Received October 1, 2018, accepted October 19, 2018, date of publication October 29, 2018, date of current version November 30, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2878311

Applied Graph Transformation and Verification With Use Cases in Malaria Surveillance

JON HAËL BRENAS[1], **MARTIN STRECKER**[2], **RACHID ECHAHED**[3], **AND ARASH SHABAN-NEJAD**[1]

[1]Oak Ridge National Laboratory, Center for Biomedical Informatics, Department of Pediatrics, University of Tennessee Health Science Center, Memphis, TN 38163, USA
[2]IRIT Institute, Université de Toulouse, 31000 Toulouse, France
[3]CNRS and the Laboratoire d'Informatique de Grenoble, Université Grenoble Alpes, Grenoble, France

Corresponding author: Arash Shaban-Nejad (ashabann@uthsc.edu)

This work was supported by the Bill and Melinda Gates Foundation under Grant OPPID 1162018.

ABSTRACT Malaria is one of the leading causes of death and illness in sub-Saharan Africa. In order to make timely decisions for control and elimination of malaria, researchers, and clinicians need access to integrated consistent knowledge sources. These knowledge sources often rely on dynamic and constantly changing databases and ontologies. It is crucial to manage changes and ensure that these changes do not cause inconsistencies in the integrated knowledge source. To this end, we propose the use of a formal model using graph transformations to monitor and manage the changes in a coherent way while preserving the consistency of the integrated structure through classical verification. In this paper, we use an algorithmic approach to graph transformation, instead of the more classical algebraic approach, to express the evolution of the data and ontological structures. In this model, each transformation is the result of applying rules to the graph, where the left-hand side is used to select a subgraph and the right-hand side is a sequence of elementary actions to be performed. Strategies are used to define how transformation rules should be applied. This approach enables us to define a Hoare-like calculus that can be used to verify the transformations and manage the changes. In this paper, we demonstrate the feasibility and significance of the proposed method through different use cases in malaria surveillance.

INDEX TERMS Graph transformation, verification, malaria surveillance, ontologies, change management.

I. INTRODUCTION

Malaria, an infectious disease disproportionately affecting low-income developing countries [1], was responsible for around 445,000 deaths worldwide in 2016 [2], mostly young children in Sub-Saharan Africa. Parasitic micro-organisms of the Plasmodium species responsible for many malaria cases are transmitted through mosquito bites. In order to efficiently perform malaria surveillance, many factors need to be taken into account, for example, the vector ecology, the climate, human behaviors [3], uses of malaria drugs, urbanization and public health infrastructures [4].

Ontologies capture the domain knowledge by defining concepts, relationships, individuals, rules, and axioms. Several ontologies, such as the Ontology for Vector Surveillance and Management (VSMO) [5], the Malaria Ontology (IDOMAL) [6], [7], the Mosquito Insecticide Resistance Ontology (MIRO) [8], [9], have been specifically created to deal with malaria and similar diseases. No single ontology covers all the aspects required for an effective malaria surveillance. So, the malaria community needs to integrate many different types of databases and ontologies and maintain the integrated structure to ensure data availability, integrity, and reliability. One of the major challenges of maintaining such integrated infrastructures is dealing with changes in their components. Such changes [10] may cause inconsistencies in the knowledge or break the interoperability with existing sources and tools. One of the main elements of a successful change management process is the ability to verify and prove that the changes propagate in the right direction.

The concept of verification is far from new [11], [12] but the knowledge of the structure on which to reason is capital, i.e., one must know the language that is used to represent the changes in order to be able to reason about them. This is currently difficult as the languages used to represent and express database modifications (e.g., SPARQL [13]) are different from the languages used to express the changes

This work is licensed under a Creative Commons Attribution 3.0 License. For more information, see http://creativecommons.org/licenses/by/3.0/

in ontologies. We use graphs as a common standard representation vehicle for data structures and ontologies and graph transformations as a formalism to express all the potential changes and modifications.

Graphs already play a central role in modeling data structures and graph transformations facilitate reasoning about a wide range of different structures. The feasibility and significance of using graph transformation and categorical logical approach for studying ontology evolution and change management have been extensively discussed in the literature [14]–[16]. Studies on the correctness of graph transformations [17]–[24] have also produced promising results. Throughout this paper, we use the formal definitions described at [25] and demonstrate the applicability of the graph transformation and verification to respond to pressing salient issues such as change management and maintaining interoperability in a real-world application to improve global health surveillance. Our approach differs from many of the previously mentioned in that it offers more flexibility on the choice of the logic used to describe the graphs. Furthermore, it uses an algorithmic approach that is more intuitive and accessible for non-expert users, e.g., the users might not be totally aware of the distinction between single- and double-pushout presented in other methods. We refer interested readers to more theoretical works, e.g., [25] for more thorough comparisons of different approaches.

In this paper, we choose a Hoare-like calculus to address the problem of the correctness of programs defined as strategy expressions over graph rewrite rules. Specifications are defined as triples of the form `{Pre}(R, strategy){Post}` where `Pre` and `Post` are conditions, `R` is a graph rewrite system and `strategy` is an expression stating how rules in `R` are to be performed.

We consider the transformations in the form of rewrite rules that follow an algorithmic approach where the left-hand sides are attributed graphs and the right-hand sides are sequences of elementary actions [26]. We study a wide range of actions such as node creation, edge relabelling and redirection, or merging. An important aspect of our approach is that graphs are labeled using logical formulae. We present some of the logics that can be used in our application. All of them are fragments of first-order logic. We present several logics because different problems can be better expressed in different logics. A particular focus is on Description Logics [27], which are the de facto standard to express ontologies with good balance between the expressiveness of the specification and the complexity of the verification.

Our goal is to present a novel approach for verifying transformations and changes in malaria surveillance data sources. Any responsible body who performs such transformations, e.g., a data collector that adds their most recent results or a knowledge engineer modifying an ontology, can use our algorithm to prove that the transformations work according to the specifications, provided the transformation and specifications are expressed in a way consistent with our algorithm. Furthermore, if a set of defined conditions on the transformations are satisfied, the correctness of the specification will be considered decidable. When the specification is incorrect, the algorithm returns a counter-model that can be used to refine the specification.

The paper is organized as follows. Section II introduces some preliminary definitions of graphs and the elementary graph transformation actions. Section III presents the logics that we use throughout this study. Section IV explains how to define specifications for the transformations and how graph verification is performed. Section V showcases applications of verification to certain types of changes in the real world malaria surveillance database or the ontologies. Finally, in Section VI we conclude with a discussion of our findings and results, and suggestions for future work.

II. GRAPHS AND TRANSFORMATIONS

We start by introducing the notion of *logically decorated graphs*. Nodes and edges of such graph structures are labeled by logic formulas. The definition below is parameterized by a given logic $\mathcal{L}$ seen as a set of formulas. Section III provides some examples of possible candidates for such a logic $\mathcal{L}$.

Definition 1 (Logically Decorated Graph): Let $\mathcal{L}$ be a logic (set of formulas). A graph alphabet *is a pair $(\mathcal{C}, \mathcal{R})$ of sets of elements of $\mathcal{L}$, that is $\mathcal{C} \subseteq \mathcal{L}$ and $\mathcal{R} \subseteq \mathcal{L}$. $\mathcal{C}$ is the set of* node formulas *or* concepts *and $\mathcal{R}$ is the set of edge formulas or* roles.[1] *Subsets of $\mathcal{C}$ and $\mathcal{R}$, respectively named $\mathcal{C}_0$ and $\mathcal{R}_0$, contain basic (propositional) concepts and roles respectively. A* logically decorated graph *G over a graph alphabet $(\mathcal{C}, \mathcal{R})$ is a tuple $(N, E, \Phi_N, \Phi_E, s, t)$ where N is a set of* nodes, *E is a set of* edges, *Φ_N is the* node labeling *function, $\Phi_N : N \to \mathcal{P}(\mathcal{C})$, Φ_E is the* edge labeling *function, $\Phi_E : E \to \mathcal{R}$, s is the* source function *$s : E \to N$ and t is the* target function *$t : E \to N$.*

Transformation of logically decorated graphs is defined following an algorithmic approach based on the notion of *elementary actions* as introduced below. These actions constitute a set of elementary graph transformations such as the addition/deletion of nodes, concepts or edges; redirection of edges; merging or cloning of nodes. Formal definitions of the defined elementary actions are given in Fig. 1.

Definition 2 (Elementary Action, Action): An elementary action, *say a, may be of the following forms:*

- *a* node addition *$add_N(i)$ (resp.* node deletion *$del_N(i)$) where i is a new node (resp. an existing node). It creates the node i. i has no incoming nor outgoing edge and it is not labeled with any concept (resp. it deletes i and all its incoming and outgoing edges).*
- *a* concept addition *$add_C(i, c)$ (resp.* concept deletion *$del_C(i, c)$) where i is a node and c is a basic concept (a proposition name) in $\mathcal{C}_0$. It adds the label c to (resp. removes the label c from) the labeling of node i.*
- *an* edge addition *$add_E(e, i, j, r)$ (resp.* edge deletion *$del_E(e, i, j, r)$) where e is an edge, i and j are nodes and*

[1]The terms *concept* and *role* are borrowed from the Description Logics' vocabulary [27].

If $\alpha = add_C(i,c)$ **then:**

$$N^{G'} = N^G, E^{G'} = E^G,$$

$$\Phi_N^{G'}(n) = \begin{cases} \Phi_N^G(n) \cup \{c\} & \text{if } n = i \\ \Phi_N^G(n) & \text{if } n \neq i \end{cases}$$

$$\Phi_E^{G'} = \Phi_E^G,\ s^{G'} = s^G,\ t^{G'} = t^G$$

If $\alpha = add_E(i,j,r)$ **then:**

$$N^{G'} = N^G,\ \Phi_N^{G'} = \Phi_N^G$$

$$E^{G'} = E^G \cup \{e\}$$

$$\Phi_E^{G'}(e') = \begin{cases} r & \text{if } e' = e \\ \Phi_E^G(e') & \text{if } e' \neq e \end{cases}$$

$$s^{G'}(e') = s^G(e') \text{ if } e' \neq e, s^{G'}(e) = i$$

$$t^{G'}(e') = t^G(e') \text{ if } e' \neq e, t^{G'}(e) = j$$

If $\alpha = add_N(i)$ **then:**

$$N^{G'} = N^G \cup \{i\} \text{ where } i \text{ is a new node}$$

$$E^{G'} = E^G,\ \Phi_E^{G'} = \Phi_E^G, s^{G'} = s^G,\ t^{G'} = t^G$$

$$\Phi_N^{G'}(n) = \begin{cases} \emptyset & \text{if } n = i \\ \Phi_N^G(n) & \text{if } n \neq i \end{cases}$$

If $\alpha = del_N(i)$ **then:**

$$N^{G'} = N^G \setminus \{i\}$$

$$E^{G'} = E^G \setminus \{e | s^G(e) = i \vee t^G(e) = i\}$$

$\Phi_N^{G'}$ is the restriction of Φ_N^G to $N^{G'}$

$\Phi_E^{G'}$ is the restriction of Φ_E^G to $E^{G'}$

$s^{G'}$ is the restriction of s^G to $E^{G'}$

$t^{G'}$ is the restriction of t^G to $E^{G'}$

If $\alpha = i \gg j$ **then:**

$$N^{G'} = N^G,\ E^{G'} = E^G$$

$$\Phi_N^{G'} = \Phi_N^G,\ \Phi_E^{G'} = \Phi_E^G, s^{G'} = s^G$$

$$t^{G'}(e) = \begin{cases} j & \text{if } t^G(e) = i \\ t^G(e) & \text{if } t^G(e) \neq i \end{cases}$$

If $\alpha = mrg(i,j)$ **then:**

$$N^{G'} = N^G \setminus \{j\}, E^{G'} = E^G, \Phi_E^{G'}(e) = \Phi_E^G(e)$$

$$\Phi_N^{G'}(n) = \begin{cases} \Phi_N^G(i) \cup \Phi_N^G(j) & \text{if } n = i \\ \Phi_N^G(n) & \text{otherwise} \end{cases}$$

$$s^{G'}(e) = \begin{cases} i & \text{if } s^G(e) = j \\ s^G(e) & \text{otherwise} \end{cases}$$

$$t^{G'}(e) = \begin{cases} i & \text{if } t^G(e) = j \\ t^G(e) & \text{otherwise} \end{cases}$$

If $\alpha = del_C(i,c)$ **then:**

$$N^{G'} = N^G, E^{G'} = E^G,$$

$$\Phi_N^{G'}(n) = \begin{cases} \Phi_N^G(n) \setminus \{c\} & \text{if } n = i \\ \Phi_N^G(n) & \text{if } n \neq i \end{cases}$$

$$\Phi_E^{G'} = \Phi_E^G,\ s^{G'} = s^G,\ t^{G'} = t^G$$

If $\alpha = del_E(i,j,r)$ **then:**

$$N^{G'} = N^G,\ \Phi_N^{G'} = \Phi_N^G$$

$$E^{G'} = E^G \setminus \{e | s^G(e) = i, t^G(e) = j \Phi_E^G(e) = r\}$$

$\Phi_E^{G'}$ is the restriction of Φ_E^G to $E^{G'}$

$s^{G'}$ is the restriction of s^G to $E^{G'}$

$t^{G'}$ is the restriction of t^G to $E^{G'}$

If $\alpha = cl(i,j,L_{in},L_{out},L_{l\_in},L_{l\_out},L_{l\_loop})$ **then:**

$$N^{G'} = N^G \cup \{j\}$$

$$E^{G'} = E^G \cup E'_{in} \cup E'_{out} \cup E'_{l\_in} \cup E'_{l\_out} \cup E'_{l\_loop}$$

$$\Phi_N^{G'}(n) = \begin{cases} \Phi_N^G(i) \cap \mathcal{C}_0 & \text{if } n = j \\ \Phi_N^G(n) & \text{otherwise} \end{cases}$$

$$\Phi_E^{G'}(e) = \begin{cases} \Phi_E^G(in(e)) & \text{if } e \in E'_{in} \\ \Phi_E^G(out(e)) & \text{if } e \in E'_{out} \\ \Phi_E^G(l\_in(e)) & \text{if } e \in E'_{l\_in} \\ \Phi_E^G(l\_out(e)) & \text{if } e \in E'_{l\_out} \\ \Phi_E^G(l\_loop(e)) & \text{if } e \in E'_{l\_loop} \\ \Phi_E^G(e) & \text{otherwise} \end{cases}$$

$$s^{G'}(e) = \begin{cases} s^G(in(e)) & \text{if } e \in E'_{in} \\ j & \text{if } e \in E'_{out} \\ i & \text{if } e \in E'_{l\_in} \\ j & \text{if } e \in E'_{l\_out} \\ j & \text{if } e \in E'_{l\_loop} \\ s^G(e) & \text{otherwise} \end{cases}$$

$$t^{G'}(e) = \begin{cases} j & \text{if } e \in E'_{in} \\ t^G(out(e)) & \text{if } e \in E'_{out} \\ j & \text{if } e \in E'_{l\_in} \\ i & \text{if } e \in E'_{l\_out} \\ j & \text{if } e \in E'_{l\_loop} \\ t^G(e) & \text{otherwise} \end{cases}$$

FIGURE 1. $G' = G[\alpha]$**, summary of the effects of the elementary actions:** $add_N(i)$, $del_N(i)$, $add_C(i,c)$, $del_C(i,c)$, $add_E(e,i,j,r)$, $del_E(e)$, $i \gg j$, $mrg(i,j)$ **and** $cl(i,j,L_{in},L_{out},L_{l\_in},L_{l\_out},L_{l\_loop})$. $\mathcal{C}$ **and** $\mathcal{R}$ **are never modified. More details can be found in [25].**

r is a basic role (edge label) in $\mathcal{R}_0$. *It adds the edge e with label r between nodes i and j (resp. removes the edge e). When the edge that is affected is clear from the context, we will usually simply write* $add_E(i,j,r)$ *(resp.* $del_E(i,j,r)$*).*

- *a* global edge redirection $i \gg j$ *where i and j are nodes. It redirects all incoming edges of i towards j.*
- *a* merge action $mrg(i,j)$ *where i and j are nodes. This action merges the two nodes. It yields a new graph in which the first node i is labeled with the union of the labels of i and j and such that all incoming or outgoing edges of any of the two nodes are gathered.*
- *a* clone action $cl(i,j,L_{in},L_{out},L_{l\_in},L_{l\_out},L_{l\_loop})$ *where i and j are nodes and* L_{in}, L_{out}, L_{l_in}, L_{l_out} *and* L_{l_loop} *are sets of basic roles. It clones a node i by creating a new node j and connects j to the rest of a host graph according to different information given in the parameters* $L_{in}, L_{out}, L_{l\_in}, L_{l\_out}, L_{l\_loop}$.

The result of performing an elementary action a on a graph $G = (N^G, E^G, C^G, R^G, \Phi_N^G, \Phi_E^G, s^G, t^G)$, *written* $G[\alpha]$, *produces the graph* $G' = (N^{G'}, E^{G'}, C^{G'}, R^{G'}, \Phi_N^{G'}, \Phi_E^{G'}, s^{G'}, t^{G'})$ *as defined in Fig. 1. An* action, *say* α, *is a sequence of elementary actions of the form* $\alpha = a_1; a_2; \ldots; a_n$. *The result of performing* α *on a graph G is written* $G[\alpha]$. $G[a;\alpha] = (G[a])[\alpha]$ *and* $G[\epsilon] = G$ *where* ϵ *is the empty sequence.*

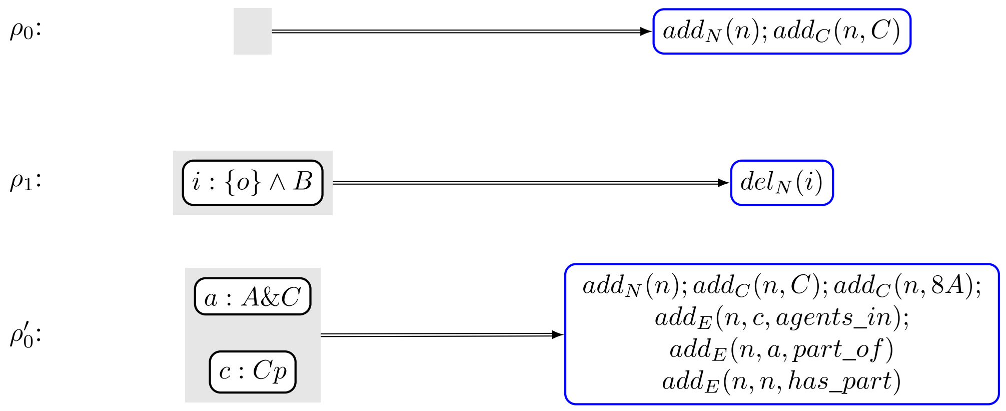


FIGURE 2. Example of graph rewriting system with three rules.

We introduce the notion of *logically decorated graph rewriting systems* (LDRS). These are extensions of the graph rewriting systems defined in [26] where graphs are attributed with formulas from a given logic. The left-hand sides of the rules are thus logically decorated graphs whereas the right-hand sides are defined as sequences of elementary actions.

Definition 3 (Rule, LDRS): A rule *ρ is a pair (LHS, α) where LHS, called the left-hand side, is a logically decorated graph and α, called the right-hand side, is an action. Rules are usually written $LHS \rightarrow \alpha$. A logically decorated graph rewriting system, LDRS, is a set of rules.*

Fig. 2 shows a graph rewriting system consisting of three rules (ρ_0, ρ_1 and ρ'_0). Each rule has a (possibly empty) labeled graph as left-hand side and a sequence of actions as right-hand side.

In order to be able to use graph rewriting systems, one needs to know where a given rule can be applied in a graph. To do so, we introduce the notion of match.

Definition 4 (Match): A match *h between a left-hand side LHS and a graph G is a pair of functions $h = (h^N, h^E)$, with $h^N : N^{LHS} \rightarrow N^G$ and $h^E : E^{LHS} \rightarrow E^G$ such that:*

1. $\forall n \in N^{LHS}, \quad \forall c \in \Phi_N^{LHS}(n), h^N(n) \models c$
2. $\forall e \in E^{LHS}, \quad \Phi_E^G(h^E(e)) = \Phi_E^{LHS}(e)$
3. $\forall e \in E^{LHS}, \quad s^G(h^E(e)) = h^N(s^{LHS}(e))$
4. $\forall e \in E^{LHS}, \quad t^G(h^E(e)) = h^N(t^{LHS}(e))$

The third and the fourth conditions are simply indicating that the source and target functions and the match have to agree. The first condition says that for every node n of the left-hand side, the node to which it is associated, $h(n)$, in G has to satisfy every concept in $\Phi_N^{LHS}(n)$. This condition clearly expresses additional negative and positive conditions which are added to the "structural" pattern matching. The second one ensures that the match respects the edge labeling.

Definition 5 (Rule Application): A graph G rewrites to graph G' using a rule $\rho = (LHS, \alpha)$ iff there exists a match h from LHS to G. G' is obtained from G by performing actions in $h(\alpha)$.[2] *Formally, $G' = G[h(\alpha)]$. We write $G \rightarrow_\rho G'$.*

[2] $h(\alpha)$ is obtained from α by replacing every node name, n, of LHS by $h(n)$.

Rules by themselves are inadequate for building very complex transformations. So, we extend the notion of transformation by using strategies that allow combining rules to perform sophisticated transformations.

Definition 6 (Strategy): Given a graph rewriting system $\mathcal{R}$, a strategy *is a word of the following language defined by s, where ρ is any rule in $\mathcal{R}$:*

$s :=$	ϵ	*(Empty strategy)*	ρ	*(Rule)*
	$s \oplus s$	*(Choice)*	$s; s$	*(Composition)*
	s^*	*(Closure)*	$\rho?$	*(Rule trial)*
	$\rho!$	*(Mandatory Rule)*		

We write $G \Rightarrow_{\mathcal{S}} G'$ to denote that graph G' is obtained from G by applying the strategy $\mathcal{S}$. In Fig. 3, we provide the rules that specify how strategies are used to rewrite a graph. For that we use the following atomic formula $\mathbf{App}(\rho)$ such that for all graphs G, $G \models \mathbf{App}(\rho)$ iff the rule ρ can be applied to G, i.e., iff there exists a match h from the left-hand side of ρ to G. The definition is then extended to strategies.

- $G \models \mathbf{App}(\epsilon)$
- $G \models \mathbf{App}(s_0 \oplus s_1)$ iff
- $G \models \mathbf{App}(s_0)$ or $G \models \mathbf{App}(s_1)$
- $G \models \mathbf{App}(s_0^*)$
- $G \models \mathbf{App}(s_0; s_1)$ iff $G \models \mathbf{App}(s_0)$

Example 7: Let us assume that we want to use the graph rewriting system $\mathcal{R}_0$ of Fig. 2. The strategy $s_0 \equiv \rho_1!; \rho_1?; (\rho_0 \oplus \rho'_0)^*$ consists in applying the rule ρ_1 at least once. ρ_1 is then applied a second time if possible. This is followed by either ρ_0 or ρ'_0 as long as it is possible to find a match for at least one of the two.

III. LOGICS

In the previous section, we have presented the definitions of logically-decorated graphs and how to modify them. Each definition relies on the use of a logic to label edges and nodes. In this section, we present some logics that can be used for such a purpose. We chose to focus on Description Logics, as they have become a de facto standard for representing ontologies, and some fragments of the First-Order Logic, which make the satisfiability problem decidable.

$$\frac{}{G \Rightarrow_{\epsilon} G}\ \text{(Empty rule)}$$

$$\frac{G \Rightarrow_{s_0} G'' \quad G'' \Rightarrow_{s_1} G'}{G \Rightarrow_{s_0;s_1} G'}\ \text{(Strategy composition)}$$

$$\frac{G \Rightarrow_{s_0} G'}{G \Rightarrow_{s_0 \oplus s_1} G'}\ \text{(Choice left)} \qquad \frac{G \Rightarrow_{s_1} G'}{G \Rightarrow_{s_0 \oplus s_1} G'}\ \text{(Choice right)}$$

$$\frac{G \not\models \mathbf{App}(s)}{G \Rightarrow_{s^*} G}\ \text{(Closure false)} \qquad \frac{G \Rightarrow_{s} G'' \quad G'' \Rightarrow_{s^*} G' \quad G \models \mathbf{App}(s)}{G \Rightarrow_{s^*} G'}\ \text{(Closure true)}$$

$$\frac{G \not\models \mathbf{App}(\rho)}{G \Rightarrow_{\rho} \top}\ \text{(Rule False)} \qquad \frac{G \models \mathbf{App}(\rho) \quad G \rightarrow_{\rho} G'}{G \Rightarrow_{\rho} G'}\ \text{(Rule True)}$$

$$\frac{G \not\models \mathbf{App}(\rho)}{G \not\Rightarrow_{\rho!}}\ \text{(Mandatory Rule False)} \qquad \frac{G \models \mathbf{App}(\rho) \quad G \rightarrow_{\rho} G'}{G \Rightarrow_{\rho!} G'}\ \text{(Mandatory Rule True)}$$

$$\frac{G \not\models \mathbf{App}(\rho)}{G \Rightarrow_{\rho?} G}\ \text{(Rule Trial False)} \qquad \frac{G \models \mathbf{App}(\rho) \quad G \rightarrow_{\rho} G'}{G \Rightarrow_{\rho?} G'}\ \text{(Rule Trial True)}$$

FIGURE 3. **Strategy application rules.**

Definition 8 (Concept, Role, $\mathcal{ALC}$): Let $\mathcal{A} = (\mathcal{O}, \mathcal{C}_0, \mathcal{R}_0)$ be an alphabet where $\mathcal{O}$ (resp. $\mathcal{C}_0$, $\mathcal{R}_0$) is the set of nominals (resp. atomic concepts, atomic roles), given $o \in \mathcal{O}$, $C_0 \in \mathcal{C}_0$, $r_0 \in \mathcal{R}_0$ and n and integer, $\mathcal{ALC}$ concepts C and roles R are defined by:

$$\begin{array}{lcll} C & := & \top & \textit{(tautology)} \\ & \mid & C_0 & \textit{(atomic concept)} \\ & \mid & \exists R.C & \textit{(existential quantifier)} \\ & \mid & \neg C & \textit{(negation)} \\ & \mid & C \vee C & \textit{(disjunction)} \\ \\ R & := & r_0 & \textit{(atomic role)} \end{array}$$

$\mathcal{ALC}$ can be extended by adding some of the following concept and role constructors:

$$\begin{array}{lcll} C & := & \{o\} & \textit{(nominals)} \\ & \mid & \exists R.Self & \textit{(self loops)} \\ & \mid & (<\ n\ R\ C) & \textit{(counting quantifiers)} \\ \\ R & := & U & \textit{(universal role)} \\ & \mid & R^- & \textit{(inverse role)} \end{array}$$

For the sake of conciseness, we define:

$$\begin{array}{lcl} \bot & \equiv & \neg\top \\ C \wedge C' & \equiv & \neg(\neg C \vee \neg C') \\ \forall R.C & \equiv & \neg(\exists R.\neg C) \\ (\geq\ n\ R\ C) & \equiv & \neg(<\ n\ R\ C) \end{array}$$

In most Description Logics-based ontologies the knowledge is spread into three different parts: an ABox containing assertions about individual pieces of data, i.e., nodes in a graph representation, a TBox containing assertions about concepts, i.e., classes of nodes, and an RBox containing assertions on roles, i.e., classes of edges/pairs of nodes.

Definition 9 (ABox, TBox, RBox): An ABox is a set of assertions of the form $C(i)$, $R(i,j)$, $\neg R(i,j)$, $i = j$ or $i \neq j$ where i and j are individuals, C is a concept and R is a role.

A TBox is a set of assertions of the form $C \subseteq C'$ where C and C' are concepts.

An RBox is a set of assertions of the form $R \subseteq R'$ or Prop(R) where R and R' are roles and Prop is a role property. Examples of role properties commonly employed in DLs are Transitive, Reflexive, Irreflexive, Symmetric, Asymmetric or Functional.

We now define the semantics of these formulas by defining their interpretations.

Definition 10 (Interpretation): An interpretation over an alphabet $(\mathcal{C}_0, \mathcal{R}_0, \mathcal{O},)$ is a tuple $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\cdot^{\mathcal{I}}$ is a function such that $c_0^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, for every atomic concept $c_0 \in \mathcal{C}_0$, $r_0^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, for every atomic role $r_0 \in \mathcal{R}_0$, $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for every nominal $o \in \mathcal{O}$. The interpretation function is extended to concept and role descriptions by the following inductive definitions:

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
- $(C \vee D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\exists R.C)^{\mathcal{I}} = \{n \in \Delta^{\mathcal{I}} | \exists m, (n,m) \in R^{\mathcal{I}}$ and $m \in C^{\mathcal{I}}\}$
- $(\exists R.Self)^{\mathcal{I}} = \{n \in \Delta^{\mathcal{I}} | (n,n) \in R^{\mathcal{I}}\}$
- $(<\ n\ R\ C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} | \#(\{m \in \Delta^{\mathcal{I}} | (\delta, m) \in R^{\mathcal{I}}$ and $m \in C^{\mathcal{I}}\}) < n\}$
- $(R^-)^{\mathcal{I}} = \{(n,m) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} | (m,n) \in R^{\mathcal{I}}\}$
- $U^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

This definition is extended to ABoxes, TBoxes and RBoxes. It is worth noting that the interpretation of an assertion is a truth value and not a sub-graph.

- $(C(i))^{\mathcal{I}} = i^{\mathcal{I}} \in C^{\mathcal{I}}$
- $(R(i,j))^{\mathcal{I}} = (i^{\mathcal{I}}, j^{\mathcal{I}}) \in R^{\mathcal{I}}$
- $(\neg R(i,j))^{\mathcal{I}} = (i^{\mathcal{I}}, j^{\mathcal{I}}) \notin R^{\mathcal{I}}$
- $(i = j)^{\mathcal{I}} = (i^{\mathcal{I}} = j^{\mathcal{I}})$
- $(i \neq j)^{\mathcal{I}} = (i^{\mathcal{I}} \neq j^{\mathcal{I}})$
- $(C \subseteq C')^{\mathcal{I}} = (C^{\mathcal{I}} \subseteq C'^{\mathcal{I}})$
- $(R \subseteq R')^{\mathcal{I}} = (R^{\mathcal{I}} \subseteq R'^{\mathcal{I}})$
- $(Transitive(R))^{\mathcal{I}} = \top$ *iff* $\forall x, y, z \in \Delta^{\mathcal{I}}.R(x, y)^{\mathcal{I}}$ *and* $R(y, z)^{\mathcal{I}}$ *implies* $R(x, z)^{\mathcal{I}}$
- $(Reflexive(R))^{\mathcal{I}} = \top$ *iff* $\forall x \in \Delta^{\mathcal{I}}.R(x, x)^{\mathcal{I}}$
- $(Irreflexive(R))^{\mathcal{I}} = \top$ *iff* $\forall x \in \Delta^{\mathcal{I}}.\neg R(x, x)^{\mathcal{I}}$
- $(Symmetric(R))^{\mathcal{I}} = \top$ *iff* $\forall x, y \in \Delta^{\mathcal{I}}.R(x, y)^{\mathcal{I}}$ *implies* $R(y, x)^{\mathcal{I}}$
- $(Asymmetric(R))^{\mathcal{I}} = \top$ *iff* $\forall x, y \in \Delta^{\mathcal{I}}.R(x, y)^{\mathcal{I}}$ *implies* $\neg R(y, x)^{\mathcal{I}}$
- $(Functional(R))^{\mathcal{I}} = \top$ *iff* $\forall x \in \Delta^{\mathcal{I}}$ *there exists at most one* $y \in \Delta^{\mathcal{I}}$ *such that* $R(x, y)^{\mathcal{I}}$

Definition 11 (Interpretation Induced by a Decorated Graph): Let $G = (N, E, \Phi_N, \Phi_E, s, t)$ *be a graph over an alphabet* $(\mathcal{C}, \mathcal{R})$ *such that* $\mathcal{C}_0 \cup \mathcal{O} \subseteq \mathcal{C}$ *and* $\mathcal{R}_0 \subseteq \mathcal{R}$. *The interpretation induced by the graph G, denoted* $(\Delta^{\mathcal{G}}, \cdot^{\mathcal{G}})$ *such that* $\Delta^{\mathcal{G}} = N$, $c_0^{\mathcal{G}} = \{n \in N | \, c_0 \in \Phi_N(n)\}$, *for every atomic concept* $c_0 \in \mathcal{C}_0$, $r_0^{\mathcal{G}} = \{(n, m) \in N \times N | \exists e \in E.s(e) = n$ *and* $t(e) = m$ *and* $r_0 = \Phi_E(e)\}$, *for every atomic role* $r_0 \in \mathcal{R}_0$, $o^{\mathcal{G}} = \{n \in N | o \in \Phi_N(n)\}$ *for every nominal* $o \in \mathcal{O}$.

We say that a node n of a graph G satisfies a formula ϕ, *written* $n \models \phi$ *if* $n \in \phi^{\mathcal{G}}$. *We say that a graph G is a model of formula* ϕ *(or satisfies* ϕ*), written* $G \models \phi$ *if* $\phi^{\mathcal{G}} = N$ *that is every node of G belongs to the interpretation of* ϕ *induced by G.*

Many different ontologies use smaller fragments of $\mathcal{ALC}$ because many inference problems turn out to be untractable for more expressive logics. The Web Ontology Language OWL [28], the standard language for publishing and sharing ontologies on the World Wide Web, defines several variants (OWL DL, OWL EL, OWL QL and OWL RL) that are compatible with Description Logics of various expressive powers, and thus of various complexity.

It is also worth mentionning that there are two ways to consider an ontology in relation to the data. In many actual uses of ontologies, the size of the instantiated data is a problem and one of the goals is to keep it as small as possible without loosing any information. Ontologies are then used to generate the complete knowledge from what is actually stored in the data. This can be crucial as some of these logics lack the finite-model property. The other approach is to consider that the ontology forms a set of conditions that have to be met by the data, i.e., to consider that the data is the complete model of the accessible knowledge. In the following, we define a pre- and postcondition, conditions that have to be true in the graph and not the ones that cannot be proven to be false. At this point, we do not worry about the data and instead focus on the theoretical model which, do not need to be instantiated and can thus be infinite. Even though Description Logics are the de facto standard to express ontologies, they may not be the best tool to describe some properties of the data or some of the changes that one needs to make. We present a dynamic logic, introduced in [29], that can be used to represent properties that focus more on the edges and the paths in the graph.

Definition 12 Given three countably infinite and pairwise disjoint alphabets Σ, *the set of* names, Φ_0, *the set of* atomic propositions, Π_0, *the set of* atomic programs, *the language of* $\mathcal{C2PDL}$ *is composed of* formulas *and* programs.[3] *We partition the set of names* Σ *into two countably infinite alphabets* Σ_1 *and* Σ_2 *such that* $\Sigma_1 \cup \Sigma_2 = \Sigma$ *and* $\Sigma_1 \cap \Sigma_2 = \emptyset$. *Formulas* ϕ *and programs* α *are defined as:*

$$\phi := \top \mid \{o\} \mid \phi_0 \mid \neg\phi \mid \phi \vee \phi \mid \langle\alpha\rangle\phi$$
$$\alpha := \alpha_0 \mid \nu_S \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \alpha^- \mid \phi?$$

where $i \in \Sigma$, $\phi_0 \in \Phi_0$, $\alpha_0 \in \Pi_0$ *and* $S \subseteq \Sigma$.

We denote by Π *the set of programs and by* Φ *the set of formulas. As ususal,* $\phi \wedge \psi$ *stands for* $\neg(\neg\phi \vee \neg\psi)$ *and* $[\alpha]\phi$ *stands for* $\neg(\langle\alpha\rangle\neg\phi)$.

Definition 13 Given a signature (Φ_0, Π_0, Σ), *where* Φ_0 *is the set of atomic formulae,* Π_0 *is the set of atomic programs and* Σ *the set of nominals and partition it into* Σ_E *and* Σ_O, *an interpretation is a tuple* $(\Delta, \cdot^{\mathcal{I}}, \chi)$ *where* $N \subseteq \Delta$, $\cdot^{\mathcal{I}}$ *is a function that takes a formula* ϕ *(resp. a program* π*) and such that* $\forall\phi_0 \in \Phi_0.\phi_0^{\mathcal{I}} \subseteq \Delta$ *(resp.* $\forall\pi_0 \in \Pi_0.\pi_0^{\mathcal{I}} \subseteq \Delta \times \Delta$*) and* χ *is the naming function, from* Σ *to* Δ. χ *is such that* $\chi(\Sigma_E) = N$ *and* $\chi(\Sigma_O) = \Delta \backslash N$. *We require that* χ *and* Φ_N *agree on* Σ_E. *In the following,* $\phi_0 \in \Phi_0$, $\pi_0 \in \Pi_0$, $o \in \Sigma$, $S \subseteq \Sigma$:

- $\top^{\mathcal{I}} = \Delta$
- $\{o\}^{\mathcal{I}} = \chi(o)$
- $(\neg\phi)^{\mathcal{I}} = \Delta \backslash \phi^{\mathcal{I}}$
- $(\phi \vee \psi)^{\mathcal{I}} = \phi^{\mathcal{I}} \cup \psi^{\mathcal{I}}$
- $([\pi]\phi)^{\mathcal{I}} = \{x \in \Delta | \forall y \in \Delta.(x, y) \in \pi^{\mathcal{I}} \Rightarrow y \in \phi^{\mathcal{I}}\}$
- $(\pi \cup \alpha)^{\mathcal{I}} = \pi^{\mathcal{I}} \cup \alpha^{\mathcal{I}}$
- $(\pi;\alpha)^{\mathcal{I}} = \{(x, y) \in \Delta \times \Delta | \exists z \in \Delta.(x, z) \in \pi^{\mathcal{I}}$ *and* $(z, y) \in \alpha^{\mathcal{I}}\}$
- $(\pi^-)^{\mathcal{I}} = \{(x, y) \in \Delta \times \Delta | (y, x) \in \pi^{\mathcal{I}}\}$
- $(\pi^*)^{\mathcal{I}} = \{(x, y) \in \Delta \times \Delta | x = y$ *or* $\exists z \in \Delta.((x, z) \in \pi^{\mathcal{I}}$ *and* $(z, y) \in (\pi^*)^{\mathcal{I}})\}$
- $(\nu_S)^{\mathcal{I}} = \{(x, y) \in \Delta \times \Delta | x \in \chi(S)$ *and* $y \in \chi(S)\}$
- $(\phi?)^{\mathcal{I}} = \{(x, y) \in \Delta \times \Delta | x = y$ *and* $x \in \phi^{\mathcal{I}}\}$

We write ν *instead of* ν_Σ *in order to shorten formulae. As usual,* $\langle\pi\rangle\phi$ *is shorthand for* $\neg([\pi]\neg\phi)$ *and* $\phi \wedge \psi$ *is shorthand for* $\neg(\neg\phi \vee \neg\psi)$.

The notion of interpretation induced by a decorated graph is obtained in the same way as it was in the case of description logics.

None of the logics we used before contain existentially or universally quantified variable. This will become important in the next section. We introduce the first-order logic that enables us to define existential and universal variables.

[3]This notion of *programs* is borrowed from Propositional Dynamic Logic.

$$
\begin{array}{ll}
wp(a,\ Q) = Q[a] & wp(a;\alpha,\ Q) = wp(a, wp(\alpha, Q)) \\
wp(\epsilon,\ Q) = Q & wp(s_0; s_1,\ Q) = wp(s_0, wp(s_1,\ Q)) \\
wp(s_0 \oplus s_1,\ Q) = wp(s_0, Q) \wedge wp(s_1, Q) & wp(s^*,\ Q) = inv_s \\
wp(\rho,\ Q) = App(\rho) \Rightarrow wp(\alpha_\rho, Q) & wp(\rho!, Q) = App(\rho) \wedge wp(\alpha_\rho, Q) \\
wp(\rho?, Q) = (App(\rho) \Rightarrow wp(\alpha_\rho, Q)) \wedge (\neg App(\rho) \Rightarrow Q) &
\end{array}
$$

FIGURE 4. **Weakest preconditions w.r.t. actions and strategies, where a (resp. α, α_ρ) stands for an elementary action (resp. action, the right-hand side of a rule ρ) and Q is a formula.**

$$
\begin{array}{lcl}
vc(\epsilon,\ Q) & = & vc(\rho,\ Q) = vc(\rho!, Q) = vc(\rho?, Q) = \top \text{ (true)} \\
vc(s_0; s_1,\ Q) & = & vc(s_0, wp(s_1,\ Q)) \wedge vc(s_1, Q) \\
vc(s_0 \oplus s_1,\ Q) & = & vc(s_0, Q) \wedge vc(s_1, Q) \\
vc(s^*,\ Q) & = & vc(s, Q) \wedge (inv_s \wedge App(s) \Rightarrow wp(s, inv_s)) \wedge (inv_s \wedge \neg App(s) \Rightarrow Q)
\end{array}
$$

FIGURE 5. **Verification conditions for strategies.**

Definition 14 (First-Order Formula): Let $\mathcal{A} = (\mathcal{V}, \mathcal{C}, \mathcal{R})$ where $\mathcal{V}$ is a set of variables, $\mathcal{C}$ is a set of unary predicates, and $\mathcal{R}$ is a set of binary predicates including equality (=). Given $x, y \in \mathcal{V}$, $C \in \mathcal{C}$ and $R \in \mathcal{R}$, the set of first-order formulas ϕ we consider is defined by:
$\phi := \top \mid C(x) \mid R(x, y) \mid x = y \mid \neg\phi \mid \phi \vee \phi \mid \exists x.\phi$

For the sake of conciseness, we define $\bot \equiv \neg\top$, $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$, $\forall x.\phi \equiv \neg(\exists x.\neg\phi)$.

A variable x is free in ϕ iff $\phi = C(t_0)$, $\phi = R(t_0, t_1)$ or $\phi =$ "$t_0 = t_1$" and x occurs in t_0 or t_1, or $\phi = \neg\psi$ or $\phi = \psi \vee \psi'$ and x is free in ψ and ψ', or $\phi = \exists y.\psi$ and x is free in ψ and x is different from y. A formula with no free variable is a sentence. We only consider sentences hereafter.

Definition 15 (Model): Let $G = (N, E, \Phi_N, \Phi_E, s, t)$ be a graph over the alphabet $(\mathcal{C}, \mathcal{R})$, an interpretation over the alphabet $(\mathcal{V}, \mathcal{C}, \mathcal{R})$ is a tuple $(\Delta, \cdot^{\mathcal{I}})$ such that $N \subseteq \Delta$ and $\cdot^{\mathcal{I}}$ is a function over formulas defined by:

- *$\top^{\mathcal{I}}$ is true*
- *$C(x)^{\mathcal{I}}$ is true if and only if $C \in \Phi_N(x)$*
- *$R(x, y)^{\mathcal{I}}$ is true if and only if $\exists e \in E.s(e) = x$ and $t(e) = y$ and $R \in \Phi_E(e)$*
- *$x =^{\mathcal{I}} y$ is true if and only if x is y*
- *$(\exists x.\phi)^{\mathcal{I}}$ is true if and only if $\exists n \in N.\phi[x \to n]^{\mathcal{I}}$ where $\phi[x \to n]$ is ϕ where each occurrence of x is replaced with n*
- *$(\neg\phi)^{\mathcal{I}}$ is true if and only if not $\phi^{\mathcal{I}}$*
- *$(\phi \vee \psi)^{\mathcal{I}}$ is true if and only if $\phi^{\mathcal{I}}$ or $\psi^{\mathcal{I}}$*

We say that a graph G models a first-order formula ϕ, written $G \models \phi$ if there exists an interpretation $(\Delta, \cdot^{\mathcal{I}})$ such that $\phi^{\mathcal{I}}$ is true.

Satisfiability of formulae in first-order logic is known to be undecidable. We thus advise to use decidable fragments of first-order logic such as $\mathcal{C}2$ [30], the two variable fragment of first-order logic with counting, i.e., only two variables (for instance, x and y) can be ever be used.

IV. VERIFICATION

Previously, we have shown how to use logics to label edges and nodes of graphs. We now go a little further and show how we can use logics to define specifications for the transformations we want to perform, i.e., how to define conditions that we want to be satisfied by the graph after the transformation is performed, given that it may have satisfied another (possibly identical) set of conditions initially.

Definition 16 (Specification): A specification *SP is a triple $\{Pre\}(\mathcal{R}, s)\{Post\}$ where Pre and Post are formulas (of a given logic), $\mathcal{R}$ is a graph rewriting system and s is a strategy.*

Example 17 If we reuse the strategy $s_0 \equiv \rho_0!; \rho_0?; (\rho_1 \oplus \rho_0')^*$ previously defined in Example II, a specification could be $\{C \subseteq 8A\}(\mathcal{R}_0, s_0)\{\neg B(o)\}$.

Definition 18 (Correctness): A specification SP is said to be correct *iff for all graphs G, G' such that $G \Rightarrow_s G'$ and $G \models Pre$, then $G' \models Post$.*

In order to prove the correctness of a specification, we use a Hoare-like approach [31]. The idea is that it is possible to split the transformation into elementary changes that impact the graph in a known and controlled way. In such a situation, given the postcondition that needs to be achieved, it becomes possible to generate the weakest precondition that ensures that the postcondition will be satisfied. This can then be iterated to generate the weakest precondition for the whole transformation.

This process is achieved by two functions: the weakest-precondition $wp(s, Q)$ and the verification condition $vc(s, Q)$ for a strategy s and a postcondition Q. More details can be found in [32]. The definitions of these functions are given in Fig. 4 and Fig. 5 respectively.

The weakest preconditions and verification conditions introduce new logic constructors to deal with elementary actions called substitutions and written $Q[a]$ where Q is a logic formula and a is an action. Intuitively, a graph G is a model of the formula $Q[a]$ if and only if $G[a]$, the graph obtained by performing action a on G, is a model of ϕ.

Definition 19 (Substitutions): To each elementary action a is associated a substitution*, written $[a]$, such that for all graphs G and DL formula ϕ, $(G \models \phi[a]) \Leftrightarrow (G[a] \models \phi)$.*

It is worth noting that the weakest precondition of a closure, s^*, is inv_s, an invariant for that closure. This invariant is not part of the original specification but needs to be specified. We thus modify the notion of specification.

Definition 20 (Annotated Specification): An annotated specification SP is a triple $\{Pre\}(\mathcal{R}, s)\{Post\}$ where Pre and Post are formulas (of a given logic), $\mathcal{R}$ is a graph rewriting system, s is a strategy and every closure in s is annotated with an invariant.

Example 21 As the strategy $s_0 \equiv \rho_0!; \rho_0?; (\rho_1 \oplus \rho_0')^*$ previously defined contains a closure, we annotate it. This yields, for instance, $s_1 \equiv \rho_0!; \rho_0?; (\rho_1 \oplus \rho_0')^*\{C \subseteq \exists agents\_in.Cp\}$ and the associated annotated specification is $\{C \subseteq 8A\}(\mathcal{R}_0, s_1)\{\neg B(o)\}$.

Now that the notions of the weakest precondition and the verification condition are defined, we can look back at the original problem we were trying to solve. We define a formula that represents the correctness of a specification.

Definition 22 (Correctness Formula): We call correctness formula *of an annotated specification $SP = \{Pre\}(\mathcal{R}, s)$ $\{Post\}$, the formula:*

$$correct(SP) = (Pre \Rightarrow wp(s, Post)) \wedge vc(s, Post).$$

Theorem 23 (Soundness): Let $SP = \{Pre\}(\mathcal{R}, s)\{Post\}$ be an annotated specification. If $correct(SP)$ is valid, then for all graphs G, G' such that $G \Rightarrow_s G'$, $G \models Pre$ implies $G' \models Post$.

Deciding whether a specification is correct can be translated into deciding the validity of a given formula. This is one of the main reasons why we focused on decidable logics in Section III. Another possible choice is to only consider tractable logic so that verification becomes achievable in a reasonable timeframe.

The decidability of the validity problem for the logic used to label the graph is not, however, the only condition for the decidability of the correctness problem. The definitions of the weakest preconditions introduced substitutions as a new formula constructor. In order for the correctness problem to be decidable, these new constructs must be expressible in the logic, i.e., the logic must be closed under substitutions. We repeat here some relevant theorems from previous papers [25], [29], [32] dealing with substitutions.

Theorem 24 [25]: The Description Logics $\mathcal{ALCUO}$, $\mathcal{ALCUOI}$, $\mathcal{ALCUO}Self$, $\mathcal{ALCUOIQ}$, $\mathcal{ALCUOI}Self$ and $\mathcal{ALCUOIQ}Self$ are closed under substitutions.

Theorem 25 [25]: The Description Logics $\mathcal{ALC}$, $\mathcal{ALCU}$, $\mathcal{ALCO}$, $\mathcal{ALCI}$, $\mathcal{ALCQ}$, $\mathcal{ALC}Self$, $\mathcal{ALCUI}$, $\mathcal{ALCUQ}$, $\mathcal{ALCU}Self$, $\mathcal{ALCOI}$, $\mathcal{ALCOQ}$, $\mathcal{ALCO}Self$, $\mathcal{ALCIQ}$, $\mathcal{ALCI}Self$, $\mathcal{ALCQ}Self$, $\mathcal{ALCUOQ}$, $\mathcal{ALCUIQ}$, $\mathcal{ALCUI}Self$, $\mathcal{ALCUQ}Self$, $\mathcal{ALCOIQ}$, $\mathcal{ALCOI}Self$, $\mathcal{ALCOQ}Self$, $\mathcal{ALCIQ}Self$, $\mathcal{ALCUOQ}Self$, $\mathcal{ALCUIQ}Self$ and $\mathcal{ALCOIQ}Self$ are not closed under substitutions.

Theorem 26 [29]: $\mathcal{C2PDL}$ is closed under substitutions.

Theorem 27 [32]: $\mathcal{C2}$ is closed under substitutions.

For all the logics that are closed under substitution, the proof consists in a set of rewrite rules that conserve the interpretation. For instance, given C_0 an atomic concept, σ a substitution, i and j individuals and π_0 a program:

- $\top\sigma \rightsquigarrow \top$
- $C_0[add_C(C_0, i)] \rightsquigarrow C_0 \vee \{i\}$
- $(\exists r_0.C)[del_E(i, j, r_0)] \rightsquigarrow (\neg\{i\} \wedge \exists r_0.(C[del_E(i, j, r_0)])) \vee (\exists r_0.(\neg\{j\} \wedge C[del_E(i, j, r_0)]))$

Another possible problem is that the logic needs to be able to express the existence (and absence) of a match. First-order logic can express $App(\rho)$ by using an existential variable for every node of the left-hand side of the rule ρ. This is not possible in the other types of logics we considered as they do not allow to define an unlimited number of variables. There is thus a limitation on what can appear on the left-hand side of the rules.

It is possible to express that a match exists at specific nodes by introducing new nominals as existential variables. If the correctness formula is valid, the formula is to be correct no matter which nodes of the graph are labeled with the additional nominals. However, the resulting formula is not equivalent to the existence of a match and, in particular, its negation is not equivalent to the non-existence of a match. It is possible to express the non-existence of a match when some conditions on the rules are satisfied (for instance, if the left-hand side is acyclic) but we do not go into details in this paper.

V. APPLICATION TO MALARIA SURVEILLANCE

Now that we have formally introduced the verification of graph transformations, we show how this framework can be used in the context of malaria surveillance and prevention, where dealing with dynamic data sources and maintaining the integrity of integrated data sources are utmost importance.

We present a few examples of graph transformations that can take place in the context of malaria surveillance and demonstrate how our method for verification can be used to verify their correctness.

A. VERIFICATION OF DATA EVOLUTION

Let us assume that we are maintaining a database, i.e., a graph, that contains data about the drugs used to treat malaria. We assume that this database is aligned and consistent with the IDOMAL ontology [7]. A new drug n is added to the database and an old one o is removed. We know that n is a version of *Chloroquine* and o is a version of *bulaquine*. Such a change is, for instance, the result of the SQL code in Fig. 6.

For the sake of simplicity, we only report here parts of the IDOMAL ontology that are relevant. The axioms that we use are $\psi_0 \equiv Bulaquine \subseteq 8 - aminoquinoline$, $\psi_1 \equiv Chloroquine \subseteq 4 - aminoquinoline$, $\psi_2 \equiv Chloroquine \subseteq \exists agent\_in.Chemoprolaxis$, $\psi_3 \equiv Chloroquine \subseteq$

```
INSERT INTO Drugs (DrugName, Type)
VALUES (n, Chloroquine);

DELETE FROM  Drugs
WHERE DrugName = 'o' AND Type = 'Bulaquine';
```

FIGURE 6. Example of SQL codes that modify a database.

$\exists part\_of.ArtesunateAndChloroquineCombination$ and $\psi_4 \equiv Reflexive(has\_part)$. In this example, we assume that the ontology represents both the initial state of the data, i.e., we assume that the data is consistent with the ontology, and the expected final state of the data, i.e., we want the data to still be consistent with the ontology after the change. We add to the precondition that there exists a drug named "o" of type "Bulaquine".

We use the rule system in Fig. 2 to represent the change. The corresponding strategy is $\rho_0!; \rho_1!$, i.e., apply ρ_0 and then ρ_1 exactly once.

Let us rewrite the precondition in $\mathcal{ALCUOS}elf$. For the sake of conciseness and to make the formulas more readable, we will abbreviate *Bulaquine* as B, *Chloroquine* as C, $8-aminoquinoline$ as $8A$, $4-aminoquinoline$ as $4A$, *Chemoprolaxis* as Cp and *ArtesunateAndChloroquine Combination* as $A\&C$. The axioms become $\psi_0 \equiv \forall U.(B \Rightarrow 8A)$, $\psi_1 \equiv \forall U.(C \Rightarrow 4A)$, $\psi_2 \equiv \forall U.(C \Rightarrow \exists agent\_in.Cp)$, $\psi_3 \equiv \forall U.(C \Rightarrow \exists part\_of.A\&C)$ and $\psi_4 \equiv \forall U.(\exists has\_part.Self)$. We now define the specification. The postcondition is $Post \equiv \psi_0 \wedge \psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4$ and the precondition is $Pre \equiv Post \wedge \exists U.(\{o\} \wedge B)$.

It is now possible to generate the correctness formula for the specification $\{Pre\}(\mathcal{R}_0, \rho_0!; \rho_1!)\{Post\}$. In order to save space, we do not show the algorithm and the proof of its correctness that lead to the exact formula but interested readers can find it in [25].

The result is $Pre \Rightarrow (\exists U.(\{o\} \wedge B) \wedge (\exists U.(\{i\} \wedge \{o\} \wedge B) \wedge \phi_0 \wedge \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4))$ where:

- $\phi_0 \equiv \forall U.((B \wedge \neg\{i\}) \Rightarrow (8A \wedge \neg\{i\}))$,
- $\phi_1 \equiv \forall U.(((C \vee \{n\}) \wedge \neg\{i\}) \Rightarrow (4A \wedge \neg\{i\}))$,
- $\phi_2 \equiv \forall U.(((C \vee \{n\}) \wedge \neg\{i\}) \Rightarrow (\neg\{i\} \wedge \exists.agent\_in.(Cp \wedge \neg\{i\})))$,
- $\phi_3 \equiv \forall U.(((C \vee \{n\}) \wedge \neg\{i\}) \Rightarrow (\neg\{i\} \wedge \exists.part\_of.(A\&C \wedge \neg\{i\})))$ and
- $\phi_4 \equiv \forall U.(\neg\{i\} \Rightarrow \exists has\_part.Self)$.

Let us now check if the specification is correct. The evalution of the ϕ_i's is affected by the labeling of the node n, the node i and all the other nodes globally. All the implications of the ψ_i are trivially true at i because the left side of the inclusion is false. It is also straighforward that for every node that is neither i nor n, $\psi_i \Rightarrow \phi_i$. This is not true, however, for n as there is no reason for $4A$ to be a label of n. This specification is thus incorrect.

In order to solve this problem, we modify the rule ρ_0. We obtain the rule ρ'_0 of Fig. 2. The left-hand side is modified to make possible the creation of the missing edges by adding their targets, a and c. The right-hand side of the rule now adds that n is an $8-aminoquinoline$ and creates edges leading to a and c respectively labeled with $part\_of$ and $agents\_in$. The resulting correctness formula is:
$Pre \Rightarrow \exists U.((\{a\} \wedge A\&C) \wedge \exists U.(\{c\} \wedge Cp)) \wedge (\exists U.(\{o\} \wedge B) \wedge (\exists U.(\{i\} \wedge \{o\} \wedge B) \wedge \chi_0 \wedge \chi_1 \wedge \chi_2 \wedge \chi_3 \wedge \chi_4))$ where:

- $\chi_0 \equiv \phi_0$,
- $\chi_1 \equiv \forall U.(((B \vee \{n\}) \wedge \neg\{i\}) \Rightarrow ((4A \vee \{n\}) \wedge \neg\{i\}))$,
- $\chi_2 \equiv \forall U.(((C \vee \{n\}) \wedge \neg\{i\}) \Rightarrow (\neg\{i\} \wedge (\{n\} \vee \exists.agent\_in.(Cp \wedge \neg\{i\}))))$,
- $\chi_3 \equiv \forall U.(((C \vee \{n\}) \wedge \neg\{i\}) \Rightarrow (\neg\{i\} \wedge (\{n\} \vee \exists.part\_of.(A\&C \wedge \neg\{i\}))))$ and
- $\chi_4 \equiv \forall U.(\neg\{i\} \Rightarrow (\{n\} \vee \exists has\_part.Self))$.

