Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 324 (2016) 31–50

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

# Formal Verification of Health Assessment Tools: a Case Study

Jonas Santos Bezerra, Andrei Costa, Leila Ribeiro, Érika Cota[1]

*Instituto de Informática*
*Universidade Federal do Rio Grande do Sul*
*Porto Alegre - RS, Brazil*

**Abstract**

Health assessment tools (treatment standardization guidelines, risk evaluation scales, disease burden estimations, and patient's perceptions questionnaires, among others) are very similar in format to a software specification, although targeted to humans. As in any document written in natural language, such medical approaches are prone to errors and misunderstandings caused by ambiguities, omissions, or inconsistencies thus reducing the applicability and efficacy of these tools. The verification of health assessment tools is an important step for standardization but it is still a manual and ad-hoc process in the medical community. This work proposes the use of a formal approach for the verification of health assessment tools. We apply and evaluate a methodology originally proposed for the verification of Use Cases to a specific medical standardization guideline. Preliminary results show that formal verification of these medical artifacts can be a cost-effective mechanism to validate and qualify health approaches.

*Keywords:* Graph Transformations, Formal Verification, Textual Documents, Medical Guidelines, Use Cases

## 1 Introduction

A treatment standardization guideline is a common tool used in a medical research protocol to ensure that all data collected meet certain requirements, allowing further comparative analysis. For instance, when a new drug is developed and tested in humans, guidelines for its application are defined as part of the research protocol. Training of the technical personnel to follow the defined standard may be necessary, not only during a trial, but also for approved treatments. Hence, it is important that this medical artefact is not only technically correct, but also clear and complete for its target audience. The focus of this work is the quality of health assessment documents that describe standardized procedures in the medical field.

A health assessment tool is a document written in natural language and is very similar in goal to a software specification document. As such, this artefact is prone

[1] Email: jsbezerra@inf.ufrgs.br, acosta@inf.ufrgs.br, leila@inf.ufrgs.br, erika@inf.ufrgs.br

http://dx.doi.org/10.1016/j.entcs.2016.09.005
1571-0661/© 2016 Published by Elsevier B.V.

to the same type of faults of a software specification, such as omission and ambiguities. In the medical community, one is mostly concerned with the validation of the assessment tool, which means, in that context, how effective is the proposed standard as well as the level of adherence to its proposal [3,5]. The verification of the document that describes the standard procedure with respect to its clarity is performed indirectly as part of the validation process. The first step in the validation process is performed through revision meetings. Such meetings involve experts in the medical procedure or disease related to the standard and their goal is to provide an overall evaluation of the proposed standard, and is mainly focused on the correctness from the medical point of view. In a second moment, validation in the field is performed, that is, the proposed procedure is used in a controlled environment and one evaluates the response of the health workers to the new method. In this phase, checklists are typically used to assess the acceptance of the proposed health assessment tool and the overall clarity of the document and instructions is typically evaluated in this moment. The verification of the guideline text using this process has two main problems. First, it is well known that manual inspection, despite following a defined methodology, is an informal process where all steps are based on aspects intuitively identified by observers from their previous knowledge and individual experience with the object in question. For example, experts on the object in question may rely on tacit knowledge and ignore the need of making this knowledge explicit in the document. Second, faults detected during later validation phases (on-field) incur additional cost and re-work and may, in the worst case, invalidate a whole research protocol.

This work proposes the use of formal methods to verify health assessment tools described in natural language. We apply the verification methodology based on Graph Transformations (GT) proposed in [6] to a new standardization guideline proposed for the treatment of Cutaneous Leishmaniasis (CL). Our preliminary results show that formal verification of this artefact can early pinpoint a number of issues that would otherwise remain unknown and might compromise the applicability of this health tool. In this paper we i) detail the verification methodology, explaining the type of error detected at each step; ii) detail the application of the methodology to the CL treatment guideline by discussing the problems raised during the verification process and explaining the evolution of the artefact throghout the analysis.

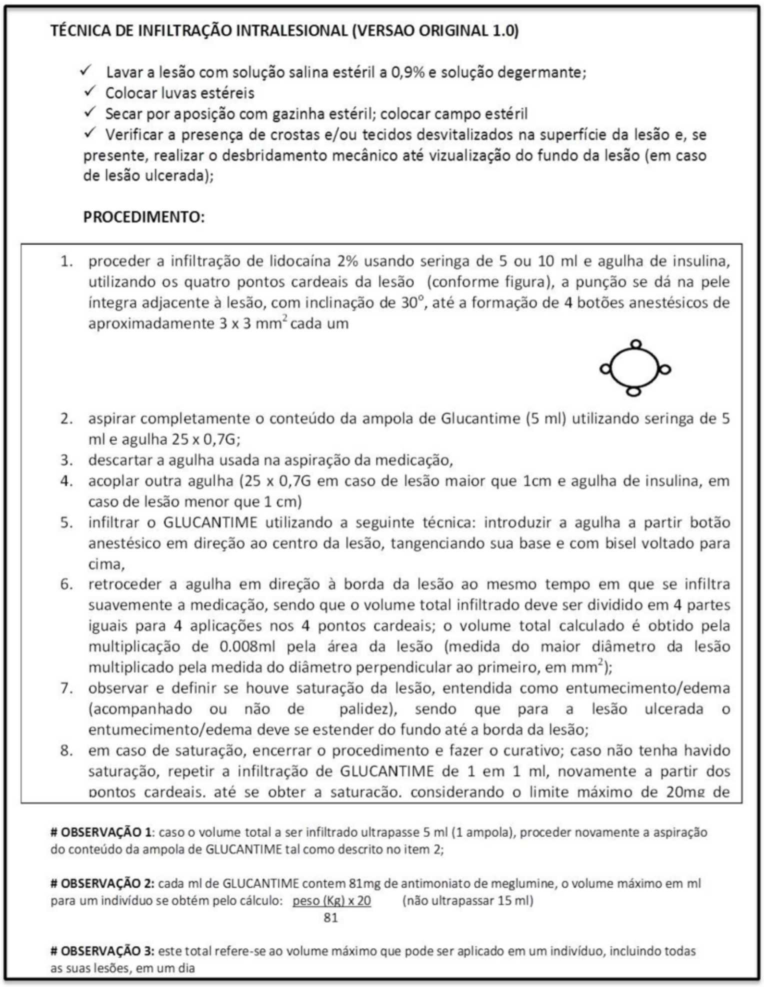
The paper is organized as follows: Section 2 describes the case study. Section 3 gives an overview of the GT-based verification methodology. Section 4 presents more details on the verification methodology while describing its application to the health standardization guideline as well as the issues raised during this process. Section 5 discusses the results of the proposed approach.

## 2 Case Study: Standard Guideline for Leishmaniasis Treatment

Cutaneous Leishmaniasis (CL) is a disease caused by parasites and characterized by skin ulcers. Treatment evidence for this type of Leishmaniasis is still poor and object of active research [1]. The treatment of CL is still a challenge not only because of the scarcity of effective drugs and the high toxicity of the drug of first choice - antimonial derivatives - but also because of the lack of cost-effectiveness studies comparing traditional and new treatments. Such an analysis is crucial for this disease specifically because of its prevalence on poor countries and neglected populations. An important part of cost-effectiveness analysis is based on experimental data that can be collected as part of a research protocol. This means data must be collected following strict standardized procedures that allow further comparison and analysis over different locations and scenarios.

René Rachou Research Center (Fiocruz-MG) is currently developing a standardization treatment guideline for a specific procedure for CL treatment. This guideline has been subject of informal revision by medical experts and is now under the process of validation (applicability and acceptance analysis). The goal of the CL treatment guideline is to standardize the infiltration procedure of a new drug. Indeed, if this procedure is not standardized, each physician may perform this infiltration in a distinct manner, which may affect the results. For instance, the location of the infiltration in the ulcer may vary, as well as the volume of medication used or the evaluation of the treatment outcome. With so many possible variations, one cannot assess the effectiveness of the new drug as divergences in the treatment outcome may have multiple causes. Thus, the proposed CL treatment guideline aims at defining the complete infiltration procedure, including preparation, anesthesia, position for infiltration, definition of possible outcomes, etc. The guideline is a one-page document composed of 4 preparation steps, 8 steps that define the standard procedure, and 3 final observations that may affect the intervention, as shown in Figure 2 (in portuguese). Each step of the procedure contains several instructions and information to be considered.

