Reusing a Declarative Specification to Check the Conformance of Different CIGs

M.A. Grando[1], Wil M.P. van der Aalst[2], and Ronny S. Mans[2]

[1] Division Biomedical Informatics, University of California San Diego, USA
mgrando@ucsd.edu
[2] Department of Information Systems, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
{w.m.p.v.d.aalst,r.s.mans}@tue.nl

Abstract. Several Computer Interpretable Guidelines (CIGs) languages have been proposed by the health community. Even though these CIG languages share common ideas each language has to be provided with his own mechanism of verification. In an earlier work we have shown that a DECLARE model can be used for checking the conformance of a PRO*forma* CIG. In this paper, we show that the same model can also be used for checking the conformance of a similar CIG expressed in the GLIF language. Besides, as the GLIF model has been expressed in terms of a Coloured Petri Net (CPN), we also elaborate on the experiences obtained when applying the model checking techniques supported by CPN tools.

Keywords: clinical guidelines, conformance checking, Petri Nets.

1 Introduction

Checking the compliance of a medical application to policies and guidelines [1], the level of adherence of clinicians with respect to the intentions of guideline authors [2] and critiquing systems by comparing actions performed by the physicians with predefined set of actions [3] are important problems. All these problems share in common that they require the use of verification techniques.

The spectrum of verification techniques used so far is broad, mainly based on: algorithms [1,2], conformance checking [4,5], model checking [3,6,7] and theorem proving [8]. But in the mentioned works the proposed verification techniques have been designed having in mind a specific CIG language and are most probably not easily reusable for checking properties in CIGs defined in languages different from the one that inspired the methodology development.

Here we consider the problem of checking the compliance of policies and guidelines which could help to reduce medical errors by detecting inconsistencies, errors of interpretation or incompleteness of an application with respect to the recommendations on which it is based. We continue the research presented in [5] based on *a semantic-based approach that is fully independent of the language used for the specification of the CIG*. By a combination of ontology matching and process mining the same set of declarative specifications of medical recommendations can be checked in an arbitrary CIG, providing a generic and reusable verification methodology. Given the diversity of languages available for the specification of CIGs [9] an ontology-based approach as the one we proposed here and in [5] is very promising. An additional advantage of our approach is that while the mentioned verification methodologies require the user to know temporal logic, our approach allows the user to specify constraints using a graphical notation that hides its equivalent semantic in temporal logic. The main drawback of semantic conformance checking approaches is that for each CIG to be checked an ontology matching between the terms used in the CIG and the concepts used in the verification tool has to be provided, and this can not be automatically performed and will not necessarily always be achievable.

In Mor Peleg et al. comparative study of languages for CIGs [9] the developers of Asbru, GLIF, GUIDE, EON and PRO*forma* languages were asked to specify CIGs for a set of recommendations inspired by the chronic cough guideline [10] for immunocompetent adult patients. The recommendations on which the study [9] was based, and the repository of the resulting CIGs, is available at the Open Clinical repository (http://www.openclinical.org). In [5] we explained that the mentioned medical recommendations, which were expressed in natural language, could be disambiguated and formalized in a declarative language called DECLARE [11]. In [5] we explained a methodology to check the conformance of the DECLARE recommendations over the PRO*forma* guideline from the Open Clinical repository. While semantic conformance checking does not guarantee correctness for all possible scenarios, it is clear that if more histories of executions are collected, the level of confidence and representativeness of the analysis increases. The main advantage of this approach is that it only requires the process history and therefore it can be applied over any CIG, independently of the language used for its implementation.

The aim of this work was to provide a proof of concept that the same DECLARE model proposed in [5] to check the conformance of the PRO*forma* CIG could be reused to check the conformance of other CIG from the same Open Clinical repository. For the best of our knowledge reusing the same specification for checking multiple CIGs defined in different languages have not been explored before. This study involved finding a suitable ontology mapping between the terms used for the DECLARE model and the terms used in the selected CIG. For this purpose we contacted the developers of the CIGs contained in the repository. The developers of the GLIF CIG provided us with an equivalent mapping of the CIG into a Petri Net (PN), which is not available in the Open Clinical Repository. We have transformed the provided PN into a Coloured Petri Net (CPN). CPNs are an extension of PNs where the tokens (flow of execution in the PN) are associated colors (types) defined by the user. We chose to perform our study over the GLIF CIG because it gave us the additional opportunity to explore the use of the model checking techniques supported by CPN tools, a well known tool for modeling and analyzing CPNs (www.cpntools.org). While languages like PRO*forma*, GLARE, GLIF and GUIDE have been mapped into

PNs, to the best of our knowledge the model checking mechanisms provided by the PN formalism have not been used for the verification of CIGs.

In Section 2 we start explaining the medical recommendations on which our example is based: recommendations taken from the chronic cough guideline [10] for immunocompetent adult patients which were selected by Mor Peleg et al. for performing the comparative study summarized in [9]. In Section 3 and 5 we explain respectively how to perform model checking and semantic conformance checking over the GLIF CPN. As for performing semantic conformance checking both DECLARE and the process mining tool ProM are used, they are both introduced in Section 4. Finally we provide the conclusions of our work.

2 Chronic Cough Guideline Recommendations

The analysis and interpretation of natural language medical recommendations is manually done, it requires in most of the cases the expertise of clinicians, as it is non-error free due to incompleteness or ambiguity of the natural language guideline's description. In this section we explain our disambiguation of one of the natural language medical recommendations from the chronic cough guideline [10]. This analysis was already explained in [5].

According to [10] if a patient has a cough which last at least 3 weeks the cough is considered chronic. The chronic cough guideline distinguishes 2 different patient classes for which different diagnostic treatments are prescribed in order to discover the most likely cause of cough and treat it. Here we only consider the case of immunocompetent adult patients.

For this study we have considered the following medical recommendation from the chronic cough guideline for immunocompetent adult patients:

R1) "chest radiographs should be ordered before any therapy is prescribed in nearly all patients with chronic cough. Chest radiographs do not have to be routinely obtained before beginning treatment for presumed PNDS [post nasal drip syndrome] in young nonsmoker, or in pregnant women, or before observing the result of discontinuation of an ACEI [angiotensin-converting enzyme inhibitor]."

R2) "When the chest X-ray is normal, PNDS, Asthma, and GERD [Gastroesophageal reflux disease] are the likely causes of chronic cough."

Given the scope of the paper, we refer the reader to [5] for our interpretation of the recommendation R2. Here we restrict yourselves to explaining our interpretation of recommendation R1):

R1)(a) **Pregnant patient or young non smoker with presumed PNDS:** in the case of pregnant women there is medical evidence of grade II-2 that the *x-ray exposes the embryo to radiation.* Evidence of grade II-2 is obtained from well-designed cohort or case-control analytic studies, preferably from more than one center or research group. Medical evidence of grade II-2 is ranked below the evidence of type I (obtained from at least one randomized controlled trial), and below the evidence of type II (obtained from well-designed controlled non-randomized trials). This recommendation is *critical* and provided with a high medical evidence, therefore it is also *mandatory* and it should be satisfied in every CIG that models the chronic cough guideline from [10].

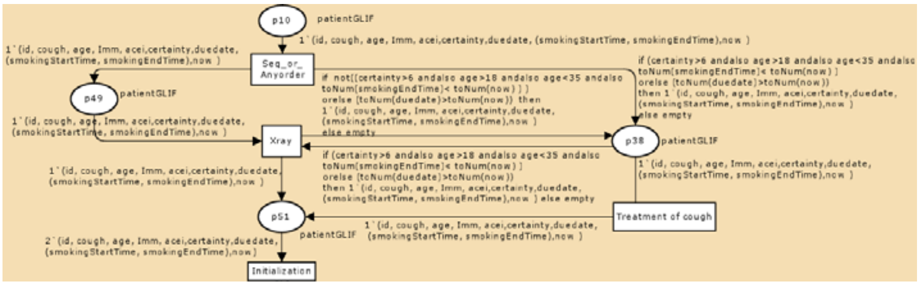
In the case of young non smoker with presumed PNDS there is medical evidence of grade II-2 that the probability of PNDS/Asthma/GERD is higher than the average population, therefore it is more *cost-effective* and less time consuming to skip *Chest X-ray.* This recommendation is *not critical* but provided with a high medical evidence and therefore should be *mandatorily enforced.*

R1) (b) **Patients for whom recommendation R1)(a) does not apply (not pregnant and not young non smokers with presumed PNDS):** therefore for this class of patients obtaining a *Chest X-ray* is strongly recommended based on evidence of grade II-2, promoting the values of maximizing *likelihood of diagnosis* and maximizing *cost-effectiveness* because the X-ray may contain results that can aid in making a correct diagnosis. This recommendation is *not critical* but is provided with a high medical evidence and therefore should be *mandatorily enforced.*

3 Model Checking Techniques for Coloured Petri Nets

The developers of the chronic cough GLIF CIG from the Open Clinical repository provided us with a PN model of the CIG specified in the Protege tool (http://protege.stanford.edu). We have extended the provided PN into a CPN by adding types (colors) and adding conditions to the arcs connecting places and transitions. The resulting CPN can be enacted in CPN tools (http://cpntools.org). Figure 1 depicts a part of the resulting CPN.

For the GLIF CIG we have tested the model checker provided by CPN tools. While languages as GLIF, PRO*forma* and GUIDE have been provided with mappings into variants of PNs, to the best of our knowledge none of them have taken advantage of the model checking features provided by PN based tools like the CPN tools.



```
1`(id, cough, age, Imm, acei,certainty,duedate, (smokingStartTime, smokingEndTime),now )

1`(id, cough, age, Imm, acei,certainty,duedate,
(smokingStartTime, smokingEndTime),now )

if not([certainty>6 andalso age>18 andalso age<35 andalso
toNum(smokingEndTime)< toNum(now) ] )
orelse (toNum(duedate)>toNum(now)) then
1`(id, cough, age, Imm, acei,certainty,duedate,
(smokingStartTime, smokingEndTime),now )
else empty

if (certainty>6 andalso age>18 andalso age<35 andalso
toNum(smokingEndTime)< toNum(now) ]
orelse (toNum(duedate)>toNum(now))
then 1`(id, cough, age, Imm, acei,certainty,duedate,
(smokingStartTime, smokingEndTime),now )
else empty

1`(id, cough, age, Imm, acei,certainty,duedate,
(smokingStartTime, smokingEndTime),now )

1`(id, cough, age, Imm, acei,certainty,duedate,
(smokingStartTime, smokingEndTime),now )

if (certainty>6 andalso age>18 andalso age<35 andalso
toNum(smokingEndTime)< toNum(now) ]
orelse (toNum(duedate)>toNum(now))
then 1`(id, cough, age, Imm, acei,certainty,duedate,
(smokingStartTime, smokingEndTime),now ) else empty

1`(id, cough, age, Imm, acei,certainty,duedate,
(smokingStartTime, smokingEndTime),now )

1`(id, cough, age, Imm, acei,certainty,duedate,
(smokingStartTime, smokingEndTime),now )

2`(id, cough, age, Imm, acei,certainty,duedate,
(smokingStartTime, smokingEndTime),now )
```

Fig. 1. Screen shot of the part of GLIF specification in CPN tools where recommendation recommendation R1) (a) iii can be checked

Below we explain a generic methodology to apply model checking over CPN specifications of CIGs:

1) Generate state-space graph from the CPN: once the medical guideline has been transformed into an equivalent CPN, tools like CPN tools can be used for model checking. In CPN tools properties can be checked only if it is possible to generate the graph of all possible combinations of transitions in the used CPN.

According to the chronic cough guideline six different cases or patient medical conditions were significant: (1) if the cough is persistent, (2) if the patient is pregnant, (3) if the patient is a young adult, (4) if the patient is a smoker, (5) if PNDS is presumed, (6) if the result of the X-ray is normal.

In the GLIF CPN the flow of execution was given by a unique token of type patient, described as the following tuple:

$< (coughStart, coughEnd), age, (immStart, immEnd), (aceiStart, aceiEnd), pndsCertainty, pregnancyDueDate, (smokingStart, smokingEnd), now >$ where:

- $coughStart, coughEnd, immStart, immEnd, aceiStart, aceiEnd, pregnancyDueDate, smokingStart, smokingEnd, now$ are of type date
- age and $pndsCertainty$ are integers.

Therefore: (1) if the difference between *now* and *coughStart* is greater that 21 day and there is no *coughEnd* then the cough is persistent, (2) if the *pregnancyDueDate* is a date after now then the patient is pregnant, (3) if the patient's *age* is greater than 18 and less than 35 then the patient is a young adult, (4) if the *smokingEnd* has not been fixed then the patient is a smoker, (5) if the *pndsCertainty* is greater than 6 then PNDS is presumed.

