

Compliance Checking of Cancer-Screening CareFlows: an Approach based on Computational Logic

Federico CHESANI^a, Evelina LAMMA^b, Paola MELLO^a, Marco MONTALI^a,
Sergio STORARI^{b,1}, Paola BALDAZZI^c, and Marilena MANFREDI^c

^a *DEIS, University of Bologna, Italy*

^b *ENDIF, University of Ferrara, Italy*

^c *Department of Public Health - Sanitary Agency of Bologna, Italy*

Abstract. Clinical guidelines and Careflow systems have been recently identified as a means to improve and standardize health care services. A number of ICT-based management solutions have been proposed, focussing on several aspects such as specification, process logs verification with respect to specification (compliance), enactment and administration of careflows.

In this paper we introduce the GPROVE framework, based on Computational Logic, and focused on the (formal) specification of careflows and on the compliance verification of the process executions w.r.t. the specified models. In particular, we show its application to the Cancer Screening Guideline used by the sanitary organization of the Emilia Romagna region, discussing its formalization in GPROVE and the results of the compliance checking applied to logs of the screening process.

Keywords. Careflow Monitoring System, Clinical Practice Guidelines, Compliance Verification, Computational Logic.

Introduction

Clinical decisions in modern health care organizations are progressively taken on evidence-based care [1]. To this end, the use of clinical practice guidelines is considered to be a fundamental step towards high quality and standardized health services. As reported in [2], clinical guidelines describe the activities of a medical team in a comprehensive manner, with the purpose of defining best practices for patient management. In particular, clinical guidelines describe also the behavioural aspects of medical work, i.e., the clinician's workflow (namely *careflow*). Careflows are individual case-based and involve the coordinated execution of multiple medical tasks performed by different health care subjects on a specific patient.

The adoption of computer-based guidelines and Careflow Management Systems (CfMS) is an important issue, especially when the health care services are provided through complex care processes which involve several health care professionals. The

¹Corresponding Author: Sergio Storari, Dipartimento di Ingegneria, Università di Ferrara, via Saragat 1, 44100 – Ferrara, Italy; E-mail: sergio.storari@unife.it.

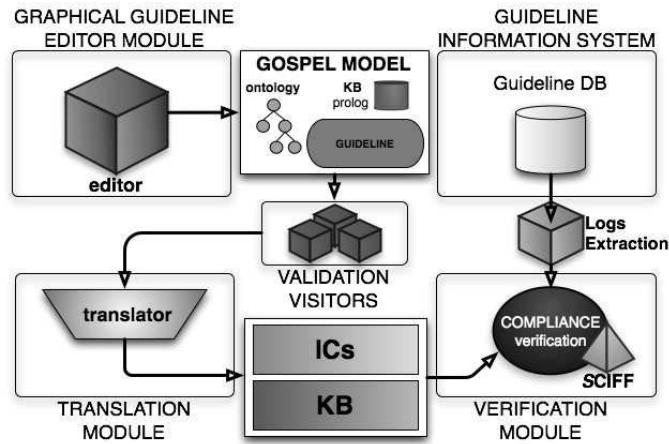


Figure 1. Architecture and modules of the GPROVE framework

goal of a CfMS is to “handle patients”, in the sense of executing medical tasks in a specific order, as effectively and efficiently as possible. This management is made through the automated coordination, control, and distribution of tasks required to satisfy a given careflow process. A CfMS then is a set of tools which support careflow design (including its formal representation), enactment, administration, and monitoring.













In this work, we describe the Guideline PROcess cOnformance VERification framework (GPROVE, [3]), and its application to the Cancer Screening Guidelines adopted by the sanitary organization of the Emilia Romagna region (Italy). GPROVE is a set of tools for the specification and a-posteriori verification of the careflow process executions, comprehending: a graphical process definition language (GOSpeL); an automatic mapping/translation towards a formal language (SCIFF); an operational counterpart (a proof procedure) of the SCIFF formalism, that is used to verify the compliance of a given execution trace w.r.t. the defined careflow process. The verification step can be used in order to identify possibly undesired behaviours, as well as to analyze and comprehend features and characteristics of the real process that are not properly represented in the modeled process: if this is the case, log traces provide a feedback to the model, possibly leading to an improvement of the model itself.

1. Careflow Compliance Checking: the GPROVE Framework

The GPROVE framework [3] allows the user to specify a guideline and then to reason about the compliance of the observed behaviour. From the architectural viewpoint GPROVE is composed of several modules, as shown in Fig. 1.

The user specifies the medical guideline by means of GOSpeL (Guideline prOcess Specification Language, [3]), a graphical language inspired by flow charts that allows the representation of the *activities* and the *flows* of activity executions. The GOSpeL representation of the guideline is then translated into a formal specification based on the SCIFF language [4]. Such translation is automatically performed by means of an algorithm [5], that “explores” the GOSpeL model and generates a set of *rules* (Integrity Constraints, *ICs* for short, and a knowledge base, SOKB for short) representing how

Table 1. GOSpeL graphical elements

family	type	notation	description
Activities	atomic activity		single atomic unit of work within the guideline
	complex activity		Non-atomic unit of work. It encapsulates a new (sub)process definition.
	iteration		For-like cyclical complex activity
	while		While-like cyclical complex activity
Gateways	exclusive choice		Data-based choice; each outgoing relation is associated to a logical guard.
	deferred choice		Non-deterministic choice, without explicit logical conditions.
	parallel fork		Spanning of multiple execution threads
	parallel join		Synchronization of multiple threads of control
Start/Completion Blocks	start		Start point of a complex activity
	cyclic start		Start point of a cyclical complex activity
	completion		Completion point of a complex activity
	abort		Abort the entire guideline

the guideline flow should be executed. Finally, a verification module takes as input the formal specification of the guideline and a log of relevant events. The compliance of such logs w.r.t. the given model is checked by the *SCIFF* Proof Procedure [4], that notifies violations of the logged events w.r.t. the expected behaviour (as specified by the model).

1.1. The GOSpeL Careflow Specification Language

Like a typical flow-chart language, GOSpeL describes the guideline evolution using *blocks* and *relations* between blocks. These blocks are grouped into three families (as shown in Table 1):

activities, blocks which represent guideline activities at the desired abstraction level;
gateways, blocks used to manage the convergence and the divergence of control flow;
start and end, start and end points of (sub)processes.

Relations represent causal binary connections between blocks, expressing that the source block will be performed always before the destination one. Moreover, order relations show how the flow navigates through blocks, imposing a partial ordering among them.

A *simple* activity is a single atomic working step within the guideline: it models a situation where a guideline participant should perform something. *Complex* activities encapsulate new sub-processes definitions or repetitions of activities, and at their specification level they are managed like atomic activities. Gateways are used for modeling complex guidelines as long as they express workflow's decision points and activities concurrence. Decision points can be deterministic, or can be deferred choices. Concurrent activities definitions instead are supported by means of the *parallel fork* and *join* blocks.

To represent domain-related knowledge, GOSpeL adopts an ontology-based approach: the user can define two taxonomies, one for modeling activities at the desired abstraction level; a second taxonomy is used to describe domain's entities, namely ac-

tors, objects and terms. Each atomic activity block is semantically specified by mapping it onto an ontological activity and a set of participants. Creation and management of ontologies is performed through the Protégé [6] tool.

1.2. Formal Representation of the GOSpeL Model and Execution Traces Verification

The GOSpeL graphical model is translated into a simplified version of the formal language proposed by Alberti et al. in the SOCS European project for the specification and verification of interaction protocols (see [4] for a complete description). The social participant behaviour is represented by a set of (ground) facts called *happened events* and denoted by the functor **H** (that stands for “happened”). For example, $\mathbf{H}(cr_ScreeningInvitation(strsrg, center1, lab1, '10-02-06'), 7)$ represents the fact that *center1* has sent the screening invitation to *strsrg* at time 7. Future, desirable behaviour of the participants is represented by means of *expectations* about events that *should/should not* happen. Expectations have the same format as events, but they will, typically, contain variables to indicate that expected events are not completely specified. CLP constraints [7] can be imposed on variables to restrict their domain, e.g., for specifying temporal deadlines. Expectations about events that should happen are also called positive expectations, and are denoted by the functor **E**; expectations about events that should not happen are named negative expectations, and are indicated by the functor **EN**.

The way new expectations are generated, given the happened events and the current expectations, is specified by means of *Social Integrity Constraints (ICs)*. An *ICs* has the form of *body* \rightarrow *head*, expressing that when *body* becomes true then it is supposed that the events specified in *head* will happen. In this way, we are able to define guidelines as sets of forward (or backward) rules, relating happened events (in the *body*) to expectations (in the *head*). Moreover, it is possible to insert in the *head* also special predicates (*abducibles*, with functor *ABD*), typically used for hypothetic reasoning (in this work adopted to signal special situations).

The compliance verification is made by the operational counterpart of *ICs*, an abductive proof procedure named *SCIFF* [8]. Given the partial or the complete history of a specific execution (*i.e.*, the set of already happened events recorded in an event log), this proof procedure generates expectations about participants behaviour so as to comply with *ICs*. The most distinctive feature of *SCIFF*, however, is the ability to check that the generated expectations are *fulfilled* by the actual participants behaviour (*i.e.*, that events expected to happen have actually happened, and forbidden events have not happened). If a participant does not behave as expected w.r.t. the model, the proof procedure detects and raises a violation. In particular, the proof, analyzing a log representing the activities performed during the careflow execution, can detect two kinds of guideline compliance violations: the first when an expected event is not found in the log (**E**(Event) without the relative **H**(Event)); the second when a prohibited event is found in the log (**EN**(Event) with a corresponding **H**(Event)).

2. The Cervical Cancer Screening Guideline

We have applied the GPROVE tools to a real case, the Cervical Cancer Screening Guideline proposed by the sanitary organization of the Emilia Romagna region of Italy [9].

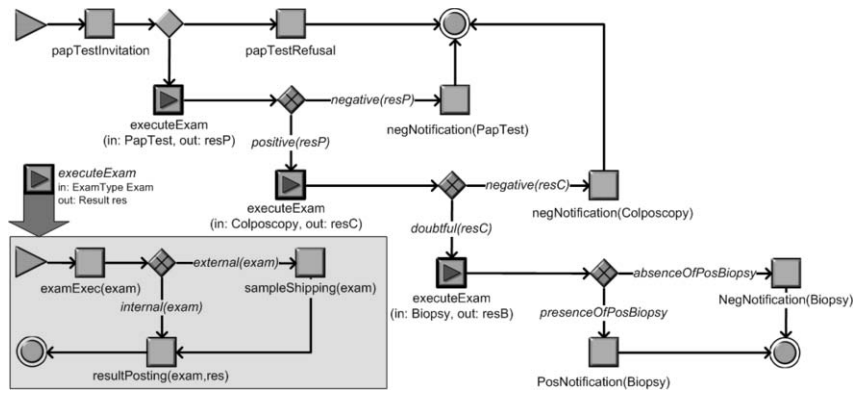


Figure 2. The GOSpeL representation of the example guideline

Cervical cancer is a disease in which malignant (cancer) cells grow in the tissues of the cervix. The screening program proposes several tests in order to early detect and treat cervical cancer, and it is usually organized in six phases: Screening Planning, Invitation Management, First Level Test (PAP-test), Second Level Test (Colposcopy), Third Level Test (Biopsy), and Therapeutic Treatment. Every participant is asked to repeat the screening process periodically (a round being thus three years long).

The workflow modeled by the Cervical Cancer Screening Guideline is quite complex and involves more than 50 activities performed by 15 different health care professionals and structures. For the sake of clarity we discuss here the formalization of a portion of the guideline (named in the paper *ExCervScreening*). *ExCervScreening* prescribes that the screening process starts when the screening center sends a PAP-test invitation to a patient. The patient can decide to refuse or to go to the PAP-test center. In the latter case, the PAP-test is executed and the collected biological sample is sent to the anatomopathology laboratory. A report is sent back to the screening center, reporting a positive classification (cancer evidence found) or a negative classification (normal). In case of a positive report, the patient is invited for a Colposcopy; if positive again the patient is invited for a Biopsy and possibly for cancer treatment. Every time a negative response is reported instead, a letter is sent to the patient and a new screening round is scheduled.

2.1. Formalization of the Guideline into GOSpeL

The first step when formalizing a careflow consists of analyzing the specific domain, and to identify the actors and the activities that more characterize the process. These actors interact with each other by means of activities that involve documents, biological samples, etc. These elements are the objects exchanged during the guideline execution. E.g., the patient, the screening center, the PAP-test center and various laboratories can be identified as *actors*; the invitation letter, the biological samples and the notification letters are *objects*; while the posting of an invitation, the execution of an exam, as well as the shipping of the samples are *activities* performed during the execution of the screening.

In Figure 2 the GOSpeL translation of the *ExCervScreening* is shown. The screening process starts with the activity block *papTestInvitation*, followed by a deferred choice that models the decision taken by the patient (*papTestRefusal* activity block, or *executeExam* macro block, instantiated on *PAP-Test*).

The *executeExam* macro block is used to represent a set of activities that are common to several exams. In particular, the PAP-test, the Colposcopy, and the Biopsy share the *examExec* and the *resultPosting* activities. Moreover, since the PAP-test and the Biopsy samples are analyzed in different labs than the execution ones, they are considered “external” and require a *sampleShipping* activity. The GOSpeL definition of the *executeExam* macro block is also shown in Figure 2: it takes as input the type of the exam, and returns as output the exam result. An exclusive choice is performed, depending on the exam type, and one among two different paths is selected (corresponding to the external/internal exam type). The criteria for such a choice can be defined in a Prolog knowledge base associated to the GOSpeL diagram.

In case the PAP-test is positive², the diagram specifies the execution of the Colposcopy (*execExam* with “colposcopy” as exam type). Again, the result (*resC* in this case) is tested and, if positive or doubtful, a further exam (Biopsy) is performed. Finally, the Biopsy macro block is followed by an exclusive choice used to distinguish if at least one sample has been found positive or not. A notification to the patient follows.

2.2. Formal Representation of the Screening Guideline

The GOSpeL diagram shown in Figure 2 has been automatically translated into a set of SCIFF *ICs*, providing a formal representation. The algorithm [5] makes a partition of the diagram into several sub-sets; then, for each sub-set, an Integrity Constraint is generated, taking into account the links between the different activities involved.

In Eq. 1 it is presented an Integrity Constraint obtained by the translation algorithm. In particular, the IC states that after an event *papTestInvitation* has occurred, either the exam is performed (positive expectation about the event *execExam*) and refusal is forbidden, or a *refuse* event is expected (and the execution of the test is prohibited).

$$\begin{aligned}
 & \mathbf{H}(\text{papTestInvitation}(\text{ScrCentre}, \text{Pat}, \text{Date}, \text{IdExam}), T_{\text{inv}}) \\
 & \rightarrow \mathbf{E}(\text{execExam}(\text{ExamCentre}, \text{Pat}, \text{papTest}, \text{IdExam}, \text{Date}), T_{\text{pap}}) \wedge T_{\text{pap}} > T_{\text{inv}} \\
 & \quad \wedge \mathbf{EN}(\text{refuse}(\text{ScrCentre}, \text{Pat}, \text{Date}), T_{\text{ref}}) \wedge T_{\text{ref}} > T_{\text{inv}} \\
 & \vee \mathbf{E}(\text{refuse}(\text{ScrCentre}, \text{Pat}, \text{Date}), T_{\text{ref}}) \wedge T_{\text{ref}} > T_{\text{inv}} \\
 & \quad \wedge \mathbf{EN}(\text{execExam}(\text{ExamCentre}, \text{Pat}, \text{papTest}, \text{IdExam}, \text{Date}), T_{\text{pap}}) \wedge T_{\text{pap}} > T_{\text{inv}}.
 \end{aligned} \tag{1}$$

It is worthy to notice several aspects of the rule presented in Eq. 1. Firstly, the rule specifies for each event a set of parameters not present in Figure 2. Such list of parameters has been derived by the ontology definition of the activity *papTestInvitation*. Secondly, temporal constraints have been imposed over the variables T_{inv} , T_{pap} and T_{ref} , to define the desired temporal sequence of the events (i.e., a *refuse* activity should happen *after* the invitation for the test, and not before). Finally, negative expectations have been introduced to make exclusive the choice between attending the test and refusing: i.e., it is not possible that a patient *Pat* attends the PAP-test and also refuses to make it.

Eq. 2 and 3 instead show how the macro block *executeExam* has been represented. In particular, the choice between the shipping of the biological sample and the result notification is made by using the user-defined prolog predicates *external(X)* and *internal(X)*.