The problem of n is now solved as it appears on both sides of the implications.[4]

As proved in [25] the least expressive logics are not suitable for verification. It is, however, possible to express the less expressive specifications in more expressive logics. This has a cost in terms of complexity of verification. The complexity of verification itself depends on the size of the specification, and not the size of data, thus it is usually much smaller.

The grammar used for OBO [33] is more expressive than the one used in [25]. For instance, the OBO format also allows the use of some axioms like $Transitive(R)$ or $R \subseteq R'$ that cannot be easily expressed in those logics. It is, however, possible to split the specifications into multiple parts and use a combination of several types of logics depending on the needs and requirements.

We use the same rules and strategy as before but the precondition and the postcondition are now *Antisymmetric* (has_part). This can be translated to $\mathcal{C}2$[5] [30]: $\forall x, y.part\_of(x, y) \Rightarrow \neg part\_of(y, x)$. As $\mathcal{C}2$ is known to be closed under substitution and to be able to express the existence of a match for the rules we used, it is possible to prove that this specification is correct.

[4]The fact that n and i are different is important here. The actual correctness formula checks that as well but it would require us to introduce details that do not add much to the narrative of this paper.

[5]$\mathcal{C}2$ strictly contains the logics in [25] but the complexity of its satisfiability problem is harder than the ones for some of the least expressive logics.

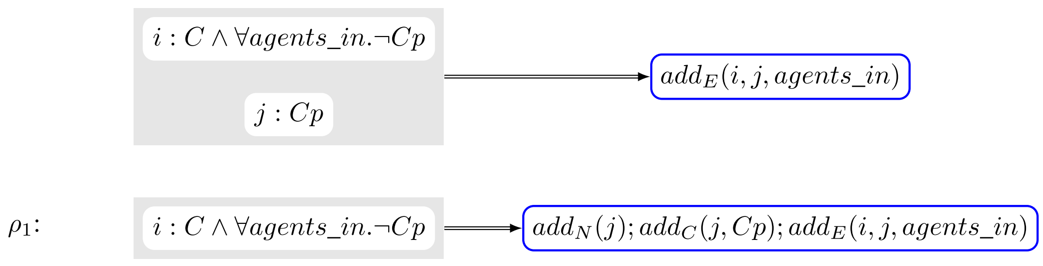

$i : C \wedge \forall agents\_in.\neg Cp$
$j : Cp$
$add_E(i, j, agents\_in)$
ρ_1:
$i : C \wedge \forall agents\_in.\neg Cp$
$add_N(j); add_C(j, Cp); add_E(i, j, agents\_in)$

FIGURE 7. Example of rules used in the knowledge base model example. They form the rewrite rule system $\mathcal{R}_{kb}$.

The correctness formula that we obtain is $Pre \Rightarrow \exists x, y.(c(x) \wedge Cp(x) \wedge a(y) \wedge A\&C(y)) \wedge \exists x.(o(x) \wedge B(x)) \wedge \forall x, y.(part\_of(x, y) \vee (n(x) \wedge a(y)) \Rightarrow \neg part\_of(y, x) \wedge (\neg n(y) \vee \neg a(x)))$. To prove that this formula is correct, we distinguish between the case where $(x, y) = (n, a)$ and the case where it is not so. As n was created by the transformation, it has no incoming or outgoing edge labeled with *part_of* except for the one to a. The former case is thus correct. The latter case is implied by the precondition and the fact that y cannot be n as n has no incoming edge.

It is also likely that data is modified in order to have an impact on what it represents. This means that either the ontology or some specific relations between elements of the data will be modified to reach the goal. In such a situation, one can use a logic completely different from the ones used in the ontology.

Let us assume that the goal of the introduction of the new drug n is to have a drug that is *part_of* something that is an *agent_in ChemotherapyOfMalaria* and *part_of* something, possibly a different one, that is an *agent_in MalariaPreventionInIndividuals*. The ontology says that *part_of* and *agent_in* are transitive but this is usually not reflected in the data. To simulate this, we use the transitive closure from $\mathcal{C2PDL}$ [29] after the transformation $\langle \nu \rangle (\langle part\_of^*; agent\_in^* \rangle ChemotherapyOfMalaria) \wedge (\langle part\_of^*; agent\_in^* \rangle MalariaPreventionInIndividual)$. A correct specification can then be found if the data is such that, i.e., satisfies a precondition containing, for instance, $[\nu](A\&C \Rightarrow \langle part\_of^*; agent\_in^* \rangle ChemotherapyOfMalaria) \wedge [\nu](Cp \Rightarrow \langle agent\_in^* \rangle MalariaPreventionInIndividual)$.

Splitting a verification problem into smaller problems using different logics is far from trivial. One needs to be able to express the labels occuring in the right hand-side of the rules in all of the logics and determining the part of the precondition and postcondition that form the right specification. This, however, allows the user to have access to more expressive power when defining the pre- and postconditions.

B. GRAPH TRANSFORMATION FOR ONTOLOGY REALIZATION

As mentioned in Section III there were two ways to view ontologies: as constraints that the data has to follow or as rules to obtain the knowledge from a model containing all the information needed to infer the data. In the application of verifications, we tend to use the first view more. It is, however, possible to realize the second view via graph transformations.

Let us assume that the ontology we are working with contains the axiom $C \subseteq \exists agents\_in.Cp$. One can use the rules ρ_0 and ρ_1 of Fig. 7 to make the axiom true. The rule ρ_0 and ρ_1 both search for a C that does not have an outgoing edge labeled with *agents_in* leading to a Cp. This means that they look for an element that does not satisfy the axiom. ρ_0 connects that element to an existing Cp while ρ_1 creates a new one. A correct specification would then be $\{\top\}(\mathcal{R}_{kb}, (\rho_0 \oplus \rho_1)^*\{\top\})\{C \subseteq \exists agents\_in.Cp\}$. This specification is correct because the correctness formula is, when translated to first-order logic, $\forall x.(\neg C(x) \vee \exists y.Cp(y) \wedge agents\_in(x, y)) \Rightarrow \forall x.(C(x) \Rightarrow \exists y.Cp(y) \wedge agents\_in(x, y))$. That formula is trivially valid because both sides of the implication are the same formula. The left-hand side is the negation of the application condition of ρ_0 and ρ_1 and the right-hand side is the postcondition.

Assuming there are rules for every axiom of the ontology, it is possible to generate all its instances. This method can then be used to reason about every possible model of the knowledge. The rules become much more complex when dealing with more than one axiom. In such a case, one needs to make sure that the axioms interact in a meaningful way.

C. VERIFICATION OF ONTOLOGY UPDATE

Data is not the only part that can change or that can be modeled as a graph. Ontologies represented in OBO format are already modeled as graphs with every concept represented as a node.

The five axioms of IDOMAL we used previously can be expressed as $is\_a(B, 8A)$, $is\_a(C, 4A)$, $agents\_in(C, Cp)$, $part\_of(C, A\&C)$ and $Reflexive(has\_part)$.

The ontologies evolve as new axioms, concepts and relationships are added, removed, or renamed, etc. These changes and transformations can be expressed as graph transformations, similar to what has been done for data.

The ontologies VSMO [5] and NCBITaxon [34] both define the concept Anopheles, a genus of mosquito known as a primary vector for transmitting malaria. Both describe the same entities but they are not always interchangeable. If an ontology imports these two ontologies, it might end up with some Anopheles according to the VSMO definition and some according to the NCBITaxon. A solution would be to search for every instance of only one of the Anopheles

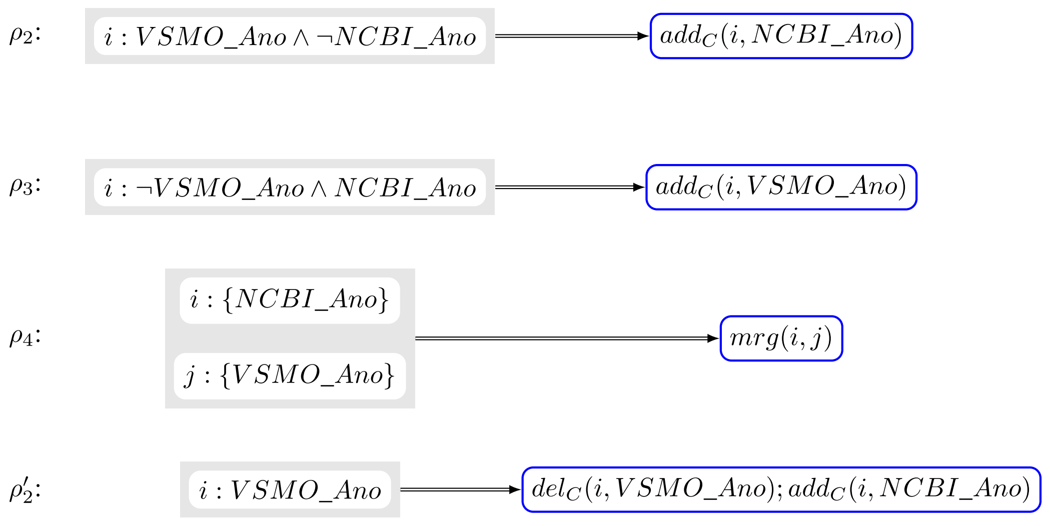


FIGURE 8. Example of rules used for the ontology update. They form the rewrite rule system $\mathcal{R}_{ou}$.

definitions and add the missing one. The rules doing that action, ρ_2 and ρ_3, are shown in Fig. 8. The strategy would then be $s_{ou} \equiv (\rho_2 \oplus \rho_3)^*$. In order to use it in an annotated specification, we annotate it with the tautological invariant: $s_{ou} \equiv (\rho_2 \oplus \rho_3)^*\{\top\}$.

Let us now look at the specification $\{\top\}(\mathcal{R}_{ou}, s_{ou})\{VSMO\_Ano \equiv NCBI\_Ano\}$. It says that after the transformation the two definitions of Anopheles will be interpreted the same way. This can easily be proved to be correct.

However, in the final result, there are still two different names for the same concept even if they agree on every interpretation. This makes things error-prone as, henceforth, every time any of the two concepts is modified the other has to be modified as well. Another solution would be considering the ontologies as the data and modifying them. To do so we use rule ρ_4 of $\mathcal{R}_{ou}$ that merges the two concepts. One can then prove that the specification $\{\top\}(\mathcal{R}_{ou}, \rho_4; \rho_2'^*\{\top\})\{\forall U.\neg\{VSMO\_Ano\} \wedge VSMO\_Ano \subseteq \bot\}$ is correct. The result of applying this strategy is shown in Fig. 9. The graph that is modified in that case contains both the ontologies and the data they represent. It is possible to split that into two different specifications, for instance $\{\top\}(\mathcal{R}_{ou}, \rho_4)\{\forall U.\neg\{VSMO\_Ano\}\}$ dealing with the ontology and $\{\top\}(\mathcal{R}_{ou}, \rho_2'^*\{\top\})\{VSMO\_Ano \subseteq \bot\}$ dealing with the data. Another possibility is to replace the specification dealing with the data with another one where the initial state of the ontology, say $\mathcal{O}$ as the precondition and the postcondition is the result we are looking for, say $\mathcal{O} \cup \{VSMO\_Ano \subseteq \bot\}$.

D. VERIFICATION OF ONTOLOGY INTERACTIONS

It is also possible to consider ontologies as nodes of the graphs. In this way, the edges of graphs represent how the ontologies interact with each other. It would, for instance, keep track of which ontologies import any given ontology or which ontologies are in conflict with each other.

Importing ontology o_0 in the ontology o_1 would then result in a specification of the sort $\{\forall x, y.imports(x, y) \Rightarrow \neg conflict(x, y)\}(\mathcal{R}, \rho)\{\forall x, y.imports(x, y) \Rightarrow \neg conflict(x, y)\}$ where ρ is a rule that simply executes the elementary action $add_E(o_0, o_1, imports)$.

Once again, the various levels of graph transformation might work together. Let us again consider the transformation shown in the previous subsection where the definitions of Anopheles in the VSMO ontology and the NCBI ontology are merged. That means these two ontologies are modified, inducing a new transformation at the ontology level: $add_C(VSMO, Modified)$; $add_C(NCBI, Modified)$. This could prevent the import of those ontologies until it has been proven that changes do not harm the consistency of the underlying knowledge bases. If all the knowledge bases are still consistent, the *Modified* tags can be removed. Otherwise, some alterations to the ontology or data may be induced by the changes, possibly inducing new changes in other dependent ontologies in the process, until a new stable state is reached. All of these changes can be modeled using graph transformations and proven to be correct, or not, using verification.

E. LANGUAGE INTEROPERABILITY

Malaria affects many people living in different countries or regions of the world and using a variety of different languages to communicate. Therefore, maintaining language interoperability is crucial in malaria surveillance systems with data sources scattered in various geographical locations and settings. There are many ways to look at language interoperability. One way is to consider that there exists a set of uniform and homogeneous ontologies, supposedly written in English, that describe malaria surveillance. However, some pieces of data coming from other countries may use a different language. To create an interoperable knowledge base, it is required to express the non-English data into the English ontologies (and vice-versa).

For instance, we plan on using IDOMAL with another piece of data expressed in French and containing the information that d is a mosquito on which a test t has been done 2 hours after a blood meal. This corresponds to the data

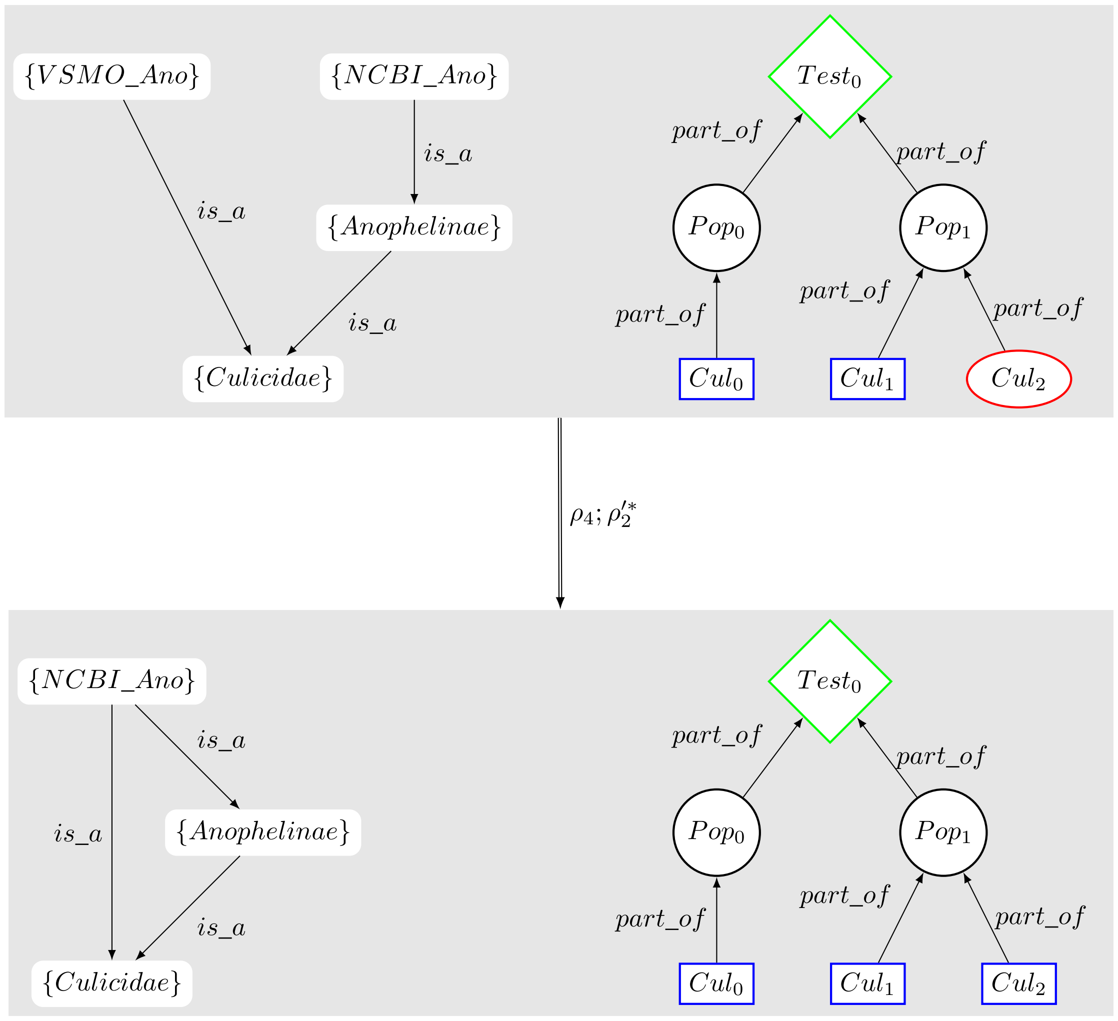


FIGURE 9. Example of application of the strategy $\rho_4; \rho_2'^*$. Blue rectangles are nodes labeled with *NCBI_Ano*, red ellipses are labeled with *VSMO_Ano*, black circles are labeled with *MosquitoPopulation* and green diamonds with *Bioassay*.

Moustique(d); *TestSurMoustique*(t); 2*HeuresApresRepas* (i); *SujetTest*(t, d); *InstantTest*(t, i). In order to express it into IDOMAL, one has to look at the ontology and find similar relations and concepts. For instance, IDOMAL contains the concepts *Culicidae*, *InvestigationRelatingToTheInsectVector* and 2*Hours AfterBloodMeal* that are near perfect translations of the concepts in French. It is thus possible to apply a transformation similar to the one that was done in the case *NCBI_Ano* and *VSMO_Ano* to modify the data so that it is now accessible in both languages.

The issue is that such a task has to be executed for every data source, ontology pair by someone that is proficient in both languages to create the rules instantiating the data in both languages. A more efficient way to deal with the problem, notwithstanding the problems of maintaining both versions at the same time, would be to create equivalent ontologies in both languages so that data can be linked to the ontology in the target language. The exact same transformations would then be used but they can be generated automatically instead of being created by hand.

Another way to approach the problem is to assume that ontologies already exist in different languages and that we want to make them cross-compatible. This amounts to finding equivalent concepts in both ontologies, e.g. *Moustique* and *Culicidae* and adding an equivalency edge between those concepts: add_E(*Moustique*, *Culicidae*, *equivalent*). As mentioned previously, such a change would trigger some other changes to both ontologies (it makes sense that *equivalent* be an equivalence relation and thus symmetric, transitive and reflexive) and in the data that they model.

VI. DISCUSSION AND CONCLUSION

In this paper, we introduced the notions of logically decorated graphs, graph transformation and verification for change management in malaria surveillance data sources and systems.

The goal is to use these theoretical tools to model, represent, integrate and manage the dynamic knowledge necessary for efficient disease surveillance and response at local, national and global scales. This approach has been successfully applied through multiple experiments within the Semantics, Interoperability, and Evolution for Malaria Analytics (SIEMA) platform [35]–[37]. The proposed framework is domain independent and can be generalized to other domains, where maintaining interoperability of dynamic evolving and temporal data sets is the integral focus. Our method isolates the parts that remain unchanged and therefore enables concurrent changes within an integrated system with minimum interruption to the system's operation for a certain time interval. This results in improved interoperability between different data sources and health systems, minimization of errors and uncertainties in malaria analytics, better utilization of limited health-care resources and improved timeliness of data collection to analysis and decision-making to reduce morbidity and mortality from Malaria.

Our use of graphs and graph transformations yields three very important results. First, they offer a standard and simple way to represent the data and, more importantly, the way it evolves. Second, the proposed framework is independent of any particular domain, algorithm, protocol, or implementation language. This is critical for improving reusability and portability of our solution for several other application domains. Third, another major contribution of our work is in terms of correction and automation. Being able to prove that the transformations have the anticipated effect is vital in maintaining interoperability and generating timely responses to temporal queries regarding the disease surveillance. Not only does it allow to check that data and ontology updates do not conflict with the domain knowledge, it allows to safely work with disparate and distributed data sources, most of which are outside of the end-users' control to modify. Being able to prove the existence of a conflict or an inconsistency is a step forward in adressing the interoperability problem. The fact that this verification can be automated makes it also much easier for anyone to manage the interoperability between heterogeneous dynamic knowledge bases in the field. Verification of graph transformation is still a recent development and there are many opportunities to improve the tools and formalisms presented in this manuscript.

Multiple stakeholders will benefit from utilizing this graph-based toolset including but are not limited to knowledge engineers, epidemiologists and biostatisticians and surveillance experts. The choice of the logic(s) used to express the conditions and label the graphs is one of the most important decisions that needs to be made in our proposed framework. While making this decision, there is a lot of ground that needs to be covered at the border between tractability or decidability and expressive power required for each specific task. Furthermore, in particular in Descriptions Logic, the validity of a formula is an unusual problem for which efficient tools and heuristics need to be researched and created. This means that the complete automation of the verification is not yet achieved. There are also many tangential ways to look at the problem from finding ways to use the values in the data in the logics, to automatic generation of specifications and transformations induced by known transformations to different definitions of graphs and the logic adapted to deal with them. To the best of our knowledge, this work is the first attempt to exploit the theories of graph transformation and verification to address change management and evolution in distributed global health surveillance systems and resources.

Our future works will focus on different directions. In this paper, we have generally assumed that the verification happened in a context where the exact changes to the data or the ontologies are known. This is not always the case in many real-world applications, where many types of changes may occur unplanned. So, we will continue our efforts on automatically identifying and classifying these changes. Furthermore, the ability to use different types of logics increases the expressivity of the specifications, however, using expressive logics may cause verification problems and an added complexity. We are including more expressive constructors, e.g., transitivity, as well as finding fragments for which the verification is tractable.

ACKNOWLEDGMENTS

This work was submitted on 08/08/2018.

REFERENCES

[1] World Health Organisation. *The Top 10 Causes of Death, 2015*. Accessed: Jun. 28, 2017. [Online]. Available: https://www.who.int/mediacentre/factsheets/fs310/en/index1.html
[2] "World malaria report 2017," World Health Org., Geneva, Switzerland, Tech. Rep., 2017.
[3] S. Hales, S. J. Edwards, and R. S. Kovats, *Impacts on Health of Climate Extremes*. 2003, pp. 79–102.
[4] J. Sachs and P. Malaney, "The economic and social burden of malaria," *Nature*, vol. 415, no. 6872, pp. 680–685, Feb. 2002, doi: 10.1038/415680a.
[5] S. Lozano-Fuentes, A. Bandyopadhyay, L. G. Cowell, A. Goldfain, and L. Eisen, "Ontology for vector surveillance and management," *J. Med. Entomology*, vol. 50, no. 1, pp. 1–14, Jan. 2013.
[6] P. Topalis, E. Mitraka, V. Dritsou, E. Dialynas, and C. Louis, "IDOMAL: The malaria ontology revisited," *J. Biomed. Semantics*, vol. 4, p. 16, Sep. 2013, doi: 10.1186/2041-1480-4-16.
[7] *Malaria Ontology*. Accessed: Jul. 2, 2017. [Online]. Available: https://bioportal.bioontology.org/ontologies/IDOMAL
[8] E. Dialynas, P. Topalis, J. Vontas, and C. Louis, "MIRO and IRbase: IT tools for the epidemiological monitoring of insecticide resistance in mosquito disease vectors," *PLOS Neglected Tropical Diseases*, vol. 3, no. 6, p. e465, Jun. 2009, doi: 10.1371/journal.pntd.0000465.
[9] *Mosquito Insecticide Resistance Ontology*. Accessed: Jul. 2, 2017. [Online]. Available: https://bioportal.bioontology.org/ontologies/MIRO
[10] A. Shaban-Nejad and V. Haarslev, "Bio-medical ontologies maintenance and change management," in *Biomedical Data and Applications*. 2009, pp. 143–168, doi: 10.1007/978-3-642-02193-0_6.
[11] H. H. Goldstine and J. von Neumann, Eds., *Planning and Coding of Problems for an Electronic Computing Instrument*. Princeton, NJ, USA: Institute for Advanced Study, 1947.
[12] A. M. Turing, "Checking a large routine," in *Proc. Rep. Conf. High Speed Autom. Calculating Machines*, 1949.
[13] *Sparql Protocol and RDF Query Language*. Accessed: Jul. 2, 2017. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/
[14] A. Shaban-Nejad and V. Haarslev, "Managing changes in distributed biomedical ontologies using hierarchical distributed graph transformation," *Int. J. Data Mining Bioinf.*, vol. 11, no. 1, pp. 53–83, Dec. 2015, doi: 10.1504/IJDMB.2015.066334.
[15] A. Shaban-Nejad and V. Haarslev, "Incremental biomedical ontology change management through learning agents," in *Proc. KES Int. Symp. Agent Multi-Agent Syst., Technol. Appl.*, Incheon, South Korea, Mar. 2008, pp. 526–535, doi: 10.1007/978-3-540-78582-8_53.
[16] A. Shaban-Nejad, O. Ormandjieva, M. Kassab, and V. Haarslev, "Managing requirement volatility in an ontology-driven clinical lims using category theory," *Int. J. Telemedicine Appl.*, vol. 2009, Dec. 2009, Art. no 917826. [Online]. Available: http://arxiv.org/abs/0906.1842

[17] A. Rensink, Á. Schmidt, and D. Varró, "Model checking graph transformations: A comparison of two approaches," in *Proc. Int. Conf. Graph Transformation*, in Lecture Notes in Computer Science, vol. 3256. Rome, Italy: Springer, 2004, pp. 226–241.

[18] A. Habel and K. Pennemann, "Correctness of high-level transformation systems relative to nested conditions," *Math. Struct. Comput. Sci.*, vol. 19, no. 2, pp. 245–296, 2009, doi: 10.1017/S0960129508007202.

[19] C. M. Poskitt and D. Plump, "A hoare calculus for graph programs," in *Proc. ICGT*, 2010, pp. 139–154, doi: 10.1007/978-3-642-15928-2_10.

[20] P. Baldan, A. Corradini, and B. König, "A framework for the verification of infinite-state graph transformation systems," *Inf. Comput.*, vol. 206, no. 7, pp. 869–907, 2008.

[21] P. Balbiani, R. Echahed, and A. Herzig, "A dynamic logic for term-graph rewriting," in *Proc. 5th Int. Conf. Graph Transformations (ICGT)*, in Lecture Notes in Computer Science, vol. 6372. Springer, 2010, pp. 59–74.

[22] J. H. Brenas, R. Echahed, and M. Strecker, "Proving correctness of logically decorated graph rewriting systems," in *Proc. 1st Int. Conf. Formal Struct. Comput. Deduction (FSCD)*, Porto, Portugal, Jun. 2016, pp. 14:1–14:15, doi: 10.4230/LIPIcs.FSCD.2016.14.

[23] B. König and J. Esparza, "Verification of graph transformation systems with context-free specifications," in *Proc. Int. Conf. Graph Transformation*, in Lecture Notes in Computer Science, vol. 6372. Enschede, The Netherlands: Springer, Sep./Oct. 2010, pp. 107–122.

[24] A. Schürr, "Logic based programmed structure rewriting systems," *Fundamenta Informaticae*, vol. 26, nos. 3–4, pp. 363–385, 1996, doi: 10.3233/FI-1996-263407.

[25] J. H. Brenas, R. Echahed, and M. Strecker, "Verifying graph transformation systems with description logics," in *Proc. Int. Conf. Graph Transformation*, Toulouse, France, Jun. 2018, pp. 155–170, doi: 10.1007/978-3-319-92991-0_10.

[26] R. Echahed, "Inductively sequential term-graph rewrite systems," in *Proc. 4th Int. Conf. Graph Transformations (ICGT)*, in Lecture Notes in Computer Science, vol. 5214. Springer, 2008, pp. 84–98.

[27] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge Univ. Press, 2003.

[28] *Web Ontology Language*. Accessed: Mar. 22, 2018. [Online]. Available: https://www.w3.org/2001/sw/wiki/OWL

[29] J. H. Brenas, R. Echahed, and M. Strecker, "C2PDLS: A combination of combinatory and converse PDL with substitutions," in *Proc. 8th Int. Symp. Symbolic Comput. Softw. Sci. (SCSS)*, Gammarth, Tunisia, Apr. 2017, pp. 29–41. [Online]. Available: http://www.easychair.org/publications/paper/C2PDLS_A_Combination_of_Combinatory_and_Converse_PDL_with_Substitutions

[30] E. Gradel, M. Otto, and E. Rosen, "Two-variable logic with counting is decidable," in *Proc. 12th IEEE Symp. Logic Comput. Sci. (LICS)*, Warschau, Poland, 1997. [Online]. Available: http://www.logic.rwth-aachen.de/pub/graedel/gorc2.ps

[31] C. A. R. Hoare, "An axiomatic basis for computer programming," *Commun. ACM*, vol. 12, no. 10, pp. 576–580, 1969, doi: 10.1145/363235.363259.

[32] J. H. Brenas, R. Echahed, and M. Strecker, "Ensuring correctness of model transformations while remaining decidable," in *Proc. Int. Colloq. Theor. Aspects Comput.*, Taipei, Taiwan, Oct. 2016, pp. 315–332, doi: 10.1007/978-3-319-46750-4_18.

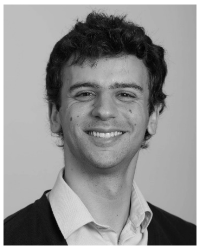
[33] The OBO Foundry's Operations Committee. *Open Biological and Biomedical Ontology Foundry*. Accessed: Mar. 14, 2018. [Online]. Available: http://obofoundry.org

[34] National Center for Biotechnology Information. *National Center for Biotechnology Information Organismal Classification*. Accessed: Mar. 14, 2018. [Online]. Available: https://bioportal.bioontology.org/ontologies/NCBITAXON

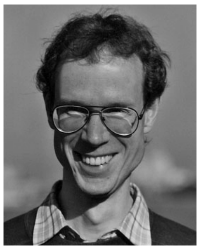
[35] M. S. A. Manir, J. H. Brenas, C. J. O. Baker, and A. Shaban-Nejad, "A surveillance infrastructure for malaria analytics: Provisioning data access and preservation of interoperability," *JMIR Public Health Surveill.*, vol. 4, no. 2, p. e10218, 2018.

[36] J. H. Brenas, M. S. A. Manir, C. J. O. Baker, and A. Shaban-Nejad, "A malaria analytics framework to support evolution and interoperability of global health surveillance systems," *IEEE Access*, vol. 5, pp. 21605–21619, 2017, doi: 10.1109/ACCESS.2017.2761232.

[37] J. H. Brenas, M. S. A. Manir, K. Zinszer, C. J. O. Baker, and A. Shaban-Nejad, "Exploring semantic data federation to enable malaria surveillance queries," *Studies Health Technol. Inform.*, vol. 247, pp. 6–10, 2018, doi: 10.3233/978-1-61499-852-5-6.



JON HAËL BRENAS received a double master's degree in mathematical modeling and digital imagery specialized in modeling, computation and simulation from the ENSIMAG, Grenoble, France, and in computer engineering from the Politecnico di Torino, Torino, Italy, and the Ph.D. degree from Université Grenoble Alpes, Grenoble, France. He is currently a Post-Doctoral Fellow with the Oak Ridge National Laboratory, Center for Biomedical Informatics, and the Department of Pediatrics at the University of Tennessee Health Science Center, Memphis, TN, USA. His Ph.D. dissertation focus was on the verification of graph transformation. His research interests lie in graphs' and graph transformations' application to biomedical and health informatics.



MARTIN STRECKER received the Diploma degree in computer science from the Technical University of Darmstadt, Germany, and the INP Grenoble, and the Ph.D. degree from the University of Ulm, where he was involved on the topic of program and proof development in Type Theory. He was a Teaching and Research Associate at the University of Ulm. He then joined Etas GmbH, a subsidiary of the Bosch group, where he was responsible for the quality assurance of a code generator used in the embedded (in particular automotive) market. He then participated in the Verificard project at the TU München which resulted in the definition of a semantics for Java and the Java runtime system and, in particular, the verification of a Java source to Java bytecode compiler. Since joining the faculty of the Université de Toulouse in 2004, he has participated in projects in particular on the verification of graph transformations and possible applications in the database domain.

RACHID ECHAHED received the M.Sc. degree, the Ph.D. degree, and the Habilitation degree. He has been a Researcher at the CNRS (the French Centre National de la Recherche Scientifique) since 1991. He is also the Head of the Computation, Algorithm, Programs and Proofs (CAPP) Group, Laboratoire d'Informatique de Grenoble, Université Grenoble Alpes, Grenoble, France. The research topics investigated by the CAPP group belong to the domains of theoretical foundations of programming, automated reasoning and quantum computing. His main research interests are around graph and term rewriting and narrowing, program verification, and multiparadigm programming.

ARASH SHABAN-NEJAD received the M.Sc. and Ph.D. degrees in computer science from Concordia University, Montreal, and the Master of Public Health from the University of California at Berkeley. He is currently an Assistant Professor with the OAK-Ridge National Lab (ORNL), Center for Biomedical Informatics, and the Department of Pediatrics at the University of Tennessee Health Science Center (UTHSC), Memphis, TN, USA. He is also an Adjunct Faculty at the Bredesen Center for Interdisciplinary Research and Graduate Education, University of Tennessee, Knoxville, TN, USA. Prior to joining UTHSC-ORNL, he was a Post-Doctoral Fellow with the McGill Clinical and Health Informatics Group, McGill University. Additional training was accrued at the Harvard School of Public Health. His primary research interest is clinical and population health intelligence, epidemiologic surveillance and big-data semantic analytics using tools and techniques from artificial intelligence, knowledge representation, and semantic Web. He is also the Principal Investigator in a global health and development research project for malaria elimination, funded by the Bill & Melinda Gates Foundation.