In order to enact the provided GLIF CPN we needed to initialize it with tokens (patient cases).

The *first option* was to extend the GLIF CPN with random distribution functions which would generate the initial tokens that represent random patient cases. However, a problem which emerges when using random distribution functions is that the resulting CPN becomes non-deterministic because for each state space calculation different random numbers are used for various transitions. This has as result that it is virtually impossible to generate the same state space twice. Consequently, model checking is not possible.

The *second option* was to provide an algorithm which would iteratively enact the CPN from its initial state until some final state, using for each iteration a randomly generated initial token that represents a patient case. In this way we could generate in each iteration a deterministic CPN from which a state space graph could be computed and saved. Each state-space graph contained for the considered patient case all the possible care paths arising from all the possible decision outcomes for each decision point in the GLIF CPN. This is the reason why this methodology is exhaustive and therefore can be considered a type of model checking technique.

2) Verify the chronic cough medical recommendations in the GLIF CPN: CPN tools provides a library which implements a model checker based on a type of CTL temporal logic called ASK-CTL. This logic is an extension of CTL

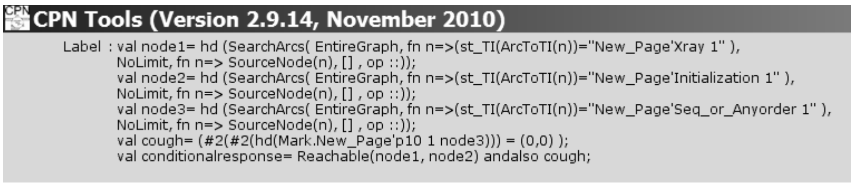
which is a branching time logic. In order to be able to model data and time, CPN tools is integrated with a functional programming language called Standard Meta Language (SML). SML functions can used in CPN tools to prove medical recommendations by traversing the state-space graph obtained from the GLIF CPN.

We have chosen to specify each recommendation for the chronic cough guideline from Section 2 as a call to SML functions which traverse the sate-space graph generated from the GLIF CPN. For instance in the case of the recommendation R1) (a) iii, it has to be checked on the section of the GLIF CPN shown in Figure 1. This recommendation is verified by checking the following: if the patient has cough and the result of the Xray is normal then every time the transition $Xray$ is enacted transition $Initialization$ has to be enacted. The transition $Initialization$ does not have to be enacted straight after the transition $Xray$ and the transition $Xray$ can be enacted multiple times before the transition $Initialization$ gets enacted. In Figure 2 we present the specification of the recommendation R1) (a) iii in CPN tools. The verification of this recommendation returns true if the following is satisfied: 1) the source nodes of the transitions $Xray$ and $Initialization$ are reachable in the corresponding state space graph, and 2) the patient still shows symptoms of cough, which is equivalent to check that the token that activates the transition $Xray$ has value $CoughEnd = (0, 0)$. The GLIF CPN does not provide any equivalent concept to the condition $normalXray$ therefore this conditions could not be checked.

For each of the chronic cough recommendations explained in Section 2 it was possible to define a function which equivalently checks whether the property is satisfied in the state-space graph generated from the GLIF CPN. While the functions were defined in terms of the GLIF ontology, they could potentially be parameterized in order to prove the same properties in another CPN which also models the chronic cough guideline.

For instance in the case of the recommendation R1) (a) iii, it could be parameterized by replacing:

1) the strings "$New\_Page'Xray$ 1", "$New\_Page'Initialization$ 1" and "$New\_Page'Seq\_or\_Anyorder$ 1" corresponding to the labels of the transitions $Xray$, $Initialization$ and $Seq\_or\_Anyorder$ for variables of type string; and 2) the label of the place $p10$, which is the source node of transition $Seq\_or\_Anyorder$, for a variable denoting a place.