Once the guideline is validated in a controlled environment, a cost-effectiveness analysis will be performed. A research protocol for a multi-center evaluation will be defined and data about this specific treatment will be collected. This means the standard will be disseminated among different medical centers to be used by untrained personnel in an uncontrolled environment. Nonetheless, one must trust that all data is collected following the same infiltration procedure, so that cost-effectiveness analysis can be carried out. Thus, it is crucial that guideline's instructions are simple, clear and complete, to minimize the risk of collecting misleading data.



**TÉCNICA DE INFILTRAÇÃO INTRALESIONAL (VERSAO ORIGINAL 1.0)**

✓ Lavar a lesão com solução salina estéril a 0,9% e solução degermante;
✓ Colocar luvas estéreis
✓ Secar por aposição com gazinha estéril; colocar campo estéril
✓ Verificar a presença de crostas e/ou tecidos desvitalizados na superfície da lesão e, se presente, realizar o desbridamento mecânico até vizualização do fundo da lesão (em caso de lesão ulcerada);

**PROCEDIMENTO:**

1. proceder a infiltração de lidocaína 2% usando seringa de 5 ou 10 ml e agulha de insulina, utilizando os quatro pontos cardeais da lesão (conforme figura), a punção se dá na pele íntegra adjacente à lesão, com inclinação de 30°, até a formação de 4 botões anestésicos de aproximadamente 3 x 3 mm$^2$ cada um
2. aspirar completamente o conteúdo da ampola de Glucantime (5 ml) utilizando seringa de 5 ml e agulha 25 x 0,7G;
3. descartar a agulha usada na aspiração da medicação,
4. acoplar outra agulha (25 x 0,7G em caso de lesão maior que 1cm e agulha de insulina, em caso de lesão menor que 1 cm)
5. infiltrar o GLUCANTIME utilizando a seguinte técnica: introduzir a agulha a partir botão anestésico em direção ao centro da lesão, tangenciando sua base e com bisel voltado para cima,
6. retroceder a agulha em direção à borda da lesão ao mesmo tempo em que se infiltra suavemente a medicação, sendo que o volume total infiltrado deve ser dividido em 4 partes iguais para 4 aplicações nos 4 pontos cardeais; o volume total calculado é obtido pela multiplicação de 0.008ml pela área da lesão (medida do maior diâmetro da lesão multiplicado pela medida do diâmetro perpendicular ao primeiro, em mm$^2$);
7. observar e definir se houve saturação da lesão, entendida como entumecimento/edema (acompanhado ou não de palidez), sendo que para a lesão ulcerada o entumecimento/edema deve se estender do fundo até a borda da lesão;
8. em caso de saturação, encerrar o procedimento e fazer o curativo; caso não tenha havido saturação, repetir a infiltração de GLUCANTIME de 1 em 1 ml, novamente a partir dos pontos cardeais, até se obter a saturação, considerando o limite máximo de 20mg de

**# OBSERVAÇÃO 1**: caso o volume total a ser infiltrado ultrapasse 5 ml (1 ampola), proceder novamente a aspiração do conteúdo da ampola de GLUCANTIME tal como descrito no item 2;

**# OBSERVAÇÃO 2:** cada ml de GLUCANTIME contem 81mg de antimoniato de meglumine, o volume máximo em ml para um indivíduo se obtém pelo cálculo: $\frac{\text{peso (Kg) x 20}}{81}$ (não ultrapassar 15 ml)

**# OBSERVAÇÃO 3:** este total refere-se ao volume máximo que pode ser aplicado em um indivíduo, incluindo todas as suas lesões, em um dia

Fig. 1. Case Study: CL treatment guideline proposed by FIOCRUZ-MG

## 3 Verification Methodology

We have proposed in [6] a verification methodology for software specification documents described as Use Cases (UC). The methodology consists of constructing a *Graph Transformation* (GT) model using the entities and actions described in the contents of a UC, and performing some analysis over this graph to detect issues such as inconsistencies and ambiguity in the text. Considering the similarities we identified between a software specification artefact and a health assessment tool we propose to use our GT-based verification approach to formally verify the CL treatment guideline proposed by FIOCRUZ-MG. In this section we review the main concepts of the verification methodology, before applying it to the problem in hand.

### *3.1 Background*

The formalism of *Graph Transformations (GT)* [7,4] is based on defining states of a system as graphs and state changes as rules that transform these graphs. Due to space limitations, in this section, we only provide an informal overview of the notions used in this paper. For formal definitions, see e.g. [7]. Examples of graphs, rules and their analysis are presented in the following sections.

*Graphs* are structures that consist of a set of nodes and a set of edges. Each edge connects two nodes of the graph, one representing a source and another representing a target. A *total homomorphism* between graphs is a mapping of nodes and edges that is compatible with sources and targets of edges. Intuitively, a total homomorphism from a graph *G1* to a graph *G2* means that all items (nodes and edges) of *G1* can be found in *G2* (but distinct nodes/edges of *G1* are not necessarily distinct in *G2*). If we have a graph, say *TG*, that represents all possible (graphical) types that are needed to describe a system, a total homomorphism $h$ from any graph $G$ to *TG* would associate a (graphical) type to each item of $G$. We call this triple $\langle G, h, TG\rangle$ a *typed graph*, and *TG* is called a *type graph* (that is, nodes of *TG* describe all possible types of nodes of a system, and edges of *TG* describe possible relationships between these types).

A *Graph Rule* describes a relationship between two graphs. It consists of: a *left-hand side (LHS)*, which describes items that must be present for this rule to be applied; a *right-hand side (RHS)*, describing items that will be present after the application of the rule; and a *mapping from LHS to RHS*, which describes items that will be preserved by the application of the rule. This mapping must be compatible with the structure of the graphs (i.e., a morphism between typed graphs) and may be partial. Items that are in the LHS and are not mapped to the RHS are *deleted*, whereas items that are in the RHS and are not in the image of the mapping from the LHS are *created*. We also assume that rules do not merge items, that is, they are injective.

A *GT System* consists of a type graph, specifying the (graphical) types of the system, and a set of rules over this type graph that define the system behavior. The application of a rule $r$ to a graph $G$ is possible if an image of the LHS of $r$ is found in $G$ (that is, there is a total typed-graph morphism from the LHS of $r$ to $G$). The result of a rule application deletes from $G$ all items that are not mapped in $r$ and adds the ones created by $r$.

Our analysis of GTs is based on concurrent rules and critical pairs, two methods of analysis independent from the initial state of the system and, thus, they are complementary to any other verification strategy based on initial states (such as testing), detailed further ahead.

### *3.2 UC Formalization and Verification Strategy*

A *Use Case (UC)* defines a contract between stakeholders of a system, describing part of the system behavior [2]. The main purpose of a UC description is the documentation of the expected system behavior and to ease the communication between stakeholders, often including non-technical people, about required system functionalities. For this reason, the most usual UC description is the textual form. A general format of a UC contains a set of sequential steps describing the successful interaction between the primary actor and the system towards the primary goal. A sequence of alternative steps are often included to represent exception flows. Pre- and post-conditions are also listed to indicate, respectively, conditions that must hold before and after the UC execution.

Figure 2 summarizes the UC formalization and verification strategy proposed in [6], which is divided into four main phases:

1 *Data Extraction*: where we identify entities and actions in the text of the UC that will be used to construct the model.

2 *Primary Verifications*: where we look for problems such as entities or conditions that were found in the previous phase but are never used and also for actions and effects not clearly defined. As these problems might affect or even prevent the creation of the model, they have to be either solved by rewriting the UC or annotated as open issues to be resolved later on if they are not prohibitive to the model construction.

3 *GT Generation*: where we construct the GT model by modeling conditions and effects as graphs, building a type graph and then modeling each step of the UC as transition rule from one state graph (the conditions) to other (the effect).

4 *UC Analysis*: where we perform a series of automated verifications over the GT model to detect possible flaws in the UC. This is done by using the AGG tool [8] and its concurrent rules, conflict and dependency analyzes.

The issues found during the methodology application are annotated as *open issues (OIs)* along with possible solutions (when applicable) to be confirmed and solved later by the analyst (see table 1 for the complete list of *Open Issues* that can be raised). With this approach, any design decision made over an OI can be documented and traced back to the original document. Through analysis, it is possible to verify whether the pre- and post-conditions were correctly included in the model, whether there are conflicting and/or dependent rules, what is the semantics of a detected conflict or dependency, and whether these results were expected or not. One important point is that, during the process of representing the document in the formal model, clarifications and decisions about the semantics of the textual description must be made. Annotated OIs force the stakeholders to be more precise and explicit about tacit knowledge and unexpressed assumptions about system invariants and expected behavior.

Open issues are classified according to their severity level: code Yellow (⚠) indicates a warning, meaning a minor problem that can probably be solved by a single person; code Orange (?) indicates a problem that requires more attention and probably a definition/confirmation from the stakeholders; code Red (✋) indicates a serious issue that requires a modification in the UC description. Below, we describe the steps of the methodology while applying it over the medical guideline as an UC.

| **Open issue** | **Verification** | **Problem** | **Severity level** | **Possible action** |
|---|---|---|---|---|
| OI.1 | An entity listed in Step 1 is not used (as actor or involved) in any action | Different names for the same entity or entities used in pre-/post-conditions are not used in the steps of the UC | ⚠ **Yellow** | Analyze whether this is actually what is intended, |
| OI.2 | A branching condition is not used in any action | The description of the actions may be too abstract | ⚠ **Yellow** | Analyze whether this is actually what is intended |
| OI.3 | The effect of an action is not clearly defined | Ambiguous description or omission | 🟥 **Red** | Provide more details in the UC description |
| OI.4 | A concurrent rule (for any alternative path in the UC) cannot be built using all the rules in the corresponding RSs | Items generated by some rule and used by another one may be missing by omission or modeling error | 🟥 **Red** | Review the rules |
| OI.5 | Multiple concurrent rules are built for a single UC scenario | Multiple instances of one or more entities are possible, leading to different (possibly unexpected) ways of combining the rules of the UC | 🟥 **Red** | Check dependencies between rules to find unexpected subpaths in the UC behavior |
| OI.6 | UC pre-conditions are not a subgraph of the LHSs of the concurrent rules | Pre-conditions may include unnecessary items | ⚠ **Yellow** | Remove unused pre-conditions from the UC text |
| OI.7 | The LHS of a concurrent rule is not a subgraph of the UC pre-conditions | UC requires something that is not explicitly stated in the pre-conditions | ? **Orange** | Identify the RS in problematic concurrent rule and check whether all actions in this path were correctly modeled. If model is correct, check for missing pre-conditions. |
| OI.8 | Post-conditions of an alternative path of the UC are not contained in the RHS of the corresponding concurrent rule | Some rule is not generating a required item (by UC omission or modeling mistake) | 🟥 **Red** | Check the rules. If all rules seem to be correct, post-conditions might be too strong. |
| OI.9 | The RHS of a concurrent rule is not contained in the corresponding UC post-condition | Some rule is not deleting a required item (by UC omission or modeling mistake) | 🟥 **Red** | If the rules seem to correctly describe each action, post-conditions might be too weak |
| OI.10 | A rule is not conflicting with itself | The rule could be applied an arbitrary number of times | ⚠ **Yellow** | Analyze whether this is actually the intended behavior |
| OI.11 | There is no conflict between rules that represent the branching points of the UC behavior | Non-deterministic behavior: any alternative path can be taken no matter the condition | 🟥 **Red** | Revise the conditions (LHSs) associated with rules representing alternative paths in the UC |
| OI.12 | Conflicts between rules other than the ones described above (with itself and branch points) | These conflicts represent branches in system execution that must be explicitly stated in the UC (and in the model) as an alternative path | ? **Orange** | Revise the conflicting rules |
| OI.13 | Dependencies listed do not represent dependencies that are desired in the system | Possible omission in the UC description or a modeling error | ⚠ **Yellow** | Check the RHS of a rule and the LHS of the other rule that depends on the first one |
| OI.14 | An expected dependency between rules does not appear | Possible omission in the UC description or a modeling error. | ⚠ **Yellow** | Check the rules involved |

Table 1
Table of Open Issues and possible solutions

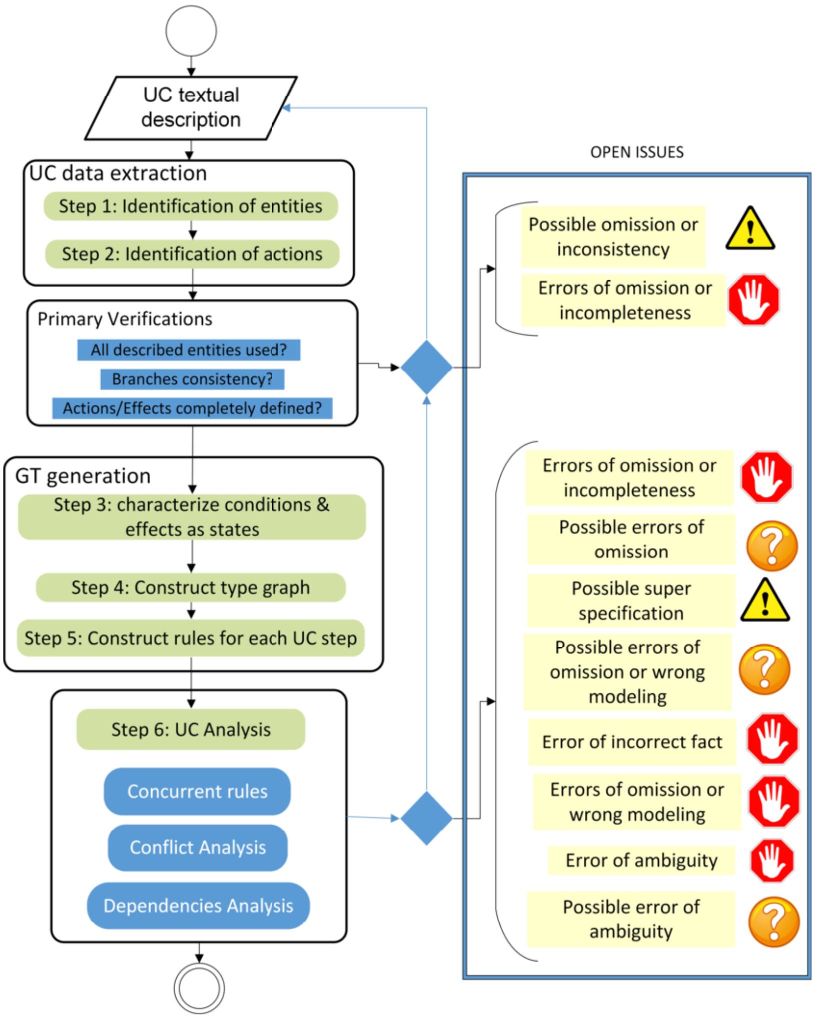

UC textual description
UC data extraction
Step 1: Identification of entities
Step 2: Identification of actions
Primary Verifications
All described entities used?
Branches consistency?
Actions/Effects completely defined?
GT generation
Step 3: characterize conditions & effects as states
Step 4: Construct type graph
Step 5: Construct rules for each UC step
Step 6: UC Analysis
Concurrent rules
Conflict Analysis
Dependencies Analysis
OPEN ISSUES
Possible omission or inconsistency
Errors of omission or incompleteness
Errors of omission or incompleteness
Possible errors of omission
Possible super specification
Possible errors of omission or wrong modeling
Error of incorrect fact
Errors of omission or wrong modeling
Error of ambiguity
Possible error of ambiguity

Fig. 2. Overview of the UC formalization and verification strategy.

## 4 Applying the verification approach to a treatment guideline

The CL treatment guideline proposed by FIOCRUZ-MG consists of a series of instructions that must be followed by a physician. Since the target user of the guideline is a human being, its text is much more dense and follows no pre-defined notation or format. Indeed, several actions are included in a single step of the original guideline. Thus, we first translated the original document to a typical UC notation where one can clearly identify a set of pre and post conditions, and each step contains a single (or a small set of) action(s). In this first translation, the same words and actions present in the original document were kept, but split into more focused steps.

Fig. 3 shows the UC derived from the standard guideline. Hereafter we refer to this UC as the original representation of the standard guideline under verification.

Due to space limitations, we present the verification process applied only to the main scenario of the UC. For the same reason, we included in Fig. 3 the modifications that resulted from the UC analysis (text in bold italic in the figure), which will be discussed along the paper.

The application of each step of the verification methodology in the medical guideline is described in detail in the following.

### *4.1 Step 1: Identification of entities*

The first step consists in manually identifying the entities that appear in the medical standardization guideline. This is a very simple step to accomplish in any document written in natural language given that entities in a document are usually represented by nouns or noun phrases.

*Example: The entities found in the guideline example are: Patient, Criteria, Area of injury, Medication, Medicine, Doctor, Medicine volume, Estimated medicine volume, Lidocaine 2%, 5ml Syringe, Insulin needle, Cardinal Point(s) of Injury, Anaesthetic buttons, 5ml ampoule of Glucantime, Glucantime, 25 x 0.7G needle, Center of Injury, Bevel, Swelling/Edema, Saturation of injury.*

### *4.2 Step 2: Identification of actions*

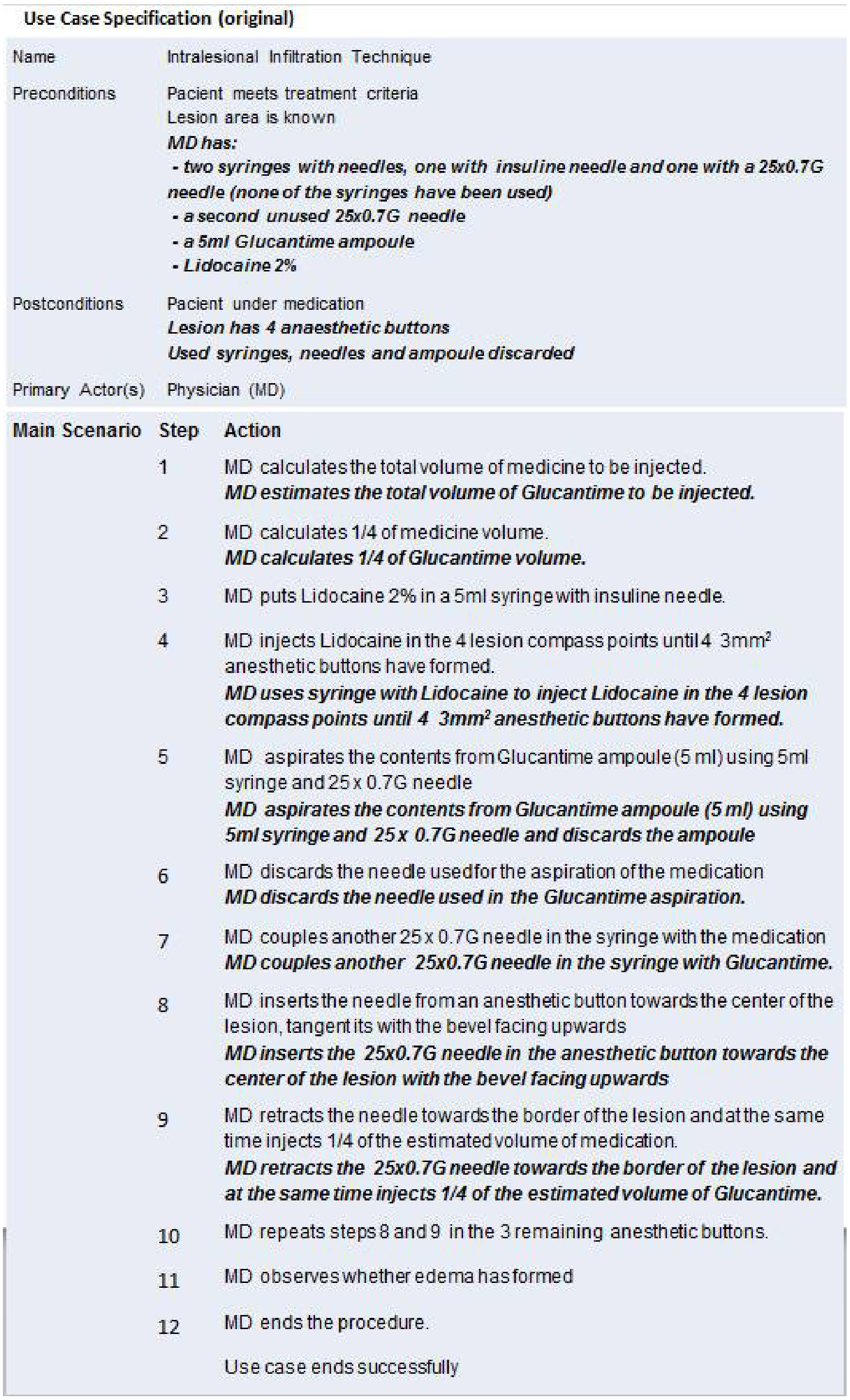
Once the entities used in the document were listed, the next step is to construct a *Table of Actions* (see table 2) by manually identifying the actions that have to be performed as well as the actors involved, the conditions for each action to be applied and the effects of applying the actions. Actions are usually represented in a document by the verbs and verb phrases.

*Example: The actions found in the example were summarized as: (1) calculate volume, (2) divide volume, (3) aspire lidocaine, (4) inject lidocaine, (5) aspirate glucantime, (6) discard needle, (7) couple new needle, (8) insert needle in anaesthetics buttons, (9) administer glucantime, (10) observe saturated injury, (11) observe not saturated injury, (12) administer more glucantime.*

### *4.3 Step 3: Characterization of conditions and effects as states*

After we extracted the entities and actions of the guideline, we define graphical representations for each one of the conditions and effects listed in the *Table of Actions*, as well as the pre- and post-conditions of the UC if any.

*Example: Consider the action "calculate volume of the medicine". Two conditions must hold for this action to execute: "The area of the injury is known" and "Patient meets criteria" (the preconditions of the guideline). This action has also one effect: "MD knows the estimated volume of the medicine". We represent the phrase "The area of the injury is known" using the entities* physician (MD) *and* area of injury*, and a relationship indicating that the MD* knows *the injury's area, as shown in the left side of Fig. 4. The other two states of the first action are represented similarly in the same figure. The same kind of characterization is made for each phrase of each action listed in Step 2.*



**Use Case Specification (original)**

| | |
|---|---|
| Name | Intralesional Infiltration Technique |
| Preconditions | Pacient meets treatment criteria<br>Lesion area is known<br>***MD has:***<br>***- two syringes with needles, one with insuline needle and one with a 25x0.7G needle (none of the syringes have been used)***<br>***- a second unused 25x0.7G needle***<br>***- a 5ml Glucantime ampoule***<br>***- Lidocaine 2%*** |
| Postconditions | Pacient under medication<br>***Lesion has 4 anaesthetic buttons***<br>***Used syringes, needles and ampoule discarded*** |
| Primary Actor(s) | Physician (MD) |

| Main Scenario | Step | Action |
|---|---|---|
| | 1 | MD calculates the total volume of medicine to be injected.<br>***MD estimates the total volume of Glucantime to be injected.*** |
| | 2 | MD calculates 1/4 of medicine volume.<br>***MD calculates 1/4 of Glucantime volume.*** |
| | 3 | MD puts Lidocaine 2% in a 5ml syringe with insuline needle. |
| | 4 | MD injects Lidocaine in the 4 lesion compass points until 4 3mm$^2$ anesthetic buttons have formed.<br>***MD uses syringe with Lidocaine to inject Lidocaine in the 4 lesion compass points until 4 3mm$^2$ anesthetic buttons have formed.*** |
| | 5 | MD aspirates the contents from Glucantime ampoule (5 ml) using 5ml syringe and 25 x 0.7G needle<br>***MD aspirates the contents from Glucantime ampoule (5 ml) using 5ml syringe and 25 x 0.7G needle and discards the ampoule*** |
| | 6 | MD discards the needle used for the aspiration of the medication<br>***MD discards the needle used in the Glucantime aspiration.*** |
| | 7 | MD couples another 25 x 0.7G needle in the syringe with the medication<br>***MD couples another 25x0.7G needle in the syringe with Glucantime.*** |
| | 8 | MD inserts the needle from an anesthetic button towards the center of the lesion, tangent its with the bevel facing upwards<br>***MD inserts the 25x0.7G needle in the anesthetic button towards the center of the lesion with the bevel facing upwards*** |
| | 9 | MD retracts the needle towards the border of the lesion and at the same time injects 1/4 of the estimated volume of medication.<br>***MD retracts the 25x0.7G needle towards the border of the lesion and at the same time injects 1/4 of the estimated volume of Glucantime.*** |
| | 10 | MD repeats steps 8 and 9 in the 3 remaining anesthetic buttons. |
| | 11 | MD observes whether edema has formed |
| | 12 | MD ends the procedure. |
| | | Use case ends successfully |

Fig. 3. Standard guideline using a UC notation - original version and updates (italic bold text)

| **Action** | **Actor** | **Entities Involved** | **Conditions** | **Effects** |
|---|---|---|---|---|
| calculate volume | Doctor | Area of injury | • The area of the injury is known<br>• Patient meets criteria | MD knows the estimated volume of the medicine |
| divide volume | Doctor | Estimated medicine volume | • MD knows the estimated volume of the medicine | The estimated medicine volume is divided in four parts |
| aspire lidocaine | Doctor | Lidocaine 2%, 5ml Syringe, Insulin needle | • MD knows the estimated volume of the medicine<br>• The 5ml syringe has has an insulin needle<br>• MD has lidocaine 2% | 5ml syringe has lidocaine 2% |
| inject lidocaine | Doctor | Lidocaine 2%, Cardinal Points, Anesthetic buttons | • 5ml syringe has lidocaine 2%<br>• The injury has four cardinal points | Injury has four anesthetic buttons |
| ... | ... | ... | ... | ... |

Table 2
Table of Actions

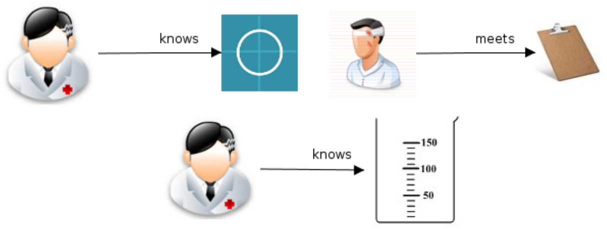


Fig. 4. Graph representation of Phrases of the UC

## *4.4 Step 4: Construction of type graph*

Having the graph representations of all elements of the guideline, we build a type graph that is a graph containing all types of elements that may appear in the UC. First we add all entities identified in Step 1: these will be the vertices of the type graph; and then we add the relationships created in Step 3: these are the arrows of the type graph. In addition to the entities and relationships, the type graph also represents the multiplicities, i. e. how many instances of one entity/relationship can exist in some state.
*Example: The type graph constructed for the example can be seen in Figure 5.*

## *4.5 Step 5: Construction of rules*

This step consists in creating transition rules for each action listed in Step 2. A rule for an action is modeled as a transition between two states, where the state in the Left Hand Side (LHS) of the rule represents the conditions that must be true for this action to occur and the one in the Right Hand Side (RHS) represents the

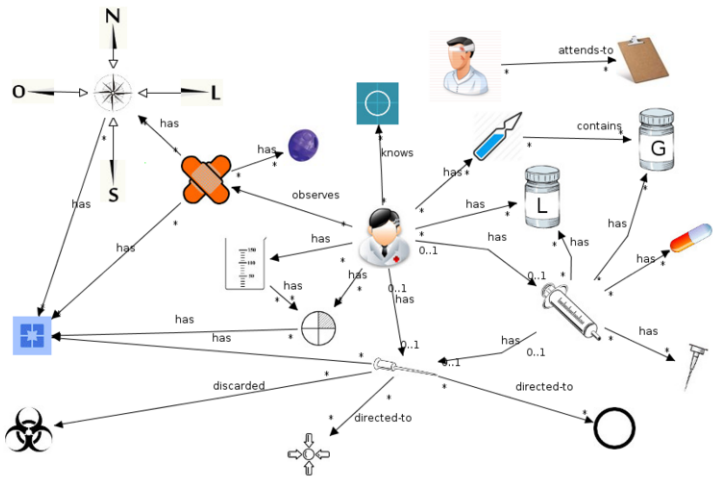


Fig. 5. Initial Type Graph of the guideline

result of the rule application.

The application of a rule $r : LHS \rightarrow RHS$ has following effect:

(i) items in the $LHS$ that do not appear in the $RHS$ are deleted;

(ii) items in the $LHS$ that appear in $RHS$ are preserved;

(iii) items in the $RHS$ that do not appear in the $LHS$ are created.

*Example: Figure 6 shows the action calculateVolume modeled as a rule using the graphical types defined in Step 3.*

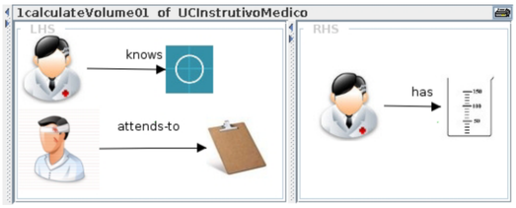


Fig. 6. Modeling the first rule of the guideline: First version

*It is important to notice that one effect of applying the rule in Fig. 6 is to delete the elements* area of injury*,* Patient *and* criteria *(because they are in LHS but not in RHS). However, this is not exactly the intended behavior since we would expect to treat the same injury of the same patient during the treatment instead of destroying them and treating new ones. Therefore, we modified the rule to preserve these elements.* (Figure 7).

*After this modification nothing is destroyed by the rule, once all elements that appear in the $LHS$ also appear in the $RHS$. This leads however to a situation where the rule can be applied indefinitely many times once enabled, because the same conditions that hold before the rule application still hold after it (the effect of multiple applications of this rule would be the existence of more than one* estimated volume of medicine*). One simple way to solve this problem, that happens when the entire $LHS$ is preserved, is the use of a Negative Application Condition ($NAC$) to specify*

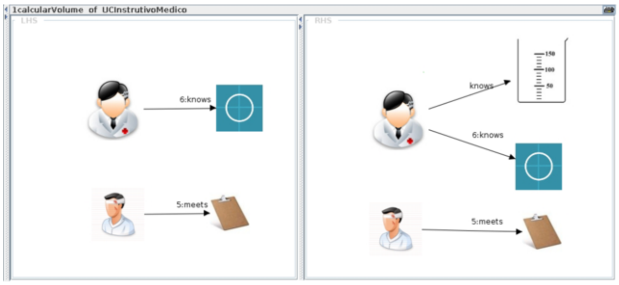


Fig. 7. Modeling the first rule of the guideline: Second version.

