Testing Careflow Process Execution Conformance by Translating a Graphical Language to Computational Logic

Federico Chesani[1], Paola Mello[1], Marco Montali[1], and Sergio Storari[2]

[1] DEIS – Università di Bologna
viale Risorgimento, 2 – 40136 – Bologna, Italy
{fchesani|pmello|mmontali}@deis.unibo.it
[2] ENDIF – Università di Ferrara
Via Saragat, 1 – 44100 – Ferrara, Italy
strsrg@unife.it

Abstract. Careflow systems implement workflow concepts in the clinical domain in order to administer, support and monitor the execution of health care services performed by different health care professionals and structures. In this work we focus on the monitoring aspects and propose a solution for the conformance verification of careflow process executions.

Given a careflow model, we have defined an algorithm capable of translating it to a formal language based on computational logic and abductive logic programming in particular. The main advantage of this formalism lies in its operational proof-theoretic counterpart, which is able to verify the conformance of a given careflow process execution (in the form of an event log) w.r.t. the model.

The feasibility of the approach has been tested on a case study related to the careflow process described in the cervical cancer screening protocol.

Keywords: Careflow management, Clinical practice guidelines, Conformance verification, Computational logic.

1 Introduction

In modern health care organizations, clinical decisions are progressively based on evidence-based care [1]. In order to achieve the goals of this approach, the adoption of clinical practice guidelines and computer-based guideline management systems is considered very important. These guidelines are effectively used in practice if they are managed by systems that are deeply integrated with the health care information systems and take care of the aspects related to the clinician's workflow (namely *careflow*).

As described in [2], careflows focus on the behavioural aspects of medical work described in clinical practice guidelines. Careflow systems implement workflow concepts in the clinical domain, coordinating the execution of health care services performed by different health care professionals and structures. The literature proposes several formalisms to represent workflows/careflows. Usually, they are graphical flow-charts that clearly express the sequence of activities to be performed. This formalization is used by the workflow management systems to administer, support and monitor the process execution.

We focus on the monitoring aspects and present a preliminary result of our research activity aimed to perform the conformance verification of careflow process executions. We have developed a tool for describing careflow models in a graphical language (named GOSpeL [3]). In this work we present an algorithm for translating a GOSpeL model into a set of rules written in a declarative language based on computationl logic, named $\mathcal{S}$CIFF [4]; careflow participants are mapped to agents, activities to events and medical knowledge to a knowledge base. Such a formalization is used by the operational proof-theoretic counterpart of the $\mathcal{S}$CIFF language, which is able to verify the conformance of a given careflow process execution (in the form of a log of the events) w.r.t. the model. We discuss how the result of the verification step could be used for better understanding the careflow model as well as for pointing out possibly non-forecasted behaviours.

2 An Overview of GOSpeL

GOSpeL [3] is a software tool for specifying careflows by means of a graphical language. In GOSpeL, a careflow is represented in terms of a (1) *flow chart*, which models the process evolution; and (2) an *ontology*, which describes at a fixed level of abstraction the application domain and provides a semantics to the flow-chart.

Careflow process evolutions are described by means of blocks (shown in Table 1) and relations between blocks. GOSpeL blocks accounts for *activities* (work units at the desired abstraction level), *gateways* (to manage the convergence and the divergence of control flows), and *start/completion point* of complex activities. A complex activity models a composite unit of work, defined at a lower abstraction level by a new (sub)process, thus allowing a top-down approach for the careflow definition. GOSpeL blocks can be connected by using *order relations* (that represent the flow path), or by using *temporal relations* (to model temporal constraints among activities like e.g. deadlines and delays). GOSpeL ontologies follow a simple template, supporting mainly two taxonomies: *Activities* to model the atomic ontological activities of the target domain; *Entities* to model actors and objects involved in an activity execution.

3 A Brief Description of the $\mathcal{S}$CIFF Framework

The $\mathcal{S}$CIFF framework was originally developed in the context of the SOCS European project [5] for the specification and verification of agents interaction protocols within open and heterogeneous societies. The framework is based on abduction, a reasoning paradigm (well suitable in the medical field) which allows to formulate hypotheses (called *abducibles*) accounting for observations. In most abductive frameworks, *integrity constraints* are imposed over possible hypotheses in order to prevent inconsistent explanations. The idea behind the SOCS framework is to formalize domain knowledge in terms of abductive logic programming and expectations about participants behaviour as abducibles, and to use Social Integrity Constraints (IC_S) to detect such behaviour that is not compliant with interaction protocols. It is assumed that the ongoing social participants behaviour is accessible and represented by a set of (ground) facts called *events*. Happened events are denotated by the functor $\mathbf{H}$: e.g.,

Table 1. GOSpeL 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, and at evaluation-time, one of the path which has a a true condition is chosen.
	deferred choice	◇	Non-deterministic choice, without explict logical conditions; the choice is delayed until one of the possible paths is actually performed by participants.
	parallel fork	🟩	Point at which multiple threads of execution are spanned
	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

$\mathbf{H}(enter(p, emergency\_ward), 7)$ represents the fact that p has entered into the hospital's $emergency\_ward$ at time 7.

Generally speaking, the participants behaviour is unpredictable. However, interaction protocols provides hints about which are the possible expectations about future events. This represents in some sense the "ideal" behaviour. Events expected to happen are indicated by the functor $\mathbf{E}$ and have the same format as happened events but they will typically contain variables, to indicate that expected events are not completely specified. These variables may be constrained by using CLP constraints [6] or bound by evaluating predicates defined in the $\mathcal{S}$CIFF abductive knowledge base (named SOKB).

Given the happened events, IC_s specifies how to generate expectations. An IC is a rule of the form $body \rightarrow head$, expressing that when $body$ becomes true then $head$ is expected. Protocols are defined as sets of rules, relating happened events to expectations about future events. E.g.:

$$\begin{aligned} &\mathbf{H}(enter(Pat, emergency\_ward), T_1) \wedge high\_priority(Pat) \\ \rightarrow &\mathbf{E}(examine(Phy, Pat), T_2) \wedge T_2 > T_1 \wedge T_2 < T_1 + 15 \end{aligned} \quad (1)$$

states that, if a patient Pat enters into the emergency ward and it is evaluated as "high priority", then a physician Phy is expected to examine him/her within a quarter of an hour (supposing that times are expressed in minutes). The $high\_priority/1$ predicate is defined in the SOKB. Notice that temporal constraints can be imposed (in particular deadlines).

Given a set of happened events (i.e. an event log or history), expectations are generated by he operational counterpart of the $\mathcal{S}$CIFF language, namely the $\mathcal{S}$CIFF Proof Procedure [7]. The most distinctive feature of this proof procedure is the ability to check that the generated expectations, considered as a particular class of abducibles, are *fulfilled* by the actual participants behaviour. If a participant does not behave as expected w.r.t. the model, the proof procedure detects and raises as soon as possible a violation.

Our approach is suitable for modeling the careflow aspects of a clinical practice guideline, especially when the execution order and the appropriateness of the health services should be strongly enforced (like for example screening protocols). In this context, we are interested in detecting two different types of violation. The first one is raised when a participant does not act as expected by the careflow model (i.e., an expectation is not fulfilled by a corresponding happened event); the second one is raised when a participant performs activities not expected by the model (i.e., a happened event is not explicitly expected). When a violation is detected, two possible hypothesis could be given in order to explain such a violation: either the participants exhibited a wrong behavior (w.r.t. the careflow model), or the model itself has not been properly defined (hence it does not fit well with the real guideline's execution). Assuming the latter hypothesis, violations are a useful to understand *how* and *where* the careflow model specification lacks.

4 Translation Algorithm

Intuitively, a careflow model specifies that, when an activity block is performed, other activities should be performed in the right order and with the right attributes. From the $\mathcal{S}$CIFF viewpoint, this is equivalent to specify an IC that relates the happened event with the future ones. Given an activity block A, the part of the diagram next to A is considered as a description of the possible behaviours which the participants has to exhibit. Therefore, the algorithm generates an IC whose body contains the happening of e_A and whose head is determined by the consequent diagram part. The notation e_A represents the event to which a generic block A has been ontologically mapped[1]; the properties of this event, namely its name and its attributes, are respectively determined by the name of the ontological activity and the set of formal participants associated to A. Leaving A and going forward in the graphical model, for each branch a new activity block will be detected (sooner or later), and will be mapped to an expectation about the future participants behaviour. Afterward, these blocks will be considered (recursively) as new starting points by the algorithm.

Note that start, return and end blocks are mapped to events too: even if they do not really represent a concrete working step during the process application, they are used as terminal points that identify the start and the conclusion of a (sub)process. In the following, we will refer to the blocks which are mapped to events as *event-blocks*.

The algorithm visits a GOSpeL diagram partitioning it into special sub-sets (called *minimal windows*) and translating each sub-set to a Social Integrity Constraint. In order to define a minimal window, we introduce some other concepts: *precursors* and *successors* sets, *path*, *window* and window's *source* and *fringe*.

Definition 1 (Precursors and Successors Sets). *Given a block b:*

- *Suc_b is the set of blocks to which b is directly connected through its outgoing relations (successors set);*

[1] We adopt an atomic model for simple activities.

– *Pre_b is the set of blocks to which b is directly connected through its incoming relations (precursors set).*

Definition 2 (Path). *A path $P(s, d)$ is a sequence of blocks through which block s and block d are connected, following the order relations. Defining the sequence as $b_0 = s, b_1, \ldots, b_{n-1}, b_n = d$, we have:*

$$b_j \in Suc_{b_{j-1}} \wedge b_j \in Pre_{b_{j+1}} \forall j = 1, \ldots, n-1$$

Definition 3 (Window). *A subset $\mathcal{W}$ of GOSpeL blocks is a* window *if it is connected, i.e.*

$$\forall b_1 \in \mathcal{W} \exists b_2 \in \mathcal{W} s.t. \exists P(b_1, b_2) \in \mathcal{W} \vee \exists P(b_2, b_1) \in \mathcal{W}$$

Note that $P(s, d) \in \mathcal{W}$ iff all the blocks of the sequence belong to $\mathcal{W}$.

Definition 4 (Window Source and Fringe). *The* source *and the* fringe *of a window $\mathcal{W}$ are respectively:*

$$S_{\mathcal{W}} = \{b \in \mathcal{W} \mid \nexists b' \in \mathcal{W} \ s.t. \ \exists P(b', b)\}$$

$$F_{\mathcal{W}} = \{b \in \mathcal{W} \mid \nexists b' \in \mathcal{W} \ s.t. \ \exists P(b, b')\}$$

Definition 5 (Minimal Window). *A window $\mathcal{W}$ is* minimal *iff $\forall b \in \mathcal{W}$ the following properties hold:*

1. *if $b \in S_{\mathcal{W}}$ then b is an event-block;*
2. *else if $b \in F_{\mathcal{W}}$ then b is an event-block;*
3. *else b is not an event-block (i.e. it is a split or merge);*
4. *if b is a split-block then $Suc_b \in \mathcal{W}$;*
5. *if b is a merge-block then $Pre_b \in \mathcal{W}$.*

Properties 4 and 5 of Def. 5 ensure that when a split-block (a merge-block respectively) belongs to the minimal window, all the branches which diverge from (converge to, resp.) it are included. Note also that, for a well-formed flow-chart, it is impossible to have a window that contains a split-block followed by a merge one (each path that connects two blocks of this type must include at least an activity-block between them).

4.1 Mapping of a Minimal Window to an IC

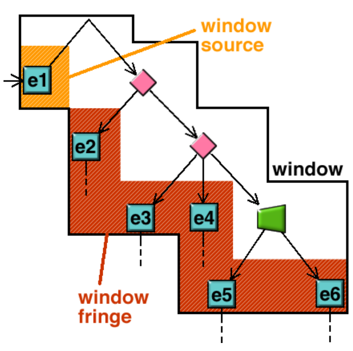
Figure 1(a) shows a minimal window and Figure 1(b) its translation. It's easy to see a tight similarity between the minimal window and the abstract parse tree of the corresponding IC.

The translation procedure of a minimal window $\mathcal{W}$, named in the following $GENERATE\_IC$, operates as follows [2,3]:

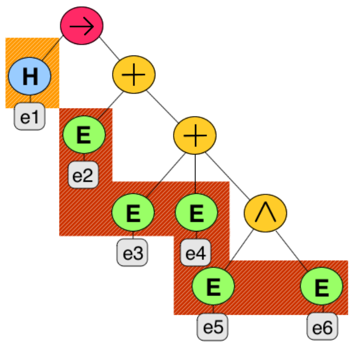
1. $\forall b \in S_{\mathcal{W}}$ generates $\mathbf{H}(e_b, T_b)$ (if b is a macroblock, its completion point is chosen);
2. creates a body of a rule by composing the happened events in a way that depends on the merge-blocks in $\mathcal{W}$;

[2] Remember that $S_{\mathcal{W}}$ and $F_{\mathcal{W}}$ contain only event-blocks (Property 1 and 2 of Definition 5).

[3] For the sake of clarity, we make the assumption that each ontological activity is associated at most to one activity block. The general algorithm does not require to state this assumption.



(a)



(b)

Fig. 1. Minimal window and abstract parse tree of its translation

3. $\forall b \in F_{\mathcal{W}}$ generates $\mathbf{E}(e_b, T_b)$ (if b is a macroblock, its start point is chosen);
4. creates a head composing the expectations in a way that depends on the split-blocks in $\mathcal{W}$.

Let us consider for example a parallel join, the only merge-block defined in GOSpeL: due to its synchronization semantics the generated IC must trigger (i.e., must have a body that becomes true) only when all the previous events happen. Therefore, in presence of a parallel join the body will contain a conjunction of the happened events generated during the first step; Property 5 of Definition 5 ensures that all the previous events are considered for the synchronization.

Similarly, split blocks determine how the expectations generated in the third step are composed; as Figure 1 suggests, the parallel fork is mapped to a logical conjunction among the expectations found on each outgoing branch, whereas the semantics of deferred choice imposes mutual exclusion between branches, generating a rule that waits for one among several events. Furthermore, the behaviour of an ex-or blocks is the same as the deferred choice one, despite the fact that each alternative is associated to its logical condition (i.e., it can be chosen iff the associated guard is evaluated to true).

4.2 General Algorithm

Giving the start block $Start$ of a GOSpeL model, the translation algorithm operates splitting the whole diagram into a set of minimal windows and mapping each window to a an IC:

```
1: $ics = \emptyset, visited = \emptyset, fringe \leftarrow Start$
2: while $fringe \neq \emptyset$ do
3:     $cur \leftarrow REMOVE\_ONE(fringe)$
4:     $\mathcal{W} \leftarrow CONSTRUCT\_MINIMAL\_WINDOW(cur)$
5:     $ics \leftarrow ics \cup GENERATE\_IC(\mathcal{W})$
6:     $visited \leftarrow visited \cup S_{\mathcal{W}}$
7:     $fringe \leftarrow [fringe \cup F_{\mathcal{W}}] - visited$
8: end while
```

The $fringe$ set, which initially contains only $Start$, represents dynamically the frontier of the already covered part. At each iteration step, one element is extracted from $fringe$, say, cur. At line 4, the minimal window $\mathcal{W}$ s.t. $cur \in S_{\mathcal{W}}$ is founded. Operationally, $\mathcal{W}$ is constructed starting from cur and visiting the diagram partially forward and partially backward (when a merge block is encountered, Property 5 of Definition 5 says that all its previous branches should be included). The mapping of cur is then handled by the $GENERATE\_IC$ procedure, which has been described in the previous paragraph. Finally, the $visited$ and $fringe$ sets are updated to avoid repetition: remember indeed that in GOSpeL different alternatives may converge to a single path. Figure 2 shows how a fragment of a simple diagram is partitioned into minimal windows.

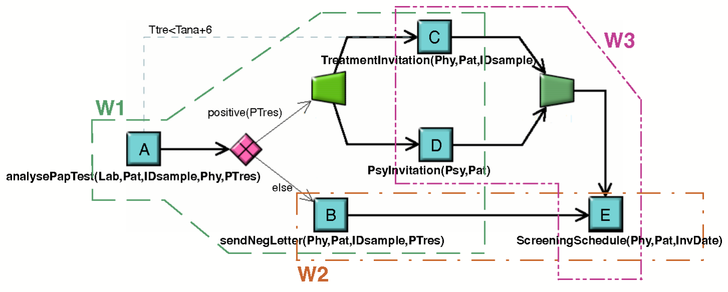

Ttre<Tana+6
W1
W2
W3
A
B
C
D
E
positive(PTres)
else
analysePapTest(Lab,Pat,IDsample,Phy,PTres)
TreatmentInvitation(Phy,Pat,IDsample)
PsyInvitation(Psy,Pat)
sendNegLetter(Phy,Pat,IDsample,PTres)
ScreeningSchedule(Phy,Pat,InvDate)

Fig. 2. Example of a GOSpeL diagram fragment

5 A Case Study

As a case study for exploiting the potentialities of our approach we choose the cervical cancer screening guideline proposed by the sanitary organization of the Emilia Romagna region of Italy [8]. Cervical cancer is a disease where 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.

For the sake of space, we describe in this section its application to the careflow model of a simplified cervical cancer screening protocol. In this careflow, a lab Lab analyses a pap-test $IDsample$ of patient Pat and sends a report $PTres$, containing a set of signs on the sample, to the screening physician Phy. Phy evaluates $PTres$ and classifies $IDsample$ as positive (cancer evidence found) or negative (normal). If positive, the protocol prescribes that Pat should be invited, in parallel, for the cancer treatment and for a psychological consult. Note that the treatment invitation should be sent to the patient within a deadline of six days. In case of a negative evaluation a letter should be sent to Pat reporting that the pap-test is normal.

The positive and negative flows converge in a single one which proposes as activity the scheduling of the next pap-test.

The GOSpeL model of this careflow is composed by an extension of the base ontology, which contains entities and activities specific of the screening domain, and by the graphical model shown in Figure 2. This model is then translated, by the algorithm described in Section 4, in a set of ICs starting from block A with $fringe$ set to $\{A\}$. At the first iteration step the algorithm extracts A and, launching a visit from it, individuates $\mathcal{W}_1$, which has $S_{\mathcal{W}_1} = \{A\}$ and $F_{\mathcal{W}_1} = \{B, C, D\}$. The following IC is produced:

$$\begin{aligned}
&\mathbf{H}(analysePapTest(Lab, Pat, IDsample, Phy, PTres), T_{ana})\\
\rightarrow &positive(PTres)\\
&\wedge \mathbf{E}(treatmentInvitation(Phy, Pat, IDsample), T_{tre})\\
&\wedge T_{tre} > T_{ana} \wedge T_{tre} < T_{ana} + 6\\
&\wedge \mathbf{E}(psyInvitation(Psy, Pat), T_{psy}) \wedge T_{psy} > T_{ana}\\
\vee &not(positive(PTres))\\
&\wedge \mathbf{E}(sendNegLetter(Phy, Pat, IDsample, PTres), T_{sen}) \wedge T_{sen} > T_{ana}
\end{aligned} \tag{2}$$

