of relevant process states is reconstructed in facts `state(S)`, that hold for each endpoint of the intervals for data values, and for each time point when an action changes state.

Since GLARE provides a construct to represent decisions in the clinical treatment, which are associated to xor-split points of the control flow, it is assumed to have explicit trace in the log about the decision undertaken, which is represented in the ASP encoding.

The GLARE representation of CG and BMK actions is encoded with ground facts which describe the control flow, preconditions and time constraints. For example, `succ(Src,A1,A2)` encodes the successor operator and states that, in knowledge source `Src` (either the guideline, or the identifier of a BMK rule), action `A2` is the successor of `A1`. As regards preconditions, `precond(Src,A,Cond,start/end,M,N)` states that `Cond` is the name of a precondition (other rules will state when it is true) for action A, and the time A_s when A starts should be such that $M \leq A_s - Cond_{start/end} \leq N$. Bounds on difference constraints between successive actions are represented in a similar way, as well as b.o.d. constraints on start/end of actions in a group. We assume that temporal constraints between actions of a group are expressed as a minimal network [14], where all the implied constraints are made explicit; and we assume that the constraints are consistent (which is automatically checked if the minimal network is computed).

BMK trigger rules can be mapped to a set of ASP rules having the trigger condition as body and one of the following as head:

- `prescribe(ID,A,T)` for a BMK rule `ID` prescribing the composite action `A` at time `T`;
- `prescribeAvoid(ID,A,T)` for a BMK rule `ID` prescribing the discard/abort of an atomic action `A` at time `T`;
- `prescribeDelayWhileC(ID,C,A,T)` for a BMK rule `ID` prescribing at time `T` the delay of `A` until the condition with name `C` becomes true.
- `prescribeDelay(ID,C,D,A,T)` for a BMK rule `ID` prescribing at time `T` the delay of `A` for time `D` given that the condition with name `C` is true.

Actions prescribed by the BMK can be either atomic or composite actions and are defined in the GLARE language, like the CG. The conditions associated with a given condition name (in rules prescribing delay) are specified with further ASP rules, using actual conditions on data. Actions and conditions are expressed using concepts of the ontology; this makes it possible to express in a BMK rule a general condition involving a class of situations.

For example, rule R2 in section 2 is encoded in ASP as:

```
prescribeDelayWhileC(r2,condR2,
  hip_rehabilitation,S):-
    holds(var(X,true),S1,S2), state(S),
    S>=S1, S<=S2, isa(X,hip_pain).
```

The rule uses the predicate `isa` which corresponds (see below) to membership to a concept; a predicate (`inf_subClass`) representing subsumption between two concepts can also be used.

The **ontological component** provides instances of `isa` and `inf_subClass`. As mentioned before, in this paper we describe the option where such inferences are performed in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

a preprocessing phase in Java using the ELK reasoner [17], a DL reasoner developed to provide high performance reasoning support for OWL 2 EL, whose underlying logic is the low complexity description logic $\mathcal{EL}^{++}$, whose expressivity is sufficent for SNOMED CT [18] (see [19] for a discussion on the expressivity needed for the medical domain). For instance checking, preprocessing involves the set of assertions in the log and the set S_1 of concepts in the terminology that occur in the CG or in the BMK. The assertions are introduced in the ontology; for each concept in S_1, its instances are retrieved and facts `isa(instance,concept)` are generated. Subsumption checking involves the set A of concepts that are CG or BMK actions and the set S_2 of concepts appearing in the triggering conditions of BMK rules as second argument of predicate `inf_subClass` or as second argument in the head of a prescribeAvoid rule. For each `Action` in S_2, its subsumed concepts are retrieved, and for each `Concept` in the result which belongs to A, a fact `inf_subClass(Concept,Action)` is generated.

The **control flow component** evaluates the control flow of CG and BMK plans. Similarly to [13], given the executed action and the control flow description, instances of `candidate(Src,A,Tca,T)` are inferred, which state that the knowledge source `Src` (either `cg`, or the BMK rule identifier) enables the execution of `A` in `T` (the time `Tca` when it became candidate identifies an action instance). An action remains candidate after is has become such; and, for example, if action `A` is a successor, in the control flow, of action `A'`, it becomes candidate at the time `T'` when `A'` ends. It ceases to be candidate when it starts, or in case (see below) it is disabled.

The **interaction component** determines the execution modality of BMK rules and their consequences. ASP *choice rules* to generate candidate answer sets are provided for each execution modality of a triggered BMK rule; in particular, they have the form `1 { ... } 1 :- Body` , meaning that, if `Body` holds, (at least and at most) one of the atoms in brackets is inferred to hold in each candidate answer set. Using ASP optimization statements (see below), the answer sets with fewer discrepancies will be selected.

When a rule prescribing a new action is triggered (`prescribe(ID,A,S)`) (and it was not in the previous state) one of six different execution modalities is chosen, i.e., a choice rule infers an instance of `start_excpt_do(S,ID,A,MOD)`, with `MOD` equal to one of: `ignore`, `concur`, `before`, `after`, `abort`, `concur_no_tc`.

The appropriate behaviors for each modality are modeled in rules which have the appropriate instances of `start_excpt_do(S,ID,A,MOD)` in the body, and introduce or cancel actions, or imply variations of the time constraints.

Of course, for the `ignore` modality nothing has to be changed. For the `before` modality, the prescribed action is enabled, and other actions are disabled; they will be enabled again once the BMK action ends, but with the additional information (for the temporal constraints component) that temporal constraints on the start of the action should not be checked. The same device is used, for the modality `concur_no_tc` where the BMK action is enabled, to avoid checking temporal constraint for start and end times of all enabled actions. For the `concur` modality, the BMK action is enabled, and the special case where the action is also prescribed, in the same state, by the CG, should be dealt with: a choice rule selects one of the 3 modalities, i.e., imposing constraints from the CG only, from the BMK only, or both; for the first 2 modalities, appropriate information is again generated for the temporal constraints component. Dealing with the `after` modality involves enabling the BMK action after the (unique) end action of the CG has been executed. For the `abort` modality, all other candidate actions are disabled, and running actions should be aborted.

BMK rules prescribing to avoid or delay actions are dealt with in a similar way. A rule prescribing to *avoid a* disables the action a, if it is candidate; if running, it should be aborted.

For a rule prescribing to *delay a for d*, remember that we consider 3 modalities: either ignoring this delay, or replacing the CG constraints with the constraint that the action should start at least d after the current time, or adding this to the CG constraints. Again, an ASP choice rule is used to choose a modality, and appropriate information is inferred, for each of them, for the temporal constraint component.

For a rule prescribing to *delay a while c*, we also consider 3 similar modalities, using a choice rule, and provide the appropriate information for the temporal constraint component.

The **temporal constraints component** (described in more detail in Appendix B) reconstructs the expected timing of actions. Considering the CG specification, such a timing depends on the temporal constraints imposed with respect to the control flow (i.e., other actions) and the temporal constraints imposed by the preconditions. As we have seen, BMK rules can introduce other temporal constraints or override the control flow temporal constraints. An action is executable in a time interval if both workflow and precondition constraints are satisfied, otherwise the action should be discarded. The control flow constraints determine a time interval (which could be the intersection of intervals provided by single b.o.d. constraints); we verify whether it contains some time when all the preconditions (with the respective time constraints) allow the execution of the action. This simplifies the analysis, given that preconditions may hold for more than one interval; moreover, keeping the two pieces of information separate allows to point out, with the annotation rules (below), whether the non-conformances are violations of workflow constraints or of preconditions constraints, and whether it was possible to respect both groups of constraints.

A predicate `tc(Src,Tca,A,started/completed,Ts,Te)` is used to specify the allowed interval `[Ts,Te]` for starting or completing action `A` prescribed by knowledge source `Src` at `Tca`. As described before, BMK rules can cancel or change the time constraints imposed by the process specification; such variations are represented through the predicate `exception(Src,A,T,ExceptionType)`, where the first three arguments identify a unique instance of action (the one candidate by `Src` at `T`), the last one specifies the kind of variation (constraints should not be considered, or overridden by constraints given by a BMK rule). Predicate `tc` is defined through other predicates; one of them is `tc_00`, which applies to actions that have no concurrent action (the case of actions in a group is of course more complex, we do not provide details); the following:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering


```
tc_00(Src,Tca,A,started,Tca+Min,Tca+Max):-
     wf_tc(Src,A,started,B,completed,Min,Max),
     candidateS(Src,A,Tca), end(Src,_,Bi,Tca),
     isa(Bi,B),
     not exception(Src,A,Tca,blockEvaStartTc).
```

is one of the rules for `tc_00`, which provides the information that action `A` should start between `Tca+Min` and `Tca+Max`, given that: `wf_tc(..)` is the representation of the constraint $Min \leq A_s - B_e \leq Max$, on the start time where A_s of `A` and the endtime B_e of `B`; an instance `Bi` of `B` ended at `Tca`; and we have no exception stating that constraint checking on the start of `A` should be suppressed (as in case a BMK action is hypothesized to be executed in the `concur_no_tc` modality).

The **annotation rules component** consists of rules which identify discrepancies between the log trace and the different behaviours considered in the answer sets. The discrepancies are violations of the execution model described in section 3, and annotation rules correspond to different way of falsifying some of the formulae in section 3.

As an example, the rules:

```
 info(act_executed_not_candidate,A,Tact):-
     act(A,started,Tact),
     not subsumer_candidate(A,Tact).
     not suppressed(Tact).
info(act_start_late_wrt_CF,A,Tca,Tact,TcS,TcE):-
     candidateWindow(Src,A,Tca,Tact),
     trans(Ai,started,Tact), isa(Ai,A),
     tc(Src,Tca,A,started,TcS,TcE),Tact>TcE,
     not suppressed(Tact).
```

detect two violations of formula (1b) in section 3. The first is the case where an action actually started, but it was not candidate (actually, none of its subsumers was candidate). The second is the case where the action started too late: temporal constraints provided the interval `[TcS,TcE]` for starting `A`, and an instance `Ai` actually started, but at a time `Tact>TcE`. The condition `not suppressed(Tact)` is because conformance checking is suppressed after the time when it is inferred that a candidate action cannot be executed following all the recommendations in the guideline. As discussed in section 3, in that case actually discarding it is not necessarily the best medical choice; moreover if it is actually discarded, there is no general solution to state if and where the execution of the guideline should restart; in an interactive system, a physician could decide where, possibly with support from a system; in an off-line analysis, which is what is supported by the approach in the paper, the more general solution is to remain neutral and suppress conformance analysis from that time on.

ASP optimization statements are used to select, with highest priority, answer sets with a smallest number of instances of the `info` predicates, i.e., answer sets with a smallest number of discrepancies. Secondarily, conformance with more knowledge is preferred. This is obtained, for predicates that are used for choosing execution modalities of BMK rules, maximizing (with smaller priority with respect to the previous criterion) the number of instances with `cg_bmk_cnstr`, `addDelay`, `concur`, and, with smaller priority, `after` and `before`.

8 Example

In figure 2 we showed an example log where the first and fourth row contain executed actions and the second one contains patient data. Actions in the first row are recommended by the CG, but several actions do not comply with the CG: *Hip surgery* and *Hip rehabilitation* are delayed, while *Chest x-ray*, *Medical decision*, *Antibiotic therapy* are not in the CG.

Our approach finds an explanation for the three extra actions, applying the BMK rule R1 triggered by the presence of cough and fever in the log. The identified modality is **concur_no_tc**, which also explains the delay of *Hip surgery* beyond the 36 hours recommended by the CG (this delay can be medically justified by the need to have a sufficient antibiotic coverage for chest infection, but this would require modeling an explicit precondition). The delay of *Hip rehabilitation*, required by the CG to be executed one day after *Hip surgery* and started after more than two days, is explained by the BMK rule R2 which prescribes to delay hip rehabilitation if the patient is has pain in the lower limbs.