CPN Tools (Version 2.9.14, November 2010)

```
Label : val node1= hd (SearchArcs( EntireGraph, fn n=>(st_TI(ArcToTI(n))="New_Page'Xray 1" ),
        NoLimit, fn n=> SourceNode(n), [] , op ::));
        val node2= hd (SearchArcs( EntireGraph, fn n=>(st_TI(ArcToTI(n))="New_Page'Initialization 1" ),
        NoLimit, fn n=> SourceNode(n), [] , op ::));
        val node3= hd (SearchArcs( EntireGraph, fn n=>(st_TI(ArcToTI(n))="New_Page'Seq_or_Anyorder 1" ),
        NoLimit, fn n=> SourceNode(n), [] , op ::));
        val cough= (#2(#2(hd(Mark.New_Page'p10 1 node3))) = (0,0) );
        val conditionalresponse= Reachable(node1, node2) andalso cough;
```

Fig. 2. Specification of the recommendation R1) (a) iii in CPN tools

Each of the defined functions was checked for each of the state-space graphs generated in 2) from the GLIF CPN. With this methodology we could prove that the GLIF guideline fully satisfies the mandatory recommendations RG1), RG2), R1) and R2)i, and that the GLIF CPN partially satisfies the optional property R2)ii.

4 Constraint-Based Specification: DECLARE and ProM

DECLARE (`www.win.tue.nl/declare/`) is a flexible and extendible constraint-based workflow management system that provides multiple declarative languages (DecSerFlow, ConDec, etc.) [11]. Unlike workflow-based languages, like PRO*forma* and GLIF, declarative languages specify *what* tasks should be performed without determining *how* to perform them. PRO*forma* and GLIF are specification methods for structured representation of guideline where processes are organized in terms of: actions, branches, decision points, synchronization steps, etc. With DECLARE instead it is possible to specify unstructured medical recommendations by means of dependencies or constraints between tasks. Dependencies between tasks can be seen as general rules that the user should comply with during a process execution. Any task in the model can be enacted by the user if and only if none of the specified constraints is violated. If an execution trace does not violate a DECLARE specification it is allowed. For a more extensive analysis of the benefits of specifying CIGs using declarative approaches we refer the reader to [12].

DECLARE uses a graphical notation and semantics based on Linear Temporal Logic (LTL). In DECLARE constraints can be *mandatory* or *optional*. Graphically, mandatory constraints are depicted as solid lines and optional constraint as dashed lines. While the considered recommendations from the chronic cough guideline have not been assigned a level of support, DECLARE allows to attach to constraints a level of support from 1 to 10. *Data attributes* can be specified and associated to relevant tasks. While executing a task, its data attributes can be read or written, as specified for that task at design-time. Constraints can be *conditional*, such that if the condition associated to the constraint is true the constraint should be satisfied. The condition can be defined in terms of data attributes. For example an X-ray should be performed only if the patient is not pregnant. Graphically we represent conditions between brackets. By associating to the DECLARE constraints different levels of support and ranges of numeric conditions it is possible to provide flexible constraint specifications. For example, while the prescription of treatment "A" is mandatory for patients with systolic blood pressure between 130 and 140, it can still be recommended (optional DECLARE specification with high level of support) for patients with systolic blood pressure greater than 141 but not exceeding 145.

Whereas declarative languages like DECLARE aim to provide flexibility, the goal of *process mining* [13] is to use information stored in information systems. The idea of process mining is to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in todays information systems. The first type of process mining is discovery. A discovery technique takes an event log and produces a model without using any a-priori information (e.g. the genetic miner). The second type of process mining is conformance. Here, an existing process model is compared with an event log of the same process. Conformance checking can be used to check if reality, as recorded in the log, conforms to the model and vice versa.

Obviously, process mining is very useful in the healthcare context as processes are not enforced by systems but emerge through human behavior. In the remainder of the paper, we show some initial applications of DECLARE and the process mining tool ProM (`www.processmining.org`) in this domain.

5 Semantic Conformance Checking of the CIG Guideline

In this section we focus on the semantic conformance checking of the CIG. In [5] we showed how to define a DECLARE model to specify the chronic cough medical recommendations explained in Section 2. The obtained DECLARE model was used to check the conformance of the constraints over the PRO*forma* CIG from the Open Clinical Repository. Here we only explain how to model in DECLARE the recommendation R1) (a)iii. The corresponding DECLARE model is shown in Figure 3. For the explanation of the other medical recommendations we refer the reader to [5]:

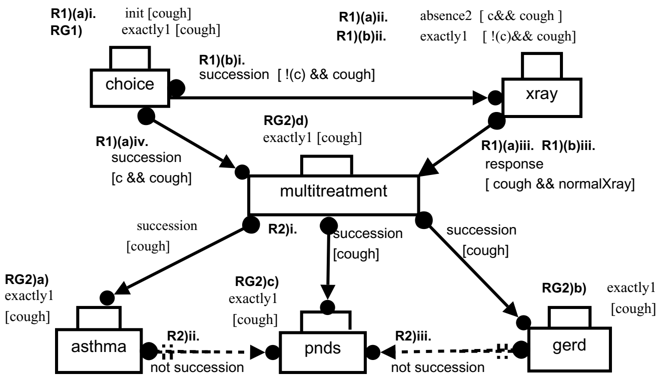


Fig. 3. DECLARE model for the considered recommendations from the chronic cough guideline

Recommendation R1) (a)iii: start the treatment for PNDS/Asthma/GERD if after the *X-ray* the patient has persistent cough and the result of the *X-ray* is normal (*cough && normalXray*) (conditional response relation between tasks *xray* and *multitreatment*);

The DECLARE model from Section 5 was used in [5] for checking the conformance of the chronic cough recommendations over a PRO*forma* CIG. The purpose of this study was to analyze how difficult would be to provide an ontology mapping between the terms used in the DECLARE model and the terms

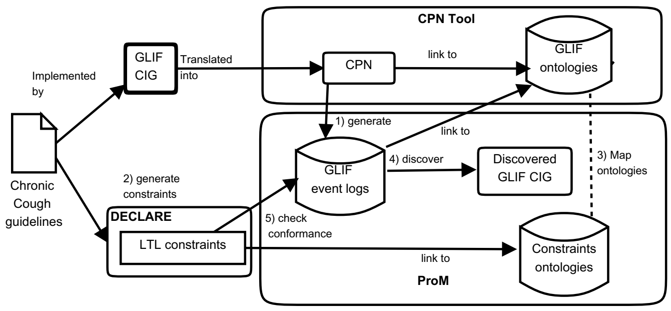


Fig. 4. Methodology proposed here: 1) Generate logs by enacting the CPN in CPN tools, 2) Generate LTL constraints from the DECLARE model, 3) Map ontologies, 4) Discover the model mined from the event logs using ProM, 5) Check conformance using ProM

used in the GLIF CIG from the Open Clinical repository. Our ultimate goal was to reuse the same DECLARE model to check the conformance of the GLIF CIG.

Below we explain in detail the steps of the semantic model checking methodology that has been applied (depicted in Figure 4): 1) Generate semantically annotated event logs from the CIG: the execution history (event logs) of a CIG is independent of the language used for the specification of the CIG. An event log contains the executions of one or more processes. To construct such log it is required that each event in the log (e.g. an X-ray) can be mapped to a single case or process instance (e.g. a patient treated for cough) and that each process instance can be mapped to a single process (e.g. the process for treating chronical cough). Similarly, every process instance has zero or more tasks. Every task or audit trail entry must have at least a name and an event type. The event type determines the state of the tasks. Although the methodology explained here is generic, we decided to explain it with the GLIF CIG used in Section 3. We chose to obtain the event logs from the enactment of the corresponding GLIF CPN representation in CPN tools.

Unfortunately none of the CIG from the Open Clinical repository have been used so far in a real medical application. So event logs had to be generated with fictitious patient cases. For this we extended the CPN specification of the GLIF CIG as explained in [14]. With the introduced extensions it was possible to choose the starting transition of the CPN and to select those transitions whose enactment was recorded in the generated event logs. Therefore we chose from the 26 transitions provided in the GLIF CPN only 7 transitions to be ontologically mapped into tasks from the DECLARE specification. For instance we chose the transitions *Evaluate asthma* and *Set PNDS evaluated* from the CPN because they could be mapped into the semantically equivalent tasks *asthma*, *pnds* from the DECLARE model from Section 5. Those transitions that were not selected were those that had no counterpart in the DECLARE model for various reasons. For example, they were generated by the algorithm used to map the GLIF guideline into an equivalent CPN (like the transition *Seq_or_any_order* used in the CPN for simulating scheduling constraints), or they were used to manipulate clinical data (for instance the transition *get_patient_cough_related_data* in the CP). None of those transitions have a counterpart in a declarative formalism like DECLARE that allows to abstract from most of these implementation details. Therefore even if we have modeled all the transitions with no counterpart in the DECLARE model the results presented here would not have changed.