Note that the temporal constraint between A and C is inserted as a CLP constraint over T_{tre} and T_{ana}. Other time constraints are automatically generated due to the partial order imposed by order relations. The exclusive choice condition is mapped to the evaluation of the predicate *positive/1*, contained in the SOKB. A pap-test is positive if almost one cervical cancer type can be detected. Since each cancer type is characterized by a specific set of laboratory results, the predicate *positive/1* verifies if almost one of three possible cancer types has more than an half of its supporting signs in $PTres$. This is a trivial description used only in order to exploit the reasoning capabilities of the SOKB. Now the algorithm proceeds updating the $fringe$ set, which becomes $fringe = \{B, C, D\}$. Supposing B is extracted, the algorithm finds window $\mathcal{W}_2$, whose translation is straightforward. After having translated $\mathcal{W}_2$ the fringe contains C, D and E. If either block C or D are extracted, due to the presence of a parallel join the algorithm finds a window which has $S_{\mathcal{W}_2} = \{C, D\}$ and $F_{\mathcal{W}_2} = \{E\}$, and generates the IC (3). The final set of ICs for the cervical cancer screening example is composed by three rules.

$$\begin{aligned}
&\mathbf{H}(treatmentInvitation(Phy, Pat, IDsample), T_{tre})\\
&\wedge \mathbf{H}(psyInvitation(Psy, Pat), T_{psy})\\
\rightarrow &\mathbf{E}(screeningSchedule(Phy, Pat, InvDate), T_{scr}) \wedge T_{scr} > T_{tre} \wedge T_{scr} > T_{psy}
\end{aligned} \tag{3}$$

Given this set of ICs, the $\mathcal{S}$CIFF proof procedure is used by the SOCS-SI [9] tool for verifying the conformance. Let us consider for example a simple execution of the above careflow process represented by a set of happened events:

1. $\mathbf{H}(analysePapTest(lab, pat, 123, phy, [res_1, \ldots, res_n]), 5)$
2. $\mathbf{H}(psyInvitation(psy, pat), 7)$
3. $\mathbf{H}(treatmentInvitation(phy, pat, 123), 20)$
4. $\mathbf{H}(screeningSchedule(phy, pat, 15apr2007), 30)$

When the pap-test analysis is passed to the proof procedure, the first IC triggers and, supposing that the predicate $positive([res_1, \ldots, res_n])$ succeeds, we have two pending expectations: $\mathbf{E}(treatmentInvitation(phy, pat, 123), T_{tre}) \wedge T_{tre} \in [6, 11]$ and $\mathbf{E}(psyInvitation(Psy, pat), T_{psy}) \wedge T_{psy} > 5$. Now we have that the second happened event fulfills the second expectation, grounding Psy to psy and T_{psy} to 20, whereas the treatment invitation event matches with the first one. Unfortunately, the match implies that T_{tre} unifies with 30, which does not satisfy the deadline and causes therefore a violation to be raised. The execution is then classified as non conformant.

This conformance verification approach has been evaluated by using the careflow model of the cervical cancer screening process [8] and a database containing 1950 careflow executions. Some of them, representing incorrect behaviours, were introduced in this database, in order to deeply test our approach and our tools. Each execution 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 conformance of the 1950 executions w.r.t. the careflow model was 12 minutes (average time of 369msec for each execution). 1091 executions resulted to be not conformant w.r.t. the formalization we have initially proposed. These results were analyzed by a screening expert which confirmed all the conformant classifications and proposed some changes to the careflow model in order to consider as conformant some particular cases, not allowed by the initial model. Using this revised model, we avoided false non conformant classifications, reducing the number of executions classified as non conformant to 44: this result agrees indeed with the "wrong behaviour" executions we introduced in the database. The conformance results were considered useful by the screening expert for the quality evaluation of the careflow process and its revision.