Let us see in more detail how this last explanation of the CG execution is retrieved by the ASP model. At time 58h, when *Post-surgery analgesia* is completed, the ASP rule modeling "sequence" makes `candidate` true for *Hip rehabilitation*. Log facts asserting that the patient suffers from *lower limbs pain* between times 55h and 143h make `prescribeDelayWhileC` true for *hip rehabilitation* for the same times, given R2. Given that both `prescribeDelayWhileC` and `candidate` are true for *Hip rehabilitation* at time 58h, the ASP choice rule for this case generates a candidate answer set for each of the three modalities of the BMK delay rule, with an instance of `start_excpt_do_not` having as a last argument one of the values `ignoreDelay`, `delay` and `addDelay`. Then:

1) In the scenario where the last parameter is `ignoreDelay`, the rules for predicate `tc`, and for the starting time of *hip rehabilitation*, are evaluated based only on the constraints in the CG. Given that the time when *hip rehabilitation* is registered is after the one specified by `tc`, the ASP rule annotating, with predicate `info`, that the action started too late, is triggered, i.e., there is a discrepancy.
2) In the scenario where the last parameter is `delay`, predicate `tc` is made true with the last arguments set to 143h and ∞, given that the action is allowed to be executed once the patient does not suffer from *lower limb pain* any more. In this candidate answer set no other instance of `tc` is true for the action *hip rehabilitation*, because predicate `exception` blocks the evaluation of the CG temporal constraints. Since rehabilitation starts at 146h, no discrepancy is detected.
3) In the scenario where the last parameter is `add_delay`, `tc` is true for both the times imposed by the CG and the ones imposed by the BMK; the two constraints are inconsistent (and the first one is violated), then the action should have been discarded, and there is a discrepancy.

Given that the optimization rules select, as best explanation of the execution, the one with the smallest number of discrepancies, the second scenario is selected, explaining

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

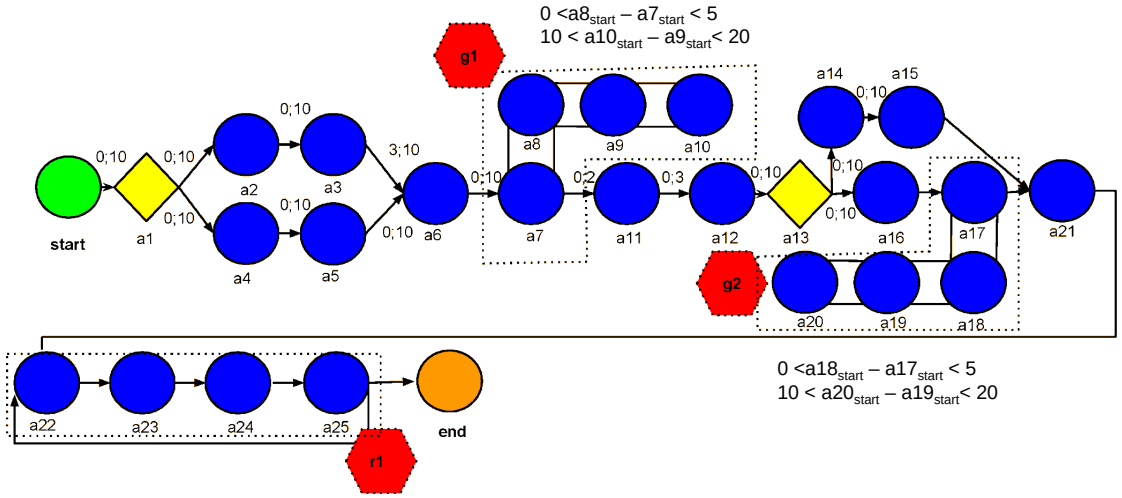

Fig. 6. CG used for evaluation

BMK/CG	1x	seq2x	par2x	seq3x	par3x	seq4x	par4x
0	0.2	0.78	0.50	1.67	0.78	2.98	1.08
40	2.88	5.42	7.76	7.87	11.36	10.69	14.53
80	17.90	27.54	46.78	39.35	69.54	44.80	73.27
120	49.58	74.99	129.39	93.90	179.49	113.23	247.15

TABLE 1
Clingo computing time

BMK/CG	1x	2x	3x	4x
0	45.36	52.82	59.64	59.65
40	55.21	60.36	65.50	70.73
80	63.15	65.68	71.42	79.15
120	68.95	77.044	82.33	87.52

TABLE 2
ELK preprocessing time

the log with a strict application of the BMK rule requiring to delay *Hip rehabilitation*. Based on the facts belonging to the selected answer set, the actual explanation presented to the user would point out that there are no discrepancies (no `info` facts), provided that rule R2 was applied, and the recommendation provided by the rule, to delay rehabilitation, was strictly followed, overriding the CG recommendation (since this is the meaning of the `delay` modality).

As a further example, in case, with the same timing for hip rehabilitation, R2 was not in the knowledge base, or *lower limb pain* was not holding, there would be a discrepancy (as in item 1 above), and the user would be presented with the fact (from the corresponding `info` fact in the answer set) that hip rehabilitation started late with respect to the recommendation in the guideline.

9 Evaluation

In order to explore the feasibility and scalability of our approach we synthesized CG and BMK models containing all the constructs offered by the two languages; increasingly larger models were considered, replicating the same structure several times. Both the CG and BMK mention concepts from SNOMED CT.

Figure 6 shows the process used as the main building block for the CG. It contains 25 atomic actions (circles), 2 decision splits (diamonds), 2 groups (g1 and g2) and a repetition (r1). Workflow temporal constraints are associated with each action; for non-concurrent actions, the b.o.d. between the end of an action and the start of the following one are given on the arrows. Constraints on groups are near the composite action constituting them. Temporal constraints on preconditions, not shown in the figure, are provided for 10 actions of the process. From this structure, further CGs versions have been generated adding copies, in sequence or parallel, of the same process with renamed actions. The largest examples, with 4 copies, in sequence or in parallel, contain 100 atomic actions.

The basic version of the BMK used in the test has a total of 40 rules; 10 of them prescribe the introduction of a new action (complex actions made of 5 atomic actions); 20 rules prescribe a delay of a CG action (equally divided between the two forms of delay); 10 rules prescribe avoiding actions. We tested the evaluation of compliance with respect to the CG without BMK, and with three different BMKs with 40, 80 and 120 rules respectively. We generated three different log traces for each version of the CG, one where the execution is compliant with the CG only, one where the execution is compliant with the BMK only, and one which is compliant with both knowledge sources; we will only show results for evaluating traces which followed both prescriptions given by the CG and BMK, results for the other cases are similar.

Tables 1 and 2 report the most relevant results of the test, where the Clingo ASP solver (version 3.0.3, 64 bit) [20] and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2734084, IEEE Transactions on Knowledge and Data Engineering

the ELK reasoner [17] were used on a machine with Intel Xeon E5520 processors (2.26Ghz) and 32 GB of RAM. Table 1 shows the running time in seconds for the ASP models where terminological reasoning was executed a-priori in ELK, with running times in table 2.

In table 1, columns correspond to different repetitions, in number and form (sequence or parallel), of the CG structure; rows correspond to a different number of rules in the BMK. Running times in seconds for the ASP solver are shown (most of the time is actually taken by *grounding*, where rules with variables are instantiated). The presence of (precomputed) subsumptions does not significantly increase running time; conformance checking without use of terminological inference (as in [21]) requires almost the same time (we do not report data). Computation time is mainly influenced by the size of the BMK. It is important to underline that most of the BMK rules in the experiment (about 80%) were triggered in the log, and this implies a high number of answer sets generated by the solver.