²To this end, two predicates *positive(resP)* and *negative(resP)* have been defined.

$$\begin{aligned} & \mathbf{H}(\text{execExam}(\text{ExamCentre}, \text{Pat}, \text{ExamType}, \text{IdExam}, \text{Date}), T_{ex}) \\ \rightarrow & \mathbf{E}(\text{sampleShipping}(\text{ExamType}, \text{IdExam}, \text{Date}), T_{ship}) \wedge T_{ship} > T_{ex} \\ & \wedge \text{external}(\text{ExamType}) \end{aligned} \quad (2)$$

$$\begin{aligned} & \forall \mathbf{E}(\text{resultPosting}(\text{ExamType}, \text{IdExam}, \text{Date}, \text{Res}), T_{res}) \wedge T_{res} > T_{ex} \\ & \wedge \text{internal}(\text{ExamType}). \\ & \mathbf{H}(\text{sampleShipping}(\text{ExamType}, \text{IdExam}, \text{Date}), T_{ship}) \\ \rightarrow & \mathbf{E}(\text{resultPosting}(\text{ExamType}, \text{IdExam}, \text{Date}, \text{Res}), T_{res}) \wedge T_{res} > T_{ship}. \end{aligned} \quad (3)$$

3. Compliance Verification

The *ExCervScreening* guideline, whose GOSpel diagram is shown in Figure 2, has been automatically translated into fourteen different *ICs*. These constraints have been provided as input to the SOCS-SI [8] tool for verifying the v of the logs.

In the case of the Emilia Romagna sanitary organization, all the cancer screening information are recorded in a database. A syntactical translation towards a more suitable and “log-like” representation has been applied to the data, and then the compliance checking has been performed. Let us consider for example a simple execution of the above careflow process, represented by a set of happened events:

1. $\mathbf{H}(\text{papTestInvitation}(\text{scr1}, '00000260', '1995-11-6', 'b_95017025'), 5)$
2. $\mathbf{H}(\text{execExam}(\text{hosp1}, '00000260', \text{papTest}, 'b_95017025', '1995-11-6'), 7)$
3. $\mathbf{H}(\text{sampleShipping}(\text{papTest}, 'b_95017025', '1995-11-6'), 20)$
4. $\mathbf{H}(\text{resultPosting}(\text{papTest}, 'b_95017025', '1995-11-22', \text{neg}), 30)$
5. $\mathbf{H}(\text{negativeNotification}(\text{scr1}, '00000260', \text{papTest}, 'b_95017025', '1995-11-22', \text{neg}), 30)$

This log represents a typical sequence of events: the patient ‘00000260’ is invited to the screening, and she attends the first-level test (log entries 1 and 2); then the biological sample is sent to the laboratory, and the result is sent back to the screening center (log entries 3 and 4). Finally, a notification letter is sent to the patient (log entry 5).

Such a log is indeed compliance with the given careflow specification: each event triggers one or more Integrity Constraints, thus generating expectations about future events. E.g., log entry 1 triggers the rule shown in Eq. 1: an expectations about executing the exam is generated. Such expectation is fulfilled by the event traced in log entry 2. Suppose now that the *execExam* event would not be present in the log: then, the expectation about it would remain not fulfilled. The SCIFF proof procedure then would try to check if the alternative behaviour specified in Eq. 1 could be satisfied instead: unfortunately, a *refuse* event is missing, and the SCIFF proof procedure would raise a violation.

3.1. Verification of the Cancer Screening Logs

The compliance verification approach discussed in this paper has been applied to a database containing 1950 careflow executions. The careflow model has been specified by the authors, on the base of the cancer screening process [9].

In order to fully test our tools, some wrong behaviours have been introduced in this database. Each screening round has been checked as a single interaction (hence we did not check the compliance for the repetition of the screening rounds). Each screening contains several events: from the minimum of one (the screening invitation followed by no response) to the maximum of 18 (the whole careflow). The total time occurred to verify the compliance of the 1950 executions w.r.t. the careflow model was 12 minutes (average time of 369 msec. for each execution). 1091 executions resulted to be not compliance w.r.t. the formalization we have initially proposed. These results were analyzed by a screening expert which confirmed all the compliant classifications and proposed some changes to the careflow model: in fact, some traces classified as compliant by the domain expert were instead considered as non compliant w.r.t. the initial model. E.g., it was not taken into account that some patients, asked to participate to the screening, simply decide to not answer at all. Then Eq. 1 has been substituted by Eq. 4.

