Abstract. In this paper, we propose a new computer-based approach to model clinical guidelines, adopting the agent-based paradigm. We first show how clinical guidelines can be modelled in an agent like fashion in the specification language Promela of the model checker SPIN. Then, we describe the impact of such a move: by using SPIN model-checking facilities, one can automatically prove a wide range of properties concerning the modeled guidelines. As a proof of concept, we apply such a methodology to the clinical guidelines in GLARE, a domain-independent prototypical system for acquiring, representing and executing clinical guidelines, which has been built within a 7-year project with Azienda Ospedaliera San Giovanni Battista in Turin (one of the largest hospitals in Italy).

1 INTRODUCTION

Clinical guidelines represent the current understanding of the best clinical practice, and constitute an important area of research in Artificial Intelligence (AI) in medicine and in medical decision making (see, e.g. [1]). Clinical guidelines may provide crucial advantages in individual-based health care, by supporting physicians in their decision making and diagnosing activities, both improving the quality of patient treatment and the efficiency of health-related organizations. Many different systems and projects have been developed in recent years in order to realize computer-assisted management of clinical guidelines (see e.g., Asbru [2], EON [3], GEM [4], GLARE [5],[6], GLIF [7], GUIDE [8], PROforma [9], and also [10], [1], [11]).

Agent technology has been rapidly developing in the last decade to answer the needs for new conceptual tools for modelling and developing complex software systems and it has given rise to a large amount of literature [12]. The agent-based approach has proved to provide crucial advantages, through the possibility of modelling complex systems and behaviors by modularly representing their components and the interactions between them [12]. As regards reasoning capabilities the model-checking approach has gained an important role in the AI community and it is widely used in the context of agent verification [13] as well as of planning [14].

In this paper we describe the activity on the specification and verification of clinical guidelines, that has been carried out in the context of the Italian project MIUR PRIN 2005 “Specification and verification of agent interaction protocols”. The project aims to define suitable logical formalisms for the specification and verification of agent interaction protocols and to develop techniques for automatic property verification and, more generally, for reasoning about communication protocols. One of the aims of the project is that of investigating the possibility of automatically translating modelling languages (including graphical ones) into the logic-based formalisms developed in the project. In this context, we aim at defining a comprehensive framework in which a computer-based approach to clinical guidelines is developed using an agent-based technology, and model checking is used in order to prove properties about guidelines and their applications on specific patients.

As a first step in this direction we have investigated how model checking techniques can be adopted in the verification of clinical guidelines and in this paper we report on our activity of verification of the medical guideline for ”stroke” (developed in collaboration with Azienda Ospedaliera San Giovanni Battista in Turin and already successfully tested in GLARE) with the model checker SPIN [15]. The guideline, as well as the different agents interacting with it, are modelled as communicating processes (agents) in the SPIN specification language Promela. Hence, as a side product, our approach also provides a clear declarative semantics to guidelines.

Though the experimentation is still ongoing, in the verification of the guideline we have been able to discover inconsistencies as well as sources of incompleteness. The paper discusses the advantages of the proposed approach, stressing that it allows an explicit representation of the interactions between the different agents/entities in the care environment (e.g., physicians, patients, clinical records), and it provide several reasoning capabilities, such as automatically checking the feasibility of a given (diagnostic and/or therapeutic) procedure under specific contextual conditions (e.g., given the resources available in a specific hospital) and/or for a given specific patient.

Although the methodology we propose is application-independent, in the following we show how we are implementing it on the basis of GLARE (GuideLine Acquisition, Representation and Execution).

2 CLINICAL GUIDELINES IN GLARE

GLARE (GuideLine Acquisition, Representation and Execution) is a domain-independent prototypical system for acquiring, representing and executing clinical guidelines which has been built within a 7-year project with Azienda Ospedaliera San Giovanni Battista in Turin (one of the largest hospitals in Italy), and has been successfully tested in different domains, including bladder cancer, reflux esophagitis, and heart failure.
2.1 GLARE Architecture

GLARE has been implemented relying on a three layered architecture, shown in figure 1.

The system is based on the assumption that knowledge in clinical guidelines is independent of its use (e.g., support, evaluation etc.), so that it is convenient (at least from the knowledge engineering point of view) to distinguish between the problem of acquiring and representing clinical guidelines and the problem of "using" them (e.g., "executing" acquired guidelines on specific patients). In accordance with this approach, at the highest layer in figure 1 it is possible to identify an acquisition and an execution modules. The acquisition module is a user-friendly and easy-to-use tool for acquiring clinical guidelines. In particular, it embeds:

1. a graphical interface, which supports primitives for drawing the control information in the guideline, and ad hoc windows to acquire the internal properties of the objects (see section 2.2); figure 2 shows part of a guideline as it appears in the acquisition tool interface;
2. facilities for browsing the guideline;
3. facilities for automatic consistency checking of temporal constraints (via the proposal of advanced AI techniques [16]).

The guidelines managed by the acquisition module are physically stored in the GL Database. On the other hand, the execution module executes an acquired guideline for a specific patient, taking into account the patient’s data, automatically retrieved from the Patient Database. The tool interacts with the user-physician via a user-friendly graphical interface as well. Advanced simulation, temporal-reasoning and decision-theory techniques are adopted in order to assist user-physicians in their decision-making activities.

The lowest layer of the architecture contains the DBMS, that physically stores the two databases described above, while the intermediate layer consists of a set of XML documents (one for each database). XML acts as an interlingua between the highest layer and the DBMS layer: the acquisition and execution modules actually interact only with the XML layer, through which they obtain the knowledge stored into the DMBS. The use of XML as an interlingua allows us to express the guidelines in a format with characteristics of legibility, and to publish them on the web, rendering easy their dissemination. On the other hand, the DBMS layer grants a homogeneous management of the data, by integrating the guideline representation with the pre-existent hospital information system in the same physical DBMS.

2.2 Representation Language

In order to guarantee usability to physicians not expert in Computer Science, in GLARE we have defined a limited set of clear representation primitives [5]. In particular, we have focused our attention on the concept of action, distinguishing between atomic and composite actions. Atomic actions can be regarded as elementary steps, in the sense that they do not need a further de-composition into sub-actions to be executed. Composite actions are composed by other actions (atomic or composite).

GLARE distinguishes between four different types of atomic actions: work actions, query actions, decisions and conclusions. Work actions are atomic actions which must be executed at a given point of the guideline, and which can be described in terms of a set of attributes, such as name, (textual) description, cost, time, resources, goals. Query actions are requests of information, that can be obtained from the outside world (physicians, databases, knowledge bases). Decision actions are specific types of actions embodying the criteria which can be used to select among alternative paths in a guideline. In particular, diagnostic decisions are represented as an open set of triples \(<\text{diagnosis}, \text{parameter}, \text{score}>\) (where, in turn, a parameter is a triple \(<\text{data}, \text{attribute}, \text{value}>>\), plus a threshold to be compared with the different diagnoses’ scores. On the other hand, therapeutic decisions are based on a pre-defined set of parameters: effectiveness, cost, side-effects, compliance, duration. Finally, conclusions represent the explicit output of a decision process.

Composite actions are defined in terms of their components, via the has-part relation (this supports for top-down refinement in the description of guidelines). On the other hand, a set of control relations establish which actions might be executed next and in what order. We distinguish among four different control relations: sequence, concurrent, alternative and repetition.

Each of the representation primitives listed above has an underlying intended semantics (which can be explicitly provided by means of model checking).

As an example, the intended semantics of a query action is the following.

Example: query action. A query action requires a set of data, which are obtained by means of the interaction of the physician who is executing the guideline with additional “agents”. For the sake of clarity, we will concentrate on a single datum, and we will reduce the set of additional agents involved to the database, in which the required datum may be stored, and a second agent (Outside henceforth), that represents the rest of the outside world (e.g. a laboratory where tests are conducted by means of bio-medical instrumentations, or the patient herself, if the datum can be collected during an interview). The datum is a quadruple \(<D, A, V, t>\), where D is the category to which it belongs (e.g. liver objective examination), A is the attribute of interest (i.e. volume of the liver), V is the value assumed by the attribute in the given case, and t is the time at which the measurement was taken.

The datum required by the query action is first searched for in the database (therefore an interaction between the guideline and the Database agents takes place). If the datum is found, the Physician agent evaluates if it is still reliable (i.e. t is not too old). In this case, the query action is completed, since the datum is available and valid. Otherwise, a second interaction between the guideline and the Outside agents is carried out, in order to obtain a (more recent) version of the datum from the competent source. The datum provision by Outside concludes the query action.

The parts in the dashed box in figure 1 represent the extensions needed to incorporate a verification component based on the model-checking approach, and will be described in section 3.4.
3 SPECIFICATION AND VERIFICATION OF CLINICAL GUIDELINES

In this section, first we describe how GLARE architecture is extended to deal with verification using SPIN, then we describe how a guideline in GLARE can be mapped to a Promela specification and, finally, we describe the types of verification which can be performed on the guideline and provide some verification examples.