*a forbidden context for rule application. To construct the $NAC$ of a rule, the $LHS$ is copied and the forbidden elements are added to it. In our example, the creation of the $NAC$ is made by adding to the $LHS$ the* estimated volume of medicine*, which means that for calculate the volume of medicine the doctor must not previously know the volume. The construction of this $NAC$ results in the rule shown in Fig. 8. Other rules are created analogously (see appendice A).*

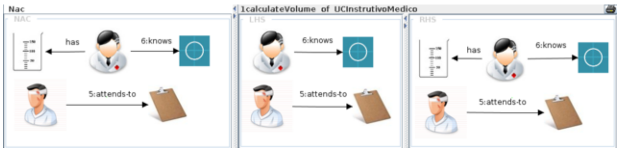


Fig. 8. Modeling the first rule of the guideline: Last version

In short, the construction of a rule for an action can be summarized as:

1 Add the conditions of the action in the $LHS$ of the rule and the effects in the $RHS$.

2 Check whether there are any elements in the $LHS$ that are not in the $RHS$ but should not be destroyed and add them to the $RHS$ as well.

3 Check if the entire $LHS$ is preserved in the $RHS$ and, if needed, create a $NAC$ to avoid the rule to be indefinitely applicable over itself.

## *4.6 Step 6: Analysis*

Having all those artefacts built in the previous steps, we performed a series of automatic analysis using the AGG tool [9] to detect problems in the guideline. The analysis threw Open Issues (OIs) regarding either some aspect of the guideline or the modeling. In the first case we have an indication of where to improve the guideline text. In the second one, an indication to change the model of the guideline. The complete list of possible OIs can be seen at table 1. Currently, three kinds of analysis are provided:

**Conflict analysis:** This type of analysis technique tells us which steps are mutually exclusive, that is, it highlights the choice points of the system by showing which rules are disabled once another one is applied. The result of the conflict (critical-pair) analysis can be seen as a graph, shown in Fig. 9, where conflicts between UC steps are depicted as solid red arrows.

**Dependency analysis:** Similarly to conflict analysis, (potential) dependency analysis identifies relationships between rules (which rules become applicable after some other rule has been applied) and can be used to check whether the dependencies that are intuitively expected to occur are actually there. In Figure 9, dependencies are shown as blue dashed arrows.

**Concurrent rule:** To build a concurrent rule, we first define a *rule sequence (RS)* for each execution path of the UC, where a RS represents the order of execution of all steps of a path. Based on the RS, we build a single rule, called *concurrent rule*, which shows the effect of the whole UC in one step. This concurrent rule allows to check whether the overall effect is really the desired one. Fig. 10 shows the concurrent rule obtained from the execution of the example UC.

Some of the more remarkable OIs in our guideline example were found on dependency analysis and concurrent rules analysis.
*Example: In the steps 5, 6 and 7 of the guideline we have the following actions:*

***(5) aspire glucantime*** *- Doctor aspires the content of the glucantime ampoule (5ml) using a 5ml syringe and a 25 x 0.7 needle.*

***(6) discard needle*** *- Doctor discards the needle used to aspire the medication.*

***(7) couple new needle*** *- Doctor couples a new 25 x 0.7G needle on the syringe that contains the medicine.*

*We would expect to find dependencies between these rules in such a way that it is necessary to apply action 5 in order to apply action 6 and it would be also necessary to apply action 6 in order of action 7 to happen. However, no dependencies were found. The problem in this case is that three entities* Glucantime*,* medication *and* medicine *represent the same object in real world, therefore they are synonyms and only one of them should be used in the text to avoid misunderstandings. We threw an OI and updated both the guideline and the modeling. Running dependency analysis once again, no problems were found regarding these actions.*

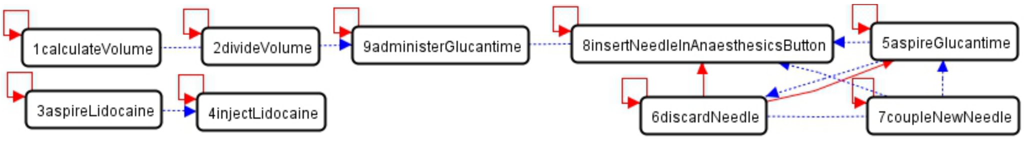


Fig. 9. Graph of Conflicts (straight) and Dependencies (dashed)

*In Fig. 10 we can see the result of building the concurrent rule for the success path of the guideline. Having this concurrent rule we can identify some problems and their possible solutions. In the LHS we have a syringe that has both an insulin needle and another 25x0.7G needle, which is not intended because syringes in real world can only have one needle. Possible solutions for this problem are (1) the two needles represent the same object in the real world, (2) the two needles are different and it is necessary to have a (separate) syringe for each needle or (3) the two needles are different, one of them is initially coupled to the syringe and the other one is not. We can also find entities and relationships that do not appear at all in the*

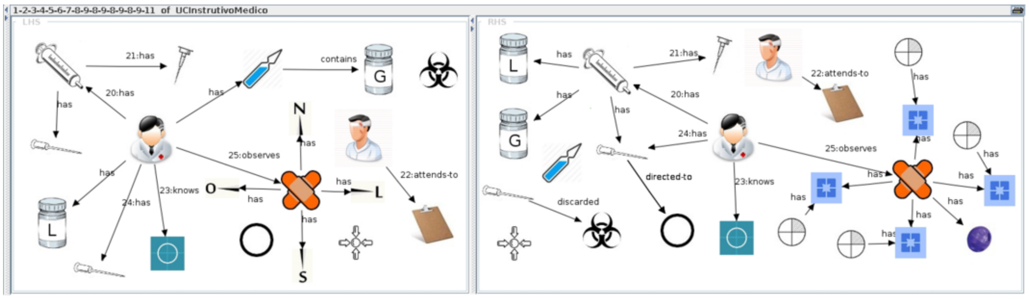


Fig. 10. Concurrent rule for the success path

*guideline preconditions. If they really must be present in the rule, this means the guideline preconditions are not strong enough and important elements are missing. Some possibilities are: the MD has an ampoule which contains Glucantime and MD has Lidocaine 2%.*

*In the RHS there are similar problems. A syringe has both Lidocaine and Glucantime at the same time, which means that the same syringe was used to apply different medicines and, at some point during the treatment, the two medicines were mixed. Since Lidocaine and Glucantime are different objects in real world, the possible solutions for this problem are (1) it is necessary to have two different syringes, one for each medicine or (2) the same syringe is used for both medications, but either all the content of the first is used or what is left is discarded before aspiring the second or (3) the two medications should be explicitly mixed.*

*Other problems can be found analysing the RHS. For example, the ampoule that exists and has no relationships with any other entity. A possible solution would be to explicitly destroy or discard the ampoule. There are elements in the RHS that do not appear in guideline postconditions, like the anaesthetics buttons and the edema of the injury. Similarly to the relation between the LHS and the preconditions we have to check if the elements must be added to the postconditions or explicitly destroyed in some action. We also have again the problem of the syringe with two needles but the solution adopted before solves this issue as well. Thus, the process of executing the analyses and throwing OIs resulted in changing the guideline and/or the modeling. The improved guideline resulting from the verification methodology can be seen in bold-italic text at Fig. 3.*

## *4.7 Evolution of the Treatment guideline*

In summary, after a first round of analysis of the CL treatment model, a total of 13 issues were raised and required modifications in the model and/or in the guideline. Many of these first issues were related to ambiguities present in the text of the original script, as detailed in the previous section. After a second verification round, 7 new issues were raised and the original script was modified accordingly once again. Most issues raised in this round were related to omission of information. Those issues were resolved by the authors of the guideline, since they are related to technical knowledge. In most cases, errors of omission occur because a certain tacit or technical knowledge is assumed in the document. However, such omissions can lead to problems when this assumption is frustrated, which can happen with less experienced physicians or health agents working under pressure. After the second round of analysis, the text of the guideline became clearer and the verification process raised issues more related to the procedure itself. Thus, for instance, the original guideline assumes a typical CL ulcer which has an oval shape. The model analysis raised a question about possible different scenarios such as different ulcer sizes and shapes. Even though a physician is supposed to identify and adjust the procedure accordingly, leaving these scenarios out of the scope of the guideline results in non-determinism in the application of the procedure, which goes against the proposed standardization. The modified script was revised by its authors and, based on their feedback about some of the issues raised during the third round of the verification process, the original 1-page guideline was transformed into a more detailed script composed of four parts: 1) Initial procedure: lesion analysis; 2) Basic procedure: infiltration in small lesions ($\leq$1cm); 3) Basic procedure: infiltration in large lesions ($>$1cm); and 4) Additional procedure: additional infiltration in case of non-saturated lesions. This modified guideline is now being re-modeled and re-verified. Thus, after 3 verification rounds, we achieved a more detailed treatment guideline that covers many possible scenarios that can be faced by the physician during the intervention. For each scenario, the standard procedure is clearly described leaving very small room for variation.

## 5 Conclusions

In this paper, we presented the application of a methodology based on GT formalism for the verification of a *medical standardization guideline.* A UC generated from the medical guideline was first modeled as a graph transformation system and submitted to a series of analysis and then updated according to the results. The results were considered promising, since it was possible to identify a considerable number of problems that could otherwise remain unknown and might compromise the applicability of the guideline. The formal verification lead to an updated UC that is less ambiguous than the original one, increasing its reliability. Thus, this work shows that the methodology has great versatility and may be used in other fields outside software engineering.

The methodology has also the potential to decrease costs and prevent accidents because a number of problems can be detected very early, rather than in later phases phases such as development or validation. In the context of the medical guideline, it prevents the misunderstanding of the guideline and the possible misapplication of the treatment which might invalidate the whole trial and cause damage to the patients.

This work has also enhanced the reliability of the methodology as all the *Open Issues* found were accepted as problems by Fiocruz and corrected to improve the guideline. An interesting future work is to develop a software tool to automate the steps of the methodology. This will help the designer of the UC (or any other UC-like document) to build a formal model. Such tool would increase the scope of use of this process to other fields, allowing the methodology to be tested and improved by the users.

## Acknowledgement

We thank Dr. Gláucia Cota from Centro de Pesquisas René Rachou (Fiocruz-MG) for providing and discussing the health assessment tool used as case study in this work.

## References

[1] Barrett, M.P., Croft, S.L.: Management of trypanosomiasis and leishmaniasis. In: British Medical Bulletin, vol. 104, pp. 175—196 (2012)

[2] Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edn. (2000)

[3] Costello, E., Plack, M., Maring, J.: Validating a standardized patient assessment tool using published professional standards. In: Journal of Physical Therapy Education. vol. 25, pp. 30—45. Elsevier (2003)

[4] Ehrig, H., Engels, G., Kreowski, H.J., Rozenberg, G. (eds.): Handbook of graph grammars and computing by graph transformation: volume II: Applications, languages, and tools. World Scientific, River Edge, USA (1999)

[5] Guimond, P., Bunna, H., O'Connora, A.M., Jacobsen, M.J., Taitc, V.K., Drakec, E.R., Grahamc, I.D., Stacey, D., Elmslie, T.: Validation of a tool to assess health practitioners' decision support and communication skills. In: Patient Education and Counseling, vol. 50, pp. 235—245. Elsevier (2003)

[6] Oliveira Junior, M., Ribeiro, L., Cota, É., Duarte, L.M., Nunes, I., Reis, F.: Use Case Analysis Based on Formal Methods: An Empirical Study. Lecture Notes in Computer Science: Recent Trends in Algebraic Development Techniques, Springer International Publishing Switzerland (2015)

[7] Rozenberg, G. (ed.): Handbook of graph grammars and computing by graph transformation: volume I: Foundations. World Scientific, River Edge, USA (1997)

[8] Taentzer, G.: AGG: A tool environment for algebraic graph transformation. In: Applications of Graph Transformations with Industrial Relevance, LNCS, vol. 1779, pp. 481–488. Springer Berlin Heidelberg (2000)

[9] Taentzer, G.: AGG: A tool environment for algebraic graph transformation. In: Applications of Graph Transformations with Industrial Relevance, pp. 481–488. Springer (2000)

# Appendices

## A Rules for the success scenario of the UC

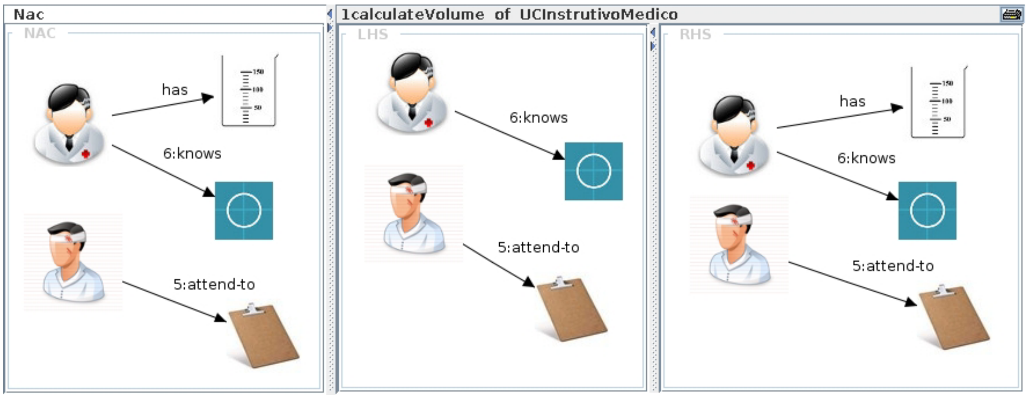


Fig. 11. Rules for the success scenario of the UC

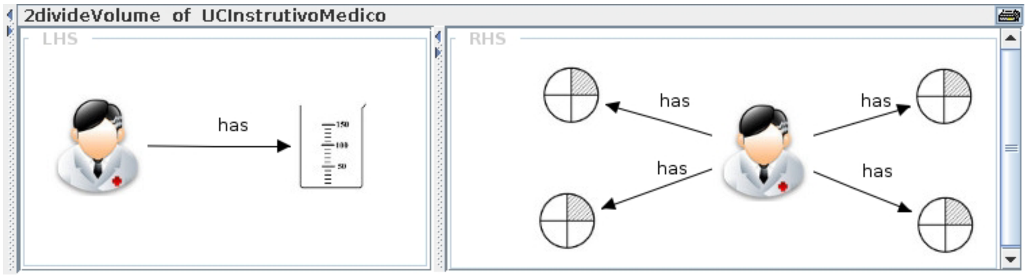


Fig. 12. Rules for the success scenario of the UC

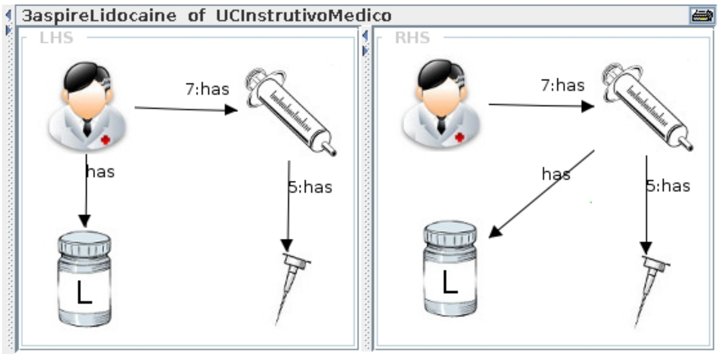


Fig. 13. Rules for the success scenario of the UC

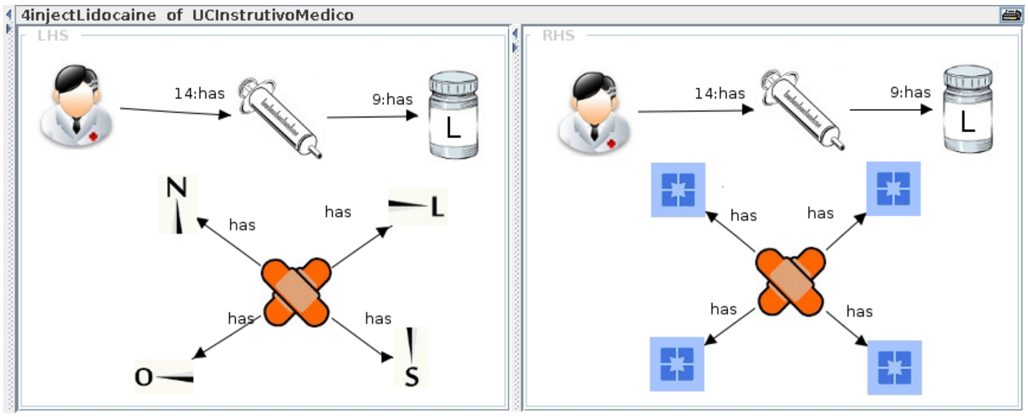


Fig. 14. Rules for the success scenario of the UC

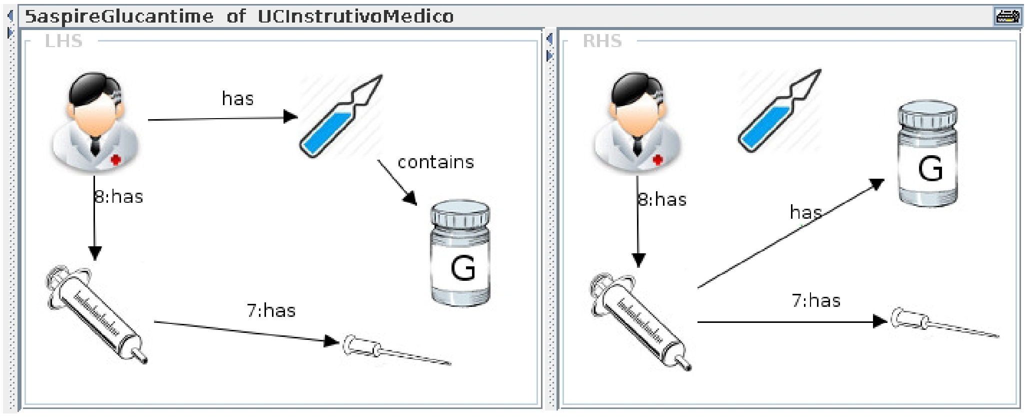


Fig. 15. Rules for the success scenario of the UC

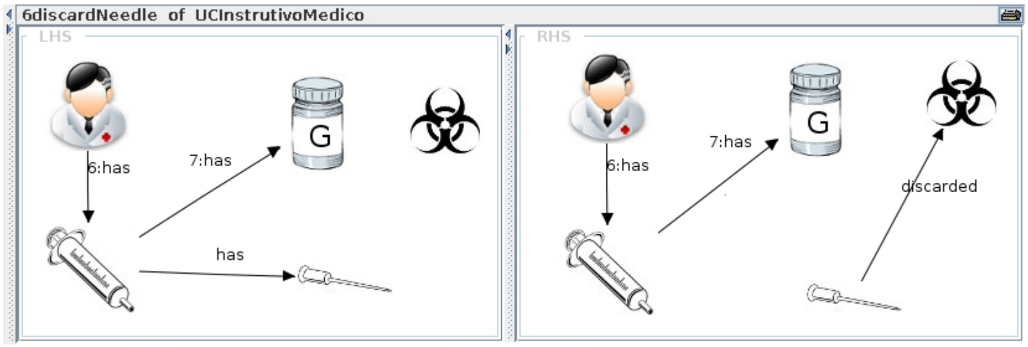


Fig. 16. Rules for the success scenario of the UC

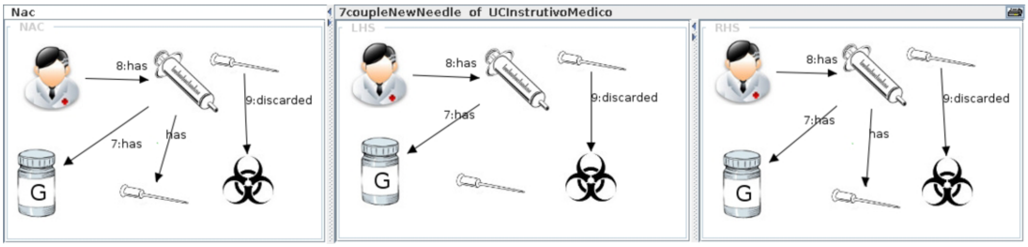


Fig. 17. Rules for the success scenario of the UC

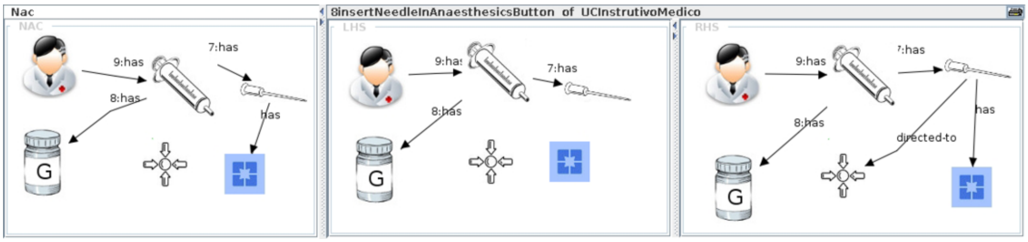


Fig. 18. Rules for the success scenario of the UC

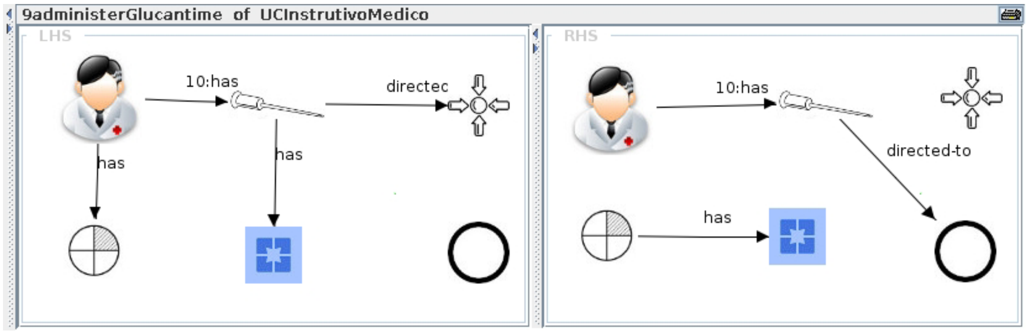


Fig. 19. Rules for the success scenario of the UC

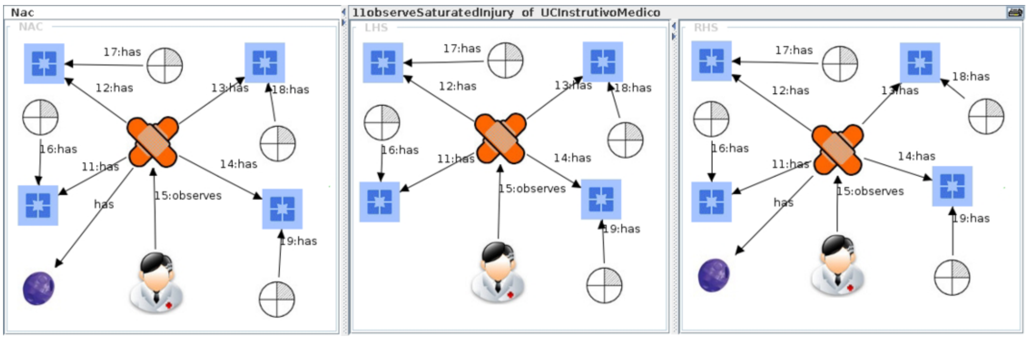


Fig. 20. Rules for the success scenario of the UC