6 Related Works

Several medical guidelines support systems have been proposed to represent and manage clinical guidelines but, for the sake of space, we limit ourselves to only three: GLARE [10], PROforma [11] and NewGuide [12]. GLARE [10] is a system for acquiring, representing and executing clinical guidelines. It provides consistency checks, advanced temporal reasoning techniques, what-if functionalities and guideline properties evaluation. PROforma [11] is a formal language capable to represent a clinical guideline in terms of a network of tasks and data items. NewGuide [12] puts together medical knowledge formalization techniques and workflow management systems (named Careflow Management Systems CfMS). The system supports the definition (in a language similar to Petri Nets), execution and monitoring of guidelines and careflows.

Comparing GLARE, PROforma and NewGuide with our approach we notice that our approach can be considered complementary w.r.t the ones proposed by GLARE and PROforma, since they do not tackle conformance verification issues on careflow execution traces. With respect to NewGuide, we think that our approach may be useful to add verification functionalities to the CfMS administration and monitoring tools [12].

7 Conclusions

In this work we have described a solution for the conformance verification of careflow process executions. We have shown how a careflow model, defined through the GOSpeL graphical language, could be automatically translated to the $\mathcal{S}$CIFF language[4], based on computational logic and abductive logic programming, and how this formalization is then used by the proof-theoretic counterpart of the $\mathcal{S}$CIFF language to verify the conformance of a given careflow process execution w.r.t. the model. The feasibility of this approach has been tested on a cervical cancer screening protocol.

We plan to investigate in future work whether our approach can be extended to the workflow patterns discussed in other guideline support systems, like in [13]. Another ongoing work is about the proof of "high level" properties on the formalized guideline specification by using an extension of the $\mathcal{S}$CIFF proof procedure (named g-$\mathcal{S}$CIFF). For instance, given the IC_S representation of the above guideline fragment, we can ask to g-$\mathcal{S}$CIFF if a history exists s.t. a treatment invitation is sent to the patient. If this is the case, g-$\mathcal{S}$CIFF will produce a successful proof, generating the corresponding history.

Acknowledgments. 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 "TOCAI.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 paper.

References

1. Muir, G.: Evidence-based Healthcare. Churchill Livingston, London (1997)
2. Careflow management systems
`http://www.openclinical.org/briefingpaperStefanelli.html`
3. Chesani, F., Matteis, P.D., Mello, P., Montali, M., Storari, S.: A framework for defining and verifying clinical guidelines: A case study on cancer screening. In: Esposito, F., Raś, Z.W., Malerba, D., Semeraro, G. (eds.) ISMIS 2006. LNCS (LNAI), vol. 4203, pp. 338–343. Springer, Heidelberg (2006)
4. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Specification and verification of agent interactions using social integrity constraints. ENTCS 85(2) (2003)
5. Societies of computees (SOCS) Available at:
`http://lia.deis.unibo.it/Research/SOCS/`
6. Jaffar, J., Maher, M.: Constraint logic programming: a survey. Journal of Logic Programming 19-20, 503–582 (1994)
7. The SCIFF abductive proof procedure, Available at
`http://lia.deis.unibo.it/Research/sciff/`
8. Cervical cancer screening in emilia romagna (italy), Available at:
`http://www.regione.emilia-romagna.it/screening/`
9. SOCS-SI web site. Available at:
`http://www.lia.deis.unibo.it/research/socs_si/socs_si.shtml`
10. Terenziani, P., Montani, S., Bottrighi, A., Torchio, M., Molino, G., Correndo, G.: Applying artificial intelligence to clinical guidelines: The GLARE approach. In: AI*IA. vol. 101, pp. 536–547 (2003)
11. Fox, J., Johns, N., Rahmanzadeh, A.: Disseminating medical knowledge-the proforma approach. Artificial Intelligence in Medicine 14, 157–181 (1998)
12. Ciccarese, P., Caffi, E., Boiocchi, L., Quaglini, S., Stefanelli, M.: A guideline management system. In: MEDINFO 2004, pp. 28–32. IOS Press, Amsterdam (2004)
13. Mulyar, N.A., v.d. Aalst, W.M.P., Peleg, M.: A pattern-based analysis of clinical computer-interpretable guideline modelling languages. BPMcenter.org Technical Note (2006)