$$\begin{aligned}
& \mathbf{H}(\text{papTestInvitation}(\text{ScrCentre}, \text{Pat}, \text{Date}, \text{IdExam}), T_{inv}) \\
\rightarrow & \mathbf{E}(\text{execExam}(\text{ExamCentre}, \text{Pat}, \text{papTest}, \text{IdExam}, \text{Date}), T_{pap}) \wedge T_{pap} > T_{inv} \\
& \wedge \mathbf{EN}(\text{refuse}(\text{ScrCentre}, \text{Pat}, \text{Date}), T_{ref}) \wedge T_{ref} > T_{inv} \\
\vee & \mathbf{E}(\text{refuse}(\text{ScrCentre}, \text{Pat}, \text{Date}), T_{ref}) \wedge T_{ref} > T_{inv} \tag{4} \\
& \wedge \mathbf{EN}(\text{execExam}(\text{ExamCentre}, \text{Pat}, \text{papTest}, \text{IdExam}, \text{Date}), T_{pap}) \wedge T_{pap} > T_{inv} \\
\vee & \mathbf{ABD}(\text{warning}(\text{invitation_not_respected}), T_{inv}) \\
& \wedge \mathbf{EN}(\text{AnyEvent}, T_{any}) \wedge T_{any} > T_{inv}.
\end{aligned}$$

In Eq. 4, the possibility of abducting a *warning* predicate has been used to consider as compliant a log composed by the invitation event only (see [4] for more details on the SCIFF abductive framework). Furthermore, any event after the invitation has been prohibited by using a negative expectation: hence, the logs considered as compliant by the modified diagram are those logs that contain only the invitation event and nothing else. Using this revised model, we avoided false non compliant classifications, reducing the number of executions classified as non compliant to 44: this result agrees indeed with the “wrong behaviour” executions we artificially introduced.

3.2. Verifying Particular Logs Features

The SCIFF Proof Procedure and the SOCS-SI tools can be used also to verify particular features or situations that characterize a certain log. For example, it could be interesting to signal when certain situations happens, without raising a violation. This can be easily done by extending the formal representation of the careflow process, e.g., by adding new integrity constraints, or by modifying existing ones. To illustrate this capability, let us introduce some details about the invitation. Contextually to the invitation, a date for the PAP-test is automatically booked and proposed to the patient. The patient can refuse the proposed date, and phone directly to the screening center for booking another appointment. This process is not explicitly defined in the screening guideline, and the database does not store any information about it. The only information stored is about the invitation to the PAP-test, and the execution of the exam (together with the date it

was executed). To understand how frequently this situation can happen, we added a new integrity constraint, shown in Eq. 5, to the formal specification.

$$\begin{aligned}
& \mathbf{H}(\text{papTestInvitation}(\text{ScrCentre}, \text{Pat}, \text{Date}, \text{IdExam}), T_{inv}) \\
& \wedge \mathbf{H}(\text{execExam}(\text{ExamCentre}, \text{Pat}, \text{papTest}, \text{IdExam}, \text{DateExec}), T_{pap}) \\
& \wedge \text{DateExec} - \text{Date} > 15 \\
& \rightarrow \text{ABD}(\text{warning}(\text{delay_higher_than_15_days}, \text{Date}, \text{DateExec}), T_{inv}).
\end{aligned} \tag{5}$$

Eq. 5 states that if a patient has been invited to attend the PAP-test, and the exam took place more than 15 days later the scheduled exam, a warning should be issued. We repeated the analysis of the logs, and we discovered that 200 times the PAP-test has been attended more than 15 days later w.r.t. the initial schedule. The delay could be explained by the fact that the screening center allocates in advance a certain number of slots: as a consequence, free slots for new booking are not immediately available.

4. Related Work and Conclusions

Several medical support systems have been proposed to represent and manage clinical guidelines, and some of them support also various verification tasks. In [10], compliance of the actual treatment of a specific patient is checked with the ideal behavior prescribed by the guideline by using temporal logic and model checking techniques. Two non-compliances are considered: non-compliant action ordering (i.e., prescribed actions are performed, but in a wrong order) and non-compliant actions (i.e., some guideline's actions cannot be prescribed at all for a specific patient). GPROVE is able to identify both the types of non-compliances; moreover, thanks to its computation logic roots, can also address other types of non-compliances such as, e.g., temporal deadlines verification. In fact, one of the main advantages of using a first order language such as computational logic w.r.t. more classical approaches based on temporal logics such as LTL or CTL, is the possibility of introducing variables and reasoning upon them. Hence, GPROVE naturally supports temporal constraints (intended as CLP constraints over a variable representing a time point), as well as constraints over terms containing variables.

Other approaches focus instead on the static verification of general and domain-dependent properties, aiming at identifying design errors and inconsistencies. A typical approach consist of mapping guidelines specification into suitable formal languages in order to provide automatic tools for verification. In [11] the authors propose the translation of GLARE [12] to the SPIN Model Checker, in order to discover inconsistencies in the model. In [13] a clinical guideline specified with Asbru [14] is viewed as a hierarchical plan and mapped onto the KIV interactive theorem prover. A variant of Interval Temporal Logic is then used to specify and verify properties.

In this work, we have introduced the GPROVE framework, showing its application to a real case, namely the Cancer Screening Guideline of the Emilia Romagna region. We have shown, by formalizing the guidelines in the logic-based formalism SCIFF, how it is possible to perform the compliance checking of the execution traces w.r.t. the modeled careflow process. Such verification can be used to identify undesired behaviours, to

comprehend features of the real process not highlighted by the model, and to identify strengths and weaknesses of the careflow process.

Future work will be devoted to complete the development of the GPROVE modules. Other graphical formalisms will be considered for the guideline specification, with the goal of providing a mapping to the SCIFF language, and to support compliance checking also in the context of other frameworks. Future extensions will tackle run-time verification, static verification of properties, and enactment of guidelines models.

Acknowledgements

This work has been partially supported by NOEMALIFE under the “SPRING” regional PRRITT project, by the PRIN 2005 project “Specification and verification of agent interaction protocols” and by the FIRB project “TOCAL.IT”. We would like to thank reviewers and participants of the “ECAI 2006 Workshop on AI Techniques in Healthcare - Evidence-based Guidelines and Protocols” for the fruitful discussions on a preliminary version of this work. We would like to thank also Dr. Natalina Collina (Sanitary Agency of Bologna) for her contribution on the screening process management.

References

- [1] G.J.A. Muir. *Evidence-based Healthcare*. Churchill Livingstone, London, 1997.
- [2] Careflow management systems. <http://www.openclinical.org/briefingpaperStefanelli.html>.
- [3] F. Chesani, P. De Matteis, P. Mello, M. Montali, and S. Storari. A framework for defining and verifying clinical guidelines: A case study on cancer screening. In F. Esposito, Z. W. Ras, D. Malerba, and G. Semeraro, editors, *ISMIS*, volume 4203 of *LNCS*, pages 338–343. Springer, 2006.
- [4] M. Alberti, F. Chesani, M. Gavaneli, E. Lamma, P. Mello, and P. Torroni. Verifiable agent interaction in abductive logic programming: the SCIFF framework. *ACM Transactions on Computational Logics*. Accepted for publication.
- [5] M. Montali F. Chesani, P. Mello and S. Storari. Testing careflow process execution conformance by translating a graphical language to computational logic. In Bellazzi et al. [15].
- [6] Protegé. <http://protege.stanford.edu/>.
- [7] J. Jaffar and M.J. Maher. Constraint logic programming: a survey. *JLP*, 19-20:503–582, 1994.
- [8] The SCIFF abductive proof procedure. Available at <http://lia.deis.unibo.it/Research/sciff/>.
- [9] Cervical Cancer Screening in Emilia Romagna (Italy). <http://www.regione.emilia-romagna.it/screening/>.
- [10] P. Groot, A. Hommersom, P. J. F. Lucas, R. Serban, A. ten Teije, and F. van Harmelen. The role of model checking in critiquing based on clinical guidelines. In Bellazzi et al. [15], pages 411–420.
- [11] L. Giordano, P. Terenziani, A. Bottrighi, S. Montani, and L. Donzella. Model checking for clinical guidelines: an agent-based approach. In *Proc. of the American Medical Informatics Association (AMIA) Symposium*, 2006.
- [12] P. Terenziani, S. Montani, A. Bottrighi, M. Torchio, G. Molino, and G. Correndo. Applying artificial intelligence to clinical guidelines: The GLARE approach. In *AI*IA*, volume 101, pages 536–547, 2003.
- [13] A. ten Teije, M. Marcos, M. Balsler, 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. *Artificial Intelligence in Medicine*, 36(3):193–209, 2006.
- [14] Y. Shahar, S. Miksch, and P. Johnson. The asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines. *Artif. Intelligence in Medicine*, 14(1-2):29–51, 1998.
- [15] R. Bellazzi, A. Abu-Hanna, and J. Hunter, editors. *11th Conf. on Artificial Intelligence in Medicine, AIME, Amsterdam, The Netherlands, July 7-11, 2007, Proc.*, volume 4594 of *LNCS*. Springer, 2007.