frontiers in Big Data

ORIGINAL RESEARCH
published: 03 May 2021
doi: 10.3389/fdata.2021.660101

Proving the Correctness of Knowledge Graph Update: A Scenario From Surveillance of Adverse Childhood Experiences

Jon Haël Brenas[1]* *and Arash Shaban-Nejad*[2]*

[1] Nuffield Department of Public Health, Big Data Institute, University of Oxford, Oxford, United Kingdom, [2] Department of Pediatrics, The University of Tennessee Health Science Center-Oak Ridge National Laboratory, Center for Biomedical Informatics, College of Medicine, Memphