Modeling clinical guidelines through Petri Nets

M. Beccuti, A. Bottrighi, G. Franceschinis, S. Montani, and P. Terenziani

DI, Univ. Piemonte Orientale "A. Avogadro", Via Bellini 25/g, Alessandria, Italy
{beccuti, alessio, giuliana, stefania,terenz}@mfn.unipmn.it

Abstract. Clinical guidelines (GLs) play an important role to standardize and organize clinical processes according to evidence-based medicine. Several computer-based GL representation languages have been defined, usually focusing on expressiveness and/or on user-friendliness. In many cases, the interpretation of some constructs in such languages is quite unclear. Only recently researchers have started to provide a formal semantics for some of such languages, thus providing an unambiguous specification for implementers, and a formal ground in which different approaches can be compared, and verification techniques can be applied. Petri Nets are a natural candidate formalism to cope with GL semantics, since they are explicitly geared towards the representation of processes, and are paired with powerful verification mechanisms. We show how Petri Nets can cope with the semantics of GLs in a clear way, taking the system GLARE formalism as a case study.

Key words: clinical guidelines, Petri net, Well-formed net.

1 Introduction

The adoption of clinical guidelines (GLs), by supporting physicians in their decision making and diagnosing activities, may provide crucial advantages, both in individual-based health care, and in the overall service offered by a health care organization. Thus, several systems and projects have been developed in recent years, to realize computer-assisted GL management (see e.g., the collections [1, 2]), and each system has been grounded on the definition of a proper GL representation language. The main goals of such languages are usually expressiveness and/or user-friendliness. However, in many cases, the interpretation of some constructs in these languages remains quite unclear, and/or is hidden in the code of the execution engine. As a consequence, today a wide agreement has been reached within the scientific community about the importance of pairing each GL representation language with a rigorous and formal description of its meaning, i.e., with a formal semantics [2]. While GL representation formalisms are used as the user-friendly interfaces to physicians, their formal semantics, due to their intrinsic technical complexity, are usually hidden to users. Nevertheless, they still play very important roles in the GL specification context. As a matter of fact, a semantic model allows one to provide a clear interpretation of the representation language, and guarantees that any operation performed on a GL

has a precisely defined and unambiguous effect. Moreover, it also gives birth to a formal common ground on which different approaches can be compared [3], assessing what each representation can and cannot capture[1]. Additionally, the frameworks which can be used to provide a semantic interpretation of GLs are often coupled with verification techniques, which can be employed for discovering logical inconsistencies in a GL, or for proving particular properties it exhibits.

Despite its importance, the issue of copying with GL semantics has been faced only recently within the medical informatics community (see e.g. [3, 5] and section 3). Moreover, existing works address the problem of representing GL primitives, but do not take into account the GL execution environment. On the other hand, in order to realistically capture the semantics of GL execution, we believe that the GL cannot be intended as an isolated process, but as one of a set of interacting processes, which also describe the behavior of additional agents (e.g. physicians, databases, labs), involved in patient care.

In this paper, we identify such processes, and describe the characteristics of their interaction. Moreover, we model the GL and GL-related processes semantics adopting the theory of Petri Nets (PNs). PNs [6] and their extensions are a family of formalisms which are well suited for modeling Discrete Event Dynamic Systems, and are explicitly geared towards the representation of interacting processes. Therefore, they are a natural candidate to cope with GL semantics in a natural and easy-to-understand way. Moreover PNs are suited to support optimal resource allocation as well as formal verification.

In the following we will consider as a reference the GLARE approach to GL representation and execution [7]. However, it is worth stressing that the methodology we propose is mostly application-independent.

2 Representing Guidelines as Petri Nets

PNs are bipartite directed graphs with two types of nodes: **place** and **transition**. The places, graphically represented as circles, correspond to the state variables of the system, while the transitions, graphically represented as boxes, correspond to the events that can induce a state change. The arcs connecting places to transitions and vice versa express the relation between states and event occurrence. Places can contain tokens drawn as black dots within places. The state of a PN, called "marking", is defined by the number of tokens in each place. The evolution of the system is given by the firing of an enabled transition [2], which removes a fixed number of tokens from its input places and adds a fixed number of tokens into its output places (according to the cardinality of its input/output arcs).

[1] [4] has made a first step towards such a comparison, but it was limited to a review of syntactic features of the representations, without considering execution semantics.

[2] A transition is enabled iff each input place contains a number of tokens greater or equal than a given threshold, and each inhibitor place contains a number of tokens strictly smaller than a given threshold. Such thresholds are defined by the cardinality of its input/inhibitor arcs [6].

In particular, in our work we use the Well-formed Net (WN) formalism, which extends the PN formalism with "colour" [8]. Its main feature is the possibility of having distinguished tokens, which can be graphically represented as dots of different colours: the colour attached to a token carries some kind of information (see the Clinical Database example below). This formalism provides two advantages: a more compact and readable representation of the system, and the possibility of using efficient solution techniques [8]. WN submodel can be composed by a composition operator [9]. The composition operator is based on the known concept of "matching labels": transitions and places are labelled and pairs of transitions (or places) with matching labels are superposed. In this paper, the labels are encoded in the transition/place name as *"Name|label"*.

In literature the majority of the GL representation languages share a set of abstract primitives[3]. These primitives can be divided in action primitives (i.e. **atomic** and **composite** actions), and control flow relation. The atomic action types are *work* actions, *query* actions and *decisions*. Work actions are atomic actions which must be executed at a given point of the GL. Query actions represent explicit or implicit requests of information, that can be obtained from a database. Decision actions embody the criteria which can be used to select among alternative paths in a GL. The composite actions are defined in terms of their components (i.e. atomic and/or composite actions) via the *part-of relation*. The control flow relations (i.e. *sequence*, *repetition*, *parallelism*) establish which actions might be executed next and in what order. It is natural to use compositional approach, so that the GLs is modelled as set of WN submodels that will be composed by the composition operator.

The WN models corresponding to atomic actions are shown in Fig. 1. In particular Fig. 1A shows how the decision action is modeled. The transition $BeginD_i|BD_i$ represents the starting of a decision process, which ends when the firing of exactly one transition $End_i|ED_i$ occurs. Observe that the transitions $End_i|ED_i$ are enabled concurrently and represent the alternative feasible paths that can be taken. The place $InputD_i|D_i$ and the place $OutputD_i|D_{i+1}$ represent the input and output of the decision process. Fig. 1B shows how the work action is modeled. Here, there is only the transition $ActionA_i|AA_i$ representing the execution of the work action. Similarly to decisions, the place $InputA_i|A_i$ and the place $OutputA_i|A_{i+1}$ represent the input and output of the work action. Fig. 1C shows how the query action is modeled. In this model, there are two transitions: $BeginQ_i|RD_i$ and $EndQ_i|AK_i$, which represent the start and the end of the data request process, respectively. Then, the places $InputQ_i|Q_i$ and $OutputQ_i|Q_{i+1}$ represent the input and output of the query action. In order to obtain the overall GL model we translate every action in the corresponding WN model and combine all these models according to the control relations specified in the GL. For instance in the case of a sequence of actions, the composition is done by superposition between the output place of the first action and the input place of the next one.

[3] Despite the generality of common concept in GL formalism, in this paper we will consider as reference the GLARE formalism.

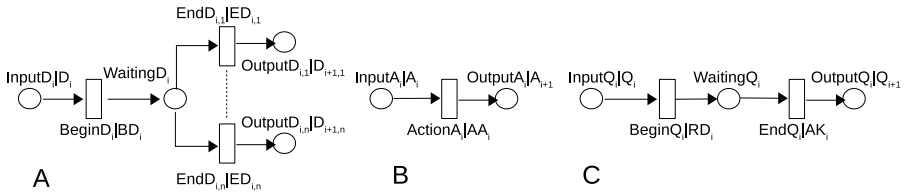

Fig. 1. The three WN models that represent the atomic actions: (A) the decision action, (B) the work action and (C) the query action

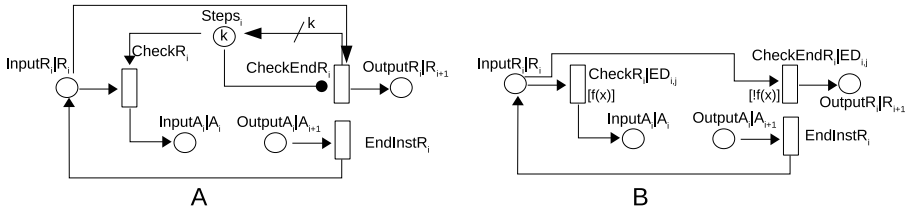

Fig. 2. The WN models representing the ways of specifying repetitions: (A) with fixed number of repetitions, (B) with exit condition

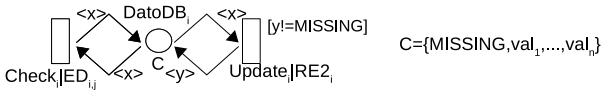

Fig. 3. The WN model describing a single datum in the clinical database

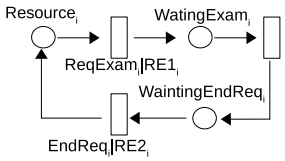

Fig. 4. The WN model describing a outside environment

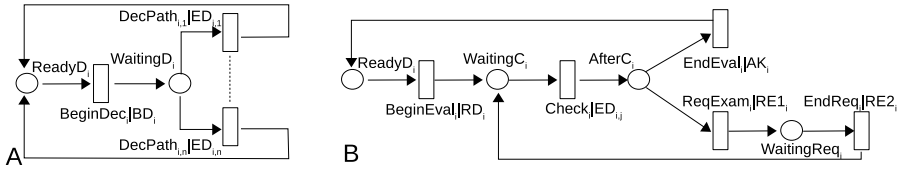

Fig. 5. The WN models describing the Physician tasks: (A) the decision process, (B) the data evaluation process

On the other hand, repetitions are managed according to two different semantics: (A) the action has to be performed a *fixed number of times*; (B) the action has to be performed until a given *exit condition* (defined on the patient data) becomes true. The WN models corresponding to these two different types of repetition are showed in Fig. 2. In the two WN models the places $InputR_i|R_i$ and $OuputR_i|R_{i+1}$ represent the input and the output of the repetition process, while the places $InputA_i|A_i$ and $OutputA_i|A_{i+1}$ represent the input and output of the actions which should be repeated.

In the former model (A), the initial marking of the place $Steps_i$ contains k tokens (graphically a k inside the place circle), that represent how many times the action has to be repeated. If there is at least a token in $Step_i$, the transition $CheckR_i$ is enabled to fire and a new instance of the action will be executed; otherwise the $CheckEndR_i$ can fire ending the repetition process and inserting k tokens in the place $Steps_i$ (this is graphically represented by a label k associated with the arc connecting $CheckEndR_i$ to $Steps_i$). Observe that the inhibitor arc (depicted as circle-headed arc) connecting $Steps_i$ to $CheckEndR_i$ assures that $CheckEndR_i$ can fire only when no tokens are in $Steps_i$.

In the latter model (B), the firing of transition $CheckR_i|ED_{i,j}$ and $CheckEndR_i|ED_{i,j}$ depend on the values of patient data (evaluated by function f(x)). It is worth noting that the two transitions cannot be enabled at the same time, because the guard of transition $CheckR_i$ is $f(x)$ and the guard of other transition is its negation [8].

In order to model and simulate GL execution on a real, specific patient, the representation of the GL *per se* is not enough: patient's characteristics need to be specified. We characterize a patient by relying on her data, which are typically maintained in the clinical database. Thus, GL execution requires the representation of the clinical database as well, interpreted as a "service" from which data can be queried, and in which updated data values can be inserted.

Updated data values are sometimes obtained from additional sources (e.g. from the hospital laboratory service). We have generically modeled such sources and services by means of a further submodel, called *outside* world.

Last but not least, GL execution is performed by a physician; therefore, the physician's behaviour needs to be modeled as well. In particular, we have identified two main tasks that the physician is expected to cover when applying a GL to a specific patient. Obviously, she is required to make decisions, i.e. she has to select exactly one diagnosis or therapy, among a set of alternative ones. In order to be as accurate and realistic as possible, we have also modeled a second task, which is the evaluation of data recency and reliability. If a data value, extracted from the database, is judged as unreliable, or not up-to-date (i.e. too old), the physician has to signal the problem, thus triggering the generation of newer data value from the outside world.

The *Clinical Database*, *Outside Environment* and *Physician* submodels representation is addressed below.

Clinical Database Net. The Clinical Database Net is represented by a set of WNs (i.e. one for each modeled datum in the database). Each WN is com-

posed by a unique coloured place $DatoDB_i$ and two transitions $Update_i|UP_i$ and $Check_i|ED_{i,j}$ as shown in Fig. 3. The domain associated with the coloured place (e.g. the colour class "C" in Fig. 3) represents the possible (discrete) values that the datum can assume (e.g. if the datum represents the patient temperature then the possible values in its domain could be *very high*, *high*, *normal* and *low*), so that this place can contain only tokens with colours belonging to this domain. Among the possible values associated with the tokens in $DatoDB_i$, there is always a special value, called $MISSING$, representing that no information about the datum is stored in the database. The transition $Update_i|UP_i$ has as input and output the place $DatoDB_i$ and models the update process of the datum. The use of different labels $\langle x \rangle$ and $\langle y \rangle$ on the transition input/output arcs models the fact that the new stored datum value will be selected among all the possible values in the domain. Moreover the transition guard assures that the new value can not be $MISSING$. Instead the transition $Check_i|ED_{i,j}$ models the retrieval process. The same label $\langle x \rangle$ on its input/output arcs models that the datum will not change due to the retrieval process itself.

Outside Environment Net. This WN model describes how the outside environment performs the update process of a datum, which can be required by the physician if she thinks that the current value is unreliable, or if the datum is missing. In fact the transition $ReqExam_i|RE1_i$ models the start of the update process, while the transition $EndReq_i|RE2_i$ models its end corresponding with the database update. The transition $Exam_i$ represents the execution of the required datum generation activity. Observe that the place $Resource_i$ represents the possibility of performing the required datum generation activity (e.g. the laboratory is or is not busy).

Physician Net. The Physician Net is represented by a set of WN models corresponding to the decision and to the data evaluation processes. The decision process describes how the physician takes a decision about alternative possible paths, while the data evaluation process describes how the physician decides whether a datum is reliable. In Fig. 5A the physician decision process is modeled. Note that this net is very similar to the net in Fig. 1A modeling the decision in the GL. The transition $BeginDec_i|BD_i$ represents the start of the decision process, which ends when one transition $DecPath_{i,j}|ED_{i,j}$ fires. The firing of a transition $DecPath_{i,j}|ED_{i,j}$ corresponds to the physician choice of one path; after that she becomes ready for another decision. In Fig. 5B the physician data evaluation process is modeled. The evaluation process starts with the firing of the transition $BeginEval_i|RD_i$ and ends when the transition $EndEval_i|AK_i$ fires (i.e. the physician decides that such datum is reliable). The transition $Check_i|ED_{i,j}$ represents the datum retrieval, while the free choice between the transitions $EndEval_i|AK_i$ and $ReqExam_i|RE1_1$ models the physician choice about the datum reliability. If the datum is judged as not reliable (or is missing) then the transition $ReqExam_i|RE1_i$ fires and the update process starts.

Net composition. The overall net, modeling the execution of a GL on a specific patient, is obtained by the composition of the previous WN submodels by superposition over transitions with the same label. In order to clarify how this

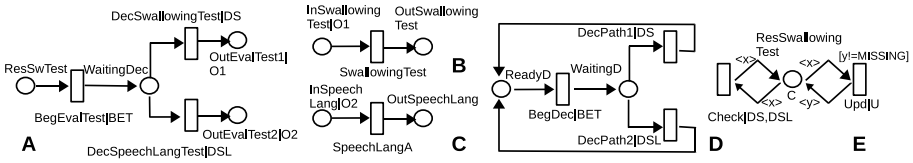

Fig. 6. The submodels involved in the composition phase

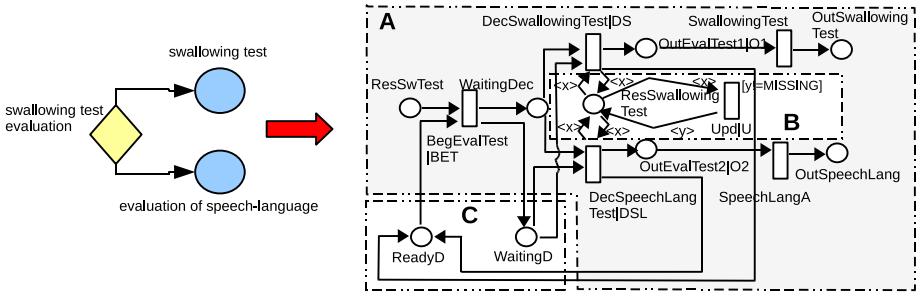

Fig. 7. The GLARE model representing the example and the corresponding composed WN model.

composition is performed we provide an example, taken from the representation of an ischemic stroke GL, developed at Azienda Ospedaliera S. Giovanni Battista in Turin, Italy.

Example: we model the following decision action: the result of a swallowing test evaluation. If the physician considers the test result as negative, action "swallowing test" is performed, otherwise action "evaluation of speech-language" is executed.

In Fig. 6 all the submodels that are involved in the composition phase are shown; models A, B, C are related to the GL net, while D and E are related to the Physician and Clinical Database net.

In order to obtain the global net, shown in Fig. 7, two composition steps are necessary. The first step composes the decision action (Fig. 6A) with the work actions *swallowing test* and *evaluation of speech-language* (Figs. 6B and 6C), so that such composed net (Fig. 7 dashed box A) represents the GL model. The composition is performed by the superposition over the places $OutEvalTest1|O1$ and $InSwallowingTest|O1$, and $OutEvalTest2|O2$ and $InSpeechLang|O2$. After that, the second composition step merges the GL model (Fig. 7 dashed box A), the Physician decision model (Fig. 6D) and the Clinical Database model (Fig. 6E) by superposition over the transitions: $BegEvalTest|BET$ and $BegDec|BET$, $DecSwallowingTest|DS$, $DecPath1|DS$ and $Check|DS, DSL$, and $DecSpeechLangTest|DSL$, $DecPath2|DSL$ and $Check|DS, DSL$.

In our current implementation, the translation of GL (expressed in the GLARE formalism) into the WN submodels is performed in two steps: first the GL stored in a XML file is translated into a set of WN submodels according to the above

rules; then the submodel are composed in a unique WN model by means of the Algebra tool belonging to the GreatSPN suite [10].

3 Related work

Although today there is a wide agreement about the importance of providing a clear semantic model for GLs, this issue has been faced only recently within the medical informatics community, and in several quite different ways. In most cases, the semantics of GLs have been only implicitly provided via an execution engine, which allows an interpretation of GLs by executing them on specific patients. Considering explicit representations, a formal operational semantics has been provided for PROforma [3] via the definition of an abstract execution engine and of rules describing how the different GL operations change the state of such an engine. On the other hand, in SAGE a mapping to standard terminologies and models (such as the virtual medical record) is advocated [11]. While the Asbru protocol representation language allows the semantics of GLs to be defined through Asbru formal semantics [5], a logical semantics to GLs has been provided in [12]. There, a graphical notation to express GLs is introduced, which can be automatically translated to the logic-based formalism provided by the SOCS computational logic framework.

We believe that our choice of relying on PNs allows us to describe GL semantics in a more natural fashion with respect to other semantic formalisms, since the mapping from GLs and GL-related processes interactions to PNs is rather straightforward. As a consequence, the output of the formalization process is easier to understand also for physicians, with respect to e.g. temporal logics.

A couple of other groups have already shown interest towards the adoption of PNs for GL representation. Quaglini's group [13], in particular, has built the system GUIDE on top of enterprise workflow standards and tools. In GUIDE, each acquired GL is translated into the Workflow Process Definition Language (WPDL), whose code is then used to build a PN. A model of the healthcare organization is also exploited to represent knowledge about available resources. A proper (commercial) software package then takes the PN and the organization model in input, and simulates the implementation of the GL in the clinical setting, in order to suggest the optimal resources allocation before the overall system is installed. Simulation enables to calculate e.g. at which time certain resources have high or low loads, what are the system bottlenecks, what are the costs of the different patients in the different stages of the GL execution, etc. Also Peleg's group has worked on the topic, studying the possibility of representing GLs as well as other complex biological processes [14] by means of PNs. They map instances of the GLIF ontology to the reference model of the Workflow Management Coalition using Protege ontology mapping rules. As in [13], then they further map this model on PNs for verification of structural properties (of biological systems) and for studying the system behaviour by simulation (for both biological systems and GLs). Unlike [13] they disregard GL resources, concentrating only on the control flow among activities.

With respect to these approaches, we perform a direct translation from GL primitives to PNs, without resorting to intermediate layers (namely, to WPDL). Moreover, and more interestingly, we do not model just the GL process, but also the behaviour of other processes involved in its execution, namely physicians, databases, etc., providing a more comprehensive view of the implementation of a GL in clinical practice. Actually, we believe that the interactions among the set of processes involved in its execution have to be properly captured, since GL semantics depend on the context in which the GL itself is meant to be applied. A more comprehensive description of the GL and of its execution environment also allows to obtain more meaningful performance indications, and to optimize resource allocation, two tasks towards which PNs are naturally very well suited.

Finally in [15] the authors propose a similar approach based on a directly translation of GL expressed in PROforma in Coloured Petri Net (CPN). We have to highlight that WN formalism used in our approach is a particular kind of CPN, that thanks to a very structured syntax for the denition of the place and transition color domains and of the arc functions and the transition guards gives the possibility to define several efficient analysis methods exploiting the intrinsic symmetries of the model. This efficient analysis methods will be very helpful when we will model a real healthcare organization instead of the executing a single GL on a single patient.