3.1 Architecture of the Overall Approach

Let us now describe the part of the system architecture which is contained in the dashed box in Figure 1. In the model checking approach [17], given a model describing all the possible evolutions of the system and a specification expressed in a temporal logic, the model is checked to see whether it satisfies the specification. In particular, in the model checker SPIN the specification of the model is given in the input language Promela (that will be shortly described in Section 3.2), and the specification (the property to be checked) is a formula of the linear time temporal logic (LTL). The fact that the model checking approach has been successfully used for protocol verification motivates the interest of this approach in the verification of clinical guidelines, which can be regarded as special kinds of protocols.

The translation of the guideline to Promela is performed starting from its XML specification in GLARE and we refer to section 3.2 for a description of how the guideline can be mapped to a Promela specification. A similar translation is applied to the patient data. The behaviour of the external environment, of the Database and of the physician are all modelled as Promela processes. SPIN translates each process (each agent) into a finite automaton, and the global behaviour of the system is obtained by computing an asynchronous interleaving product of automata. The resulting automaton represents the global state space of the system (the model containing all the possible executions of the guideline) and can be built on-the-fly during the verification process.

A property which has to be verified on the system is passed to the verifier through an interface, which maps it into a temporal formula, as required by SPIN. SPIN converts the negation of the temporal formula into a Büchi automaton and computes its synchronous product with the system global state space. If the language of the resulting Büchi automaton is empty then the property is true on all the possible execution of the system, otherwise the verifier provides a counterexample for the property (an execution path on which it is false).

3.2 Representing GLARE Clinical Guidelines in Promela

The model checker SPIN is used in the verification of clinical guidelines by modelling the guideline as well as all the agents interacting with it as processes in the specification language Promela. Promela allows a high level model of a distributed system to be defined by modelling each agent in an extended pseudo C code, including synchronization primitives and message exchange primitives.

SPIN generates an optimized on-the-fly verification program from the high level specification of the system and it allows the verification of correctness claims that are specified as temporal logic formulas.

Before describing how the guideline is encoded in Promela, let us point out the underlying hypotheses. First of all, the stroke guideline only defines qualitative constraints on the temporal ordering of actions. More precisely the actions can be executed sequentially or concurrently, and the evaluation of temporal constraints is not required during the execution of the guideline. For this reason the timestamps associated with the data (describing the time at which the measurement of each datum was taken) are not used in constraints to be evaluated during the execution of the guideline. They are only used by the Physician who has to decide if they are still reliable. As we model the Physician as a nondeterministic process answering the requests from the guideline, we do not need to have the time values available during the computation and hence to represent them in the model. Hence, we only represent the fact that the timestamps are exchanged between the guidelines and other agents, but we will only give symbolic (constant) value to them, to mean that the time value is known at that point of the computation.

As anticipated in the section 2.2, the model of the system is described by specifying the following agents as Promela processes: the Guideline agent; the Physician agent; the Outside agent; the Database agent.

The Physician agent is modelled as a nondeterministic process which interacts with the guideline by evaluating the patient data, choosing among the different alternative feasible paths and deciding among the different "backtracking" alternatives in the case of action failure.

The Outside agent, representing the outside world, provides up to date values for data (together with the time of their measurement) when they are not already available from the database. It also stores data in the database, executes work actions and reports about their success or failure.

The Database agent models the behaviour of the patient database, allowing for data insertion and retrieval.

The Guideline agent models the overall behaviour of the guideline. Each construct in the guideline is mapped to a Promela statement or (for complex statements) to a Promela piece of code. For instance, each work action is mapped to a conditional statement in which the action preconditions are evaluated and, in the case they hold, the action is sent to the Outside agent for execution. The sequence relation and the concurrent relation are mapped to the sequence and to the parallel composition constructs in Promela. Decisions are mapped to a sequence of send operations which inform the physician about the supported paths, followed by a selection construct which allows the selection of the path of the guideline to be followed according to the physician decision. As a concrete example, the following Promela code models the executing a Query action, which has been described at the end of section 2.

A: LGtoDB!data[0].D,data[0].A;
LGfromDB?data[0].D,data[0].A,data[0].V,
Properties concerning a guideline “per se”. One can check if the guideline contains a path of actions satisfying a given set of properties (e.g., a path including actions X, Y and Z, or a path in which no action of type X is executed, or a path nor requiring a given laboratory test, or a path requiring only a given set of resources, and so on).

2. Properties of a guideline in a given context. Specific contexts of execution may impose several limitations on the executable actions of guidelines, related, e.g., to the lack of certain resources (e.g., laboratory instruments). The consequences of such limitations may be automatically investigated taking advantage of the model checker. For instance, the model checker can prove whether there is or not a therapy for a patient affected by a given disease, in the case a specific set of resources is available (not available).

3. Properties of a guideline when applied to a specific patient. Provided that the model checker can in input all the data in the patient record, the feasibility of a given action or path of actions on the specific patient can be proved.

4. Integrated proofs. Of course, given the flexibility and the task-independence of the model checker, any combination of the above types of proofs is feasible. For instance, one may ask whether, given a patient with a specific disease and set of symptoms, and given a hospital with a specific set of resources, there is a path in the guideline which applies to that patient and satisfies a given set of properties.

During the verification of the stroke guideline we have been able to discover some inconsistencies in the original formulation. In particular, we wanted to check that, if a recovery treatment has been excluded at some point of the guideline, later on the guideline does not prescribe a continuation of the recovery treatment. This property happens to be false. The property to be verified has been represented with the following LTL formula:

$$
\Box(conclusion == \neg\text{recovery\_treatment\_excluded})
$$

SPIN produces a counterexample to this property.

As an example of property concerning the contextualization of the guideline to a given hospital, let us suppose that the angiography is not available in the hospital. We want to check if the angiography is eventually required on every execution of the guideline or if there exist some execution on which it not required. We need to verify the following LTL formula:

$$
\Diamond(required\_test == \neg\text{angiography})
$$

This property is not true on all the runs of the stroke guideline and a counterexample is returned from the model checker.

To conclude, it is worth stressing that, with the exception of recent proposals, that will be referred in the related work, the verification capabilities mentioned above are not usually available in the GL systems in the literature. In many cases, in fact, such systems do associate only very specific and limited inferential mechanisms to the knowledge represented in the guideline, so that, in some sense, such a knowledge is “passive” and no property can be proved unless a specific software application is built to do so. On the contrary, in our approach, we model GL in Promela formalism which is automatically manipulated by the model checker; thus the model checker, which is a general-purpose system, can automatically prove properties (expressed in the model-checker language), on the basis of such a knowledge (which, in some sense, has now an “active” role in the overall framework).

4 RELATED WORK AND CONCLUSIONS

In this paper we have proposed a model checking approach for the verification of clinical guidelines. Clinical guidelines are translated
to the specification language Promela of the model checker SPIN and the verification of properties of the guidelines can be done by formulating the property to be checked in the temporal logic LTL. Although the experimentation is still ongoing, the automatic verification of properties on the stroke guideline has been able to discover some inconsistencies in the guideline that had not been previously detected by hand. Also, we have been able to contextualize the guideline to specific contexts and specific patients and to prove properties about them.

In [18, 19] a theorem proving approach is proposed to deal with the problem of protocol verification. A medical protocol is modelled in the Asbru language as a hierarchical plan and then it is mapped to a specification in KIV, an interactive theorem prover for higher order logic. Properties are expressed in a variant of Interval Temporal Logic. In particular, [19] provides an evaluation of the feasibility of the approach based on the formalization and verification of two real-life medical protocols, namely the “jaundice” protocol and the “diabetes mellitus” protocol.

[20] describes how efficient model checking techniques can be used in the verification of the guidelines properties. The Asbru model is translated into the input language of SMV model checker by making use of a suitable abstraction which eliminates time. Structural and medical properties of the jaundice protocol are formalized as ACTL formulas (that is CTL formulas only allowing universal path quantifiers). From several points of view our approach is similar to this last one, although the adoption of different model checking tools allows for the verification of slightly different properties (as ACTL and LTL are incomparable).

Although there is a wide agreement about the importance of providing a clear semantic model for clinical guidelines, this issue has been faced in several quite different ways within the medical informatics community. In most cases, the semantics of guidelines has been only implicitly provided via an execution engine, which provides an interpretation of guidelines by executing them on specific patients. Considering explicit representations, a formal operational semantics has been provided for PROforma [21], via the definition of an abstract execution engine and of rules describing how the different guideline operations change the state of such an engine. On the other hand, in SAGE a mapping to standard terminologies and for their constant support and supervision in the development of the GLARE system.

REFERENCES