The preprocessor calls ELK for instance and subsumption checking. The loading step of the ontology and the ELK preprocessing requires a significant time, about 40 seconds, while each query (both for subsumption evaluation and instance checking) is executed in about 0.002/0.005 seconds; table 2 provides the overall times needed for terminological reasoning in the different cases.

The results show that conformance analysis with respect to the CG only scales up well; the analysis is feasible also in case a small number of BMK rules applies in each specific case in a log. In general, both reasoning on a large set of BMK rules firing for each specific case, and terminological reasoning, can be considered bottlenecks for this approach to conformance evaluation.

10 Conclusions and Related Work

CGs do not include all knowledge that physicians have to take into account when treating patients, since patient states and contexts of execution cannot always be foreseen in the definition of a guideline. In this paper we propose an approach for analyzing temporal conformance of execution traces with respect to a richer form of medical knowledge, which may be used to explain deviations from a strict application of the guideline, both as regards extra actions for situations that are not foreseen (and whose treatment may alter the timing of guideline execution), and cancellation or delay of actions that are prescribed by the guideline, when there are reasons to do so. Given that we do not assume that all exceptions and interactions are modeled, interpretations provided by the approach can only be an approximation of medically correct executions. For example, the approach assumes that a temporally non-conformant execution of a guideline for treating a patient condition is always potentially explained in case another treatment, triggered by the BMK, is being executed.

Several approaches in the literature have addressed some of the issues in our proposal, or related ones. One of such issues is the verification of properties of clinical guidelines, i.e., in order to improve quality of guidelines, proving that some properties, expressed in a temporal logic, hold for all executions of a guideline, whose model is expressed in a formal language. Theorem proving techniques have been adopted for verification in the Protocure and Protocure II projects [22], [25], while model checking techniques have been explored in Protocure II and GLARE [23], [24].

The integration of CGs with general medical knowledge has been considered in some case (see e.g. [26]) using such knowledge as a source of definitions of clinical terms and abstractions. In Medintel [27] different medical information sources (e.g., guidelines, reference texts, scientific literature) are used to improve decision support and the quality of care provided by general practitioners. The approach in [28] considers different forms of general medical knowledge in the context of CG verification: knowledge on physiological mechanisms, and knowledge concerning good practice in treatment selection (leading to quality requirements), assuming the availability of a preference relation between treatments. Verification is used in order to check whether, given a class of patients, the CG achieves a set of intentions, satisfying the quality requirements.

A limited number of approaches have dealt with verifying conformance of a trace of actions with recommendations in a CG. In [6], differences between actual actions and CG prescriptions are detected and analyzed, e.g. by comparing, for a non-compliant actions, actual findings with findings that support the action according to the CG. However, neither general medical knowledge nor quantitative time are considered in such a work. On the other hand, [2] focuses on the interaction between clinical guidelines (CGs) and the basic medical knowledge (BMK) in the light of the conformance problem. Our approach presents several similarities to it, but it is based on ASP, and, more importantly, it is more flexible as it considers different modalities of CG-BMK interaction. Also, the approach in this paper is an in-depth extension of earlier papers [13], [21]; here, we address *together* the temporal dimension and terminological reasoning, and we provide a more detailed logical model and an experimental evaluation of the approach.

The general problem of conformance analysis of event log with process models has been dealt with in [29], [30]. In particular, in [29] metrics are introduced to measure the degree of correspondence between model and traces.

Finally, our proposal is related to work about the treatment of CG *exceptions* [3]–[5], i.e., conditions that may suddenly arise, and whose treatment is not directly foreseen in the CGs. Indeed, part of the BMK we consider in our approach regards such conditions, and some of the modalities for CG-BMK interactions have already been identified in the context of exception handling.

In summary, the main advances of the approach in this paper with respect to the state of the art are:

- complementing conformance analysis with an *explanation* of how potentially constrasting knowledge sources (CG and BMK) have been applied to a specific case;
- the *temporal* analysis of conformance and explanation.