4 Conclusions

In this paper, we have afforded the problem of providing a formal semantic interpretation of GLs. In particular, having observed that GL execution is a complex phenomenon that cannot be modeled just by representing the GL *per se*, we have introduced a more comprehensive way of capturing the GL dynamics and of its execution environment, based on the idea of representing a set of processes, whose interaction models in a more realistic way the GL execution itself. PNs have thus appeared as a natural candidate to represent such environment.

Of course, a PN-based approach also allows for performance analysis and resource allocation optimization. This facility can become even more helpful by shifting the perspective from the one of executing a single GL on a single patient, to the one of dealing a real healthcare setting, in which different agents (physicians, nurses, labs) cooperate, and several, different GLs have to be executed, in order to care a set of patients. We plan to follow this direction as a future work, thus extending the approach in [13].

Moreover, PNs can be employed to support formal GL verification (i.e. for discovering logical inconsistencies in the GL, or for proving particular properties it exhibits, see chapter 4 in [2]. The use of PN in model checking (instead of other, logic-based formalisms) would provide a more easily interpretable output to end users. Additionally, PN can be easily interfaced with SPOT [16], a model checking library which relies on Transition-based Generalized Büchi Automata, allowing more compact translations of LTL formulas with respect to traditional approaches (e.g. SPIN), and which exploits global symmetries of the system,

thus speeding up computation. In the future, we plan to investigate PN-based GL verification as well, and to complete the integration (and testing) of our approach within the GLARE system.

References

1. D.B. Fridsma (Guest Ed.). Special issue on workflow management and clinical guidelines. *Journal of the American Medical Informatics Association*, 1(22):1–80, 2001.
2. A. ten Teije, S. Miksch and P. Lucas, editors. *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends*. IOS Press, Amsterdam, 2008.
3. D.R. Sutton and J. Fox. The syntax and semantics of the PROforma guideline modeling language. *Journal of the American Medical Informatics Association*, 10:433–443, 2003.
4. M. Peleg and al. Comparing models of decision and action for guideline-based decision support: a case-study approach. *Journal of the American Medical Informatics Association*, 10:52–68, 2003.
5. M. Balser, C. Duelli, and W. Reif. Formal semantics of asbru - an overview. In *Proc. IDPT*, 2002.
6. J.L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
7. P.Terenziani, S. Montani, A. Bottrighi, G. Molino, and M. Torchio. Applying artificial intelligence to clinical guidelines: the glare approach. In A. TenTeije, S. Miksch, and P. Lucas, editors, *Computer-based medical guidelines and protocols: A primer and current trends*. IOS Press, Amsterdam, 2008.
8. G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic Well-formed Coloured nets for symmetric modelling applications. *IEEE Transactions on Computers*, 42:(11): 1343 – 1360, 1993.
9. S. Bernardi, S. Donatelli, and A. Horvath. Implementing compositionality for stochastic petri nets. *International Journal on Software Tools for Technology Transfer*, 3(4), 2001.
10. S. Baarir, M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, and G. Franceschinis. The GreatSPN Tool: Recent Enhancements. *ACM Performance Evaluation Review Spec.Issue on Tools for Perf.Eval.*, 36:(4): 4–9, 2009.
11. C.G. Parker, R.A. Rocha, J.R. Campbell, S.W. Tu, and S.M. Huff. Detailed clinical models for sharable, executable guidelines. In *Proc. Medinfo*, pages 45–148, 2004.
12. M.Alberti, A.Ciampolini, F.Chesani, M.Gavanelli, P.Mello, M.Montali, S.Storari, and P.Torroni. Protocol specification and verification using computational logic. In *Proc. WOA*, 2005.
13. S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso, and S. Panzarasa. Flexible guideline-based patient careflow systems. *Artificial Intelligence in Medicine*, 22:65–80, 2001.
14. M. Peleg, D. Rubin, and R.B. Altman. Using petri nets tools to study propertuies and dynamics of bilogical systems. *Journal of the American Medical Informatics Association*, 12:181–199, 2005.
15. M.A. Grando, D. W. Glasspool and J. Fox. Petri Nets as a formalism for comparing expressiveness of workflow-based Clinical Guideline Languages. In *Proc. PROHealth08. Springer-Verlag. Lecture Notes in Computer Science series*, 2008.
16. A. Duret-Lutz and D. Poitrenaud. SPOT: an Extensible Model Checking Library Using Transition-Based Generalized Büchi Automata.