The event logs had to be generated considering random patient cases. But for each of the patient cases more than one process instance could be generated, depending on the flexibility of the decision points provided in the GLIF CIG. For instance in the GLIF CIG, the clinician can chose to carry on the multitreatment for asthma, GERD and PNDS in any of the 6 sequential combinations. For instance, it is possible to test for asthma, then for GERD and then for PNDS. Next to that the clinician can also decide not to carry on any test.

By using the CPN Tools import plug-in provided by the ProM$_{import}$ framework (www.prom.win.tue.nl/tools/promimport/) we could automatically convert the event logs generated from the extended CPN into a format that can be interpreted by ProM. So semantic model checking as explained in [5] can be used to check the conformance of the medical recommendations expressed in the DECLARE tool in the CIG:

2) Generate LTL properties from the DECLARE model: the DECLARE tool automatically generates the LTL properties from the constraint model of the medical recommendations explained in Section 5. From the generated LTL properties two ontologies were obtained: a) a data ontology obtained from the DECLARE data perspective and b) an ontology of activities obtained the from the DECLARE tasks. The top rectangle in Figure 5 shows the graphical representation of the ontology of activities, where for instance the DECLARE activities *asthma* and *gerd* are depicted.

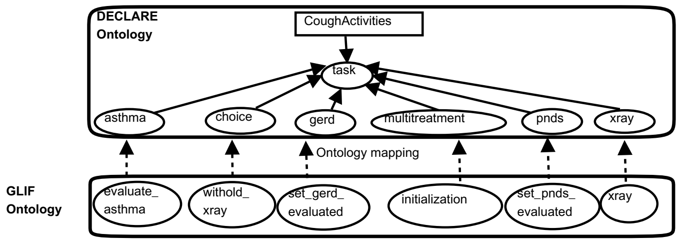


Fig. 5. Ontology matching between DECLARE and GLIF activities

3) Map the DECLARE ontology with the GLIF ontology: we performed the ontology mapping shown in Figure 6 between the concepts used in the GLIF CPN and the concepts from the DECLARE ontologies (generated in 2)). For instance the GLIF task *evaluate_asthma* maps into the DECLARE concept *asthma*.

4) Discover the GLIF model from the semantically annotated event logs. Figure 6 shows the GLIF process mined from the generated event logs using the ProM framework.

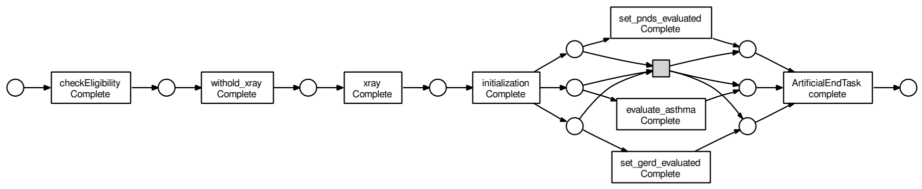

checkEligibility Complete
withold_xray Complete
xray Complete
initialization Complete
set_pnds_evaluated Complete
evaluate_asthma Complete
set_gerd_evaluated Complete
ArtificialEndTask complete

Fig. 6. GLIF CPN discovered by the ProM framework. The grey rectangle represents a hidden transition which allows for executing the "set pnds evaluated", "set gerd evaluated", and "evaluate asthma" transitions in any order or not at all.

5) Perform semantic conformance checking of the discovered GLIF model: with the semantic LTL checker plug-in provided by the PROM tool we could check the conformance of the recommendations specified in the DECLARE model (Section 5) in the discovered CPN.

With this methodology we obtained the same results proved in Section 3 with model checking.

6 Conclusions

The use of specification languages as DECLARE opens the possibility that medical recommendations become available as formal models defined in terms of standard medical ontologies like the Unified Medical Language System (http://www.nlm.nih.gov/research/umls/). Furthermore if the developers of a CIG, independently of the specification language, can provide a mapping between the terminology used in the CIG and the concepts used to specify the DECLARE model then it is possible to uniformly check the conformance of the CIG. Considering that multiple languages coexist for the specification of CIG and no common standard language has been adopted yet by the Health community, this result could have a considerable practical benefit. To support our claim we have shown how the DECLARE model presented in section 5 can be used to perform conformance checking of a similar GLIF and PRO*forma* CIG.