Acknowledgement

The authors are grateful to Gianpaolo Molino for his valuable advice on the medical issues in the paper.

REFERENCES

[1] M. Field and K. Lohr, Eds., *Guidelines for clinical practice: from development to use*. Institute of Medicine, Washington, D.C: National Academy Press, 1992.

[2] A. Bottrighi, F. Chesani, P. Mello, M. Montali, S. Montani, and P. Terenziani, "Conformance checking of executed clinical guidelines in presence of basic medical knowledge," in *Business Process Management Workshops*, 2011, pp. 200–211.

[3] A. Grando, M. Peleg, and D. Glasspool, "A goal-oriented framework for specifying clinical guidelines and handling medical errors," *Journal of Biomedical Informatics*, vol. 43, no. 2, pp. 287–299, 2010.

[4] G. Leonardi, A. Bottrighi, G. Galliani, P. Terenziani, A. Messina, and F. Della Corte, "Exceptions handling within GLARE clinical guideline framework," in *AMIA 2012, American Medical Informatics Association Annual Symposium*, 2012.

[5] S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso, and S. Panzarasa, "Flexible guideline-based patient careflow systems," *Artif. Intell. in Medicine*, vol. 22, no. 1, pp. 65–80, 2001.

[6] P. Groot, A. Hommersom, P. J. F. Lucas, R. Merk, A. ten Teije, F. van Harmelen, and R. Serban, "Using model checking for critiquing based on clinical guidelines," *Artif. Intell. in Medicine*, vol. 46, no. 1, pp. 19–36, 2009.

[7] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub, *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012.

[8] J. McCarthy, "Elaboration tolerance," in *Commonsense 1998. Revised version available at http://jmc.stanford.edu/articles/elaboration.html*, 1998.

[9] P. Terenziani, G. Molino, and M. Torchio, "A modular approach for representing and executing clinical guidelines," *Artif. Intell. in Medicine*, vol. 23, no. 3, pp. 249–276, 2001.

[10] L. Anselma, P. Terenziani, S. Montani, and A. Bottrighi, "Towards a comprehensive treatment of repetitions, periodicity and temporal constraints in clinical guidelines," *Artif. Intell. in Medicine*, vol. 38, no. 2, pp. 171–195, 2006.

[11] International Health Terminology Standards Development Organization. SNOMED CT. http://www.ihtsdo.org/snomed-ct/.

[12] British National Institute for Health and Care Excellence. Hip fracture: The management of hip fracture in adults, http://www.nice.org.uk/guidance/cg124.

[13] M. Spiotta, A. Bottrighi, L. Giordano, and D. Theseider Dupré, "Conformance analysis of the execution of clinical guidelines with basic medical knowledge and clinical terminology," in *Knowledge Representation for Health Care - 6th International Workshop, KR4HC 2014, LNCS 8903*, 2014, pp. 62–77.

[14] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artif. Intell.*, vol. 49, no. 1-3, pp. 61–95, 1991.

[15] M. Krötzsch, "Efficient inferencing for OWL EL," in *Proc. JELIA 2010*, 2010, pp. 234–246.

[16] The Potsdam Answer Set Solving Collection, http://potassco.sourceforge.net/.

[17] ELK. https://code.google.com/p/elk-reasoner/.

[18] K. Dentler, R. Cornet, A. ten Teije, and N. de Keizer, "Comparison of reasoners for large ontologies in the OWL 2 EL profile," *Semantic Web*, vol. 2, no. 2, pp. 71–87, 2011.

[19] A. L. Rector, "Medical informatics," in *Description Logic Handbook*, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., 2007.

[20] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider, "Potassco: The Potsdam answer set solving collection," *AI Comm.*, vol. 24, no. 2, pp. 105–124, 2011.

[21] M. Spiotta, P. Terenziani, and D. T. Dupré, "Answer set programming for temporal conformance analysis of clinical guidelines execution," in *Knowledge Representation for Health Care - 7th International Workshop, KR4HC/Prohealth 2015, LNCS 9485*, 2015.

[22] A. ten Teije, M. Marcos, M. Balser, J. van Croonenborg, C. Duelli, F. van Harmelen, P. J. F. Lucas, S. Miksch, W. Reif, K. Rosenbrand, and A. Seyfang, "Improving medical protocols by formal methods," *Artif. Intell. in Medicine*, vol. 36, no. 3, pp. 193–209, 2006.

[23] S. Bäumler, M. Balser, A. Dunets, W. Reif, and J. Schmitt, "Verification of medical guidelines by model checking - A case study," in *13th International SPIN Workshop*, 2006, pp. 219–233.

[24] A. Bottrighi, L. Giordano, G. Molino, S. Montani, P. Terenziani, and M. Torchio, "Adopting model checking techniques for clinical guidelines verification," *Artif. Intell. in Medicine*, vol. 48, no. 1, pp. 1–19, 2010.

[25] A. Hommersom, P. J. F. Lucas, and P. van Bommel, "Checking the quality of clinical guidelines using automated reasoning tools," *TPLP*, vol. 8, no. 5-6, pp. 611–641, 2008.

[26] A. ten Teije, S. Miksch, and P. Lucas, Eds., *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends*, ser. Studies in Health Technology and Informatics, vol. 139. Amsterdam: IOS Press, 2008.

[27] C. J. Brandhorst, D. Sent, R. A. Stegwee, and B. M. A. G. van Dijk, "Medintel: Decision support for general practitioners: A case study," in *MIE 2009*, 2009, pp. 688–692.

[28] A. Hommersom, P. Groot, P. J. F. Lucas, M. Balser, and J. Schmitt, "Verification of medical guidelines using background knowledge in task networks," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 6, pp. 832–846, 2007.

[29] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[30] R. Mans, W. M. P. van der Aalst, and R. J. B. Vanwersch, *Process Mining in Healthcare - Evaluating and Exploiting Operational Healthcare Processes*, ser. Springer Briefs in Business Process Management. Springer, 2015.

Matteo Spiotta received the Master Degree in Computer Science in 2011 from Università del Piemonte Orientale and the PhD in computer science in 2016 from Università di Torino. His research activity is centered on the verification and conformance analysis of Business and Clinical Processes in Answer Set Programming, combined with Constraint Reasoning and Terminological Reasoning.

Paolo Terenziani received the PhD in computer science in 1993 from Università di Torino and Università di Milano. Since 2000 he is Full Professor at DISIT, Università del Piemonte Orientale, Italy. He is also Adjunct Professor at the Griffith University, Brisbane, Australia. His research activity covers different areas of Computer Science. In Artificial Intelligence, his activity concerned Knowledge Representation, with specific attention to the fields of temporal reasoning (constraint propagation algorithms, treatment of periodicity) and diagnosis. In the field of DataBases, he mainly focused on the extension of "standard" relational models and algebrae to deal with time-related phenomena, and with the semantics of temporal databases. In Medical Informatics, since 1997 he was involved with Azienda Ospedaliera S. Giovanni Battista in Turin (the third hospital in Italy) in a long-term project for the development of GLARE, a semi-automatic manager of clinical guidelines. Paolo Terenziani published more than 150 papers about these topics in refereed international journals and conference proceedings. In 1998 he received the annual "Artificial Intelligence Prize" from the Italian Association for Artificial Intelligence.

Daniele Theseider Dupré received the PhD in computer science (1994) from Università di Torino. He is associate professor at DISIT, Università del Piemonte Orientale, Italy. His research interests are in the area of Knowledge Representation and Reasoning and include Abductive and Diagnostic Reasoning, Temporal Reasoning, Constraint Reasoning, Nonmonotonic Reasoning, Terminological Reasoning, Answer Set Programming, and their use in verification and analysis of Business and Clinical Processes. He published more than 80 papers in refereed journals and conference proceedings.