From this work we also learned that in the DECLARE language it is not possible to specify temporal conditions in constraints. For instance, a medical encounter needs to start between 2 or 3 days after the patient asked for an appointment and the duration of the medical encounter should not last more than 15 minutes. Therefore as future work we plan to extend DECLARE with more expressive ways to specify temporal restrictions.

Besides from the experience we gained in Sections 3 and 5 from model checking and conformance checking the GLIF CPN it seems that:

1) CPN model checking can easily lead to an explosion of the state space graph and the impossibility to perform any analysis.

2)Both techniques provide equivalent mechanisms to define the ontology mapping between the CIG's terms and the DECLARE model.

3) When using the semantic conformance checker it is possible to: differentiate between *mandatory* and *optional* constraints and choose significant transitions while ignoring others (by using event filtering in ProM). None of the mentioned features are provided by the model checker supported by CPN tools.

References

1. Quaglini, S., Stefanelli, M., Lanzola, G., Caporusso, V., Panzarasa, S.: Flexible guideline-based patient careflow systems. Journal AIME 22, 65–80 (2001)
2. Advani, A., Shahar, Y., Musen, M.A.: Medical Quality Assessment by Scoring Adherence to Guideline Intentions. In: AMIA 2001 (2001)
3. Groot, P., Hommersom, A., Lucas, P.J.F., Robbert-Jan, M., ten Teije, A., Harmelen, F.V.: Using model checking for critiquing based on clinical guidelines. Journal AIME 46, 19–36 (2009)
4. Bottrighi, A., Chesani, F., Mello, P., Molino, G., Montali, M., Montani, S., Storari, S., Terenziani, P., Torchio, M.: A Hybrid Approach to Clinical Guideline and to Basic Medical Knowledge Conformance. In: Combi, C., Shahar, Y., Abu-Hanna, A. (eds.) AIME 2009. LNCS, vol. 5651, pp. 91–95. Springer, Heidelberg (2009)
5. Grando, M.A., Schonenberg, M.H., van der Aalst, W.: Semantic Process Mining for the verification of medical recommendations. In: Int. Conf. of Health Informatics 2011, Rome, pp. 5–16 (2011)
6. Bäumler, S., Balser, M., Dunets, A., Reif, W., Schmitt, J.: Verification of Medical Guidelines by Model Checking – A Case Study. In: Valmari, A. (ed.) SPIN 2006. LNCS, vol. 3925, pp. 219–233. Springer, Heidelberg (2006)
7. Giordano, L., Terenziani, P., Bottrighi, A., Montani, S., Donzella, L.: Model checking for clinical guidelines: an agent-based approach. In: AMIA, pp. 289–293 (2006)
8. Marcos, M., Balser, M., ten Teije, A., van Harmelen, F., Duelli, C.: Experiences in the Formalisation and Verification of Medical Protocols. In: Dojat, M., Keravnou, E.T., Barahona, P. (eds.) AIME 2003. LNCS (LNAI), vol. 2780, pp. 132–141. Springer, Heidelberg (2003)
9. Peleg, M., Tu, S.W., Bury, J., Ciccarese, P., Fox, J., et al.: Comparing Computer-Interpretable Guideline Models: A Case-Study Approach. JAMIA 10(1), 55–68 (2003)
10. Irwin, R.S., Boulet, L.S., Cloutier, M.M., et al.: Managing Cough as a Defense Mechanism and as a Symptom, A Consensus Panel Report of the American College of Chest Physicians. Chest 114(2), 133–181 (1998)
11. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative Workflows: Balancing Between Flexibility and Support. Computer Science - Research and Development 23(2), 99–113 (2009)
12. Mulyar, N., Pesic, M., van der Aalst, W.M.P., Peleg, M.: Towards Flexibility in Clinical Guideline Modelling Languages. In: 1st International Workshop on Process-oriented Information Systems in Healthcare, pp. 24–29 (2007)
13. van der Aalst, W.M.P.: Process Mining: Discovery. Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011)
14. Alves De Medeiros, A.K., Günther, C.W.: Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms. In: Proc. of the Sixth Workshop and Tutorial on Practical Use of CPNs and the CPN Tools, pp. 177–190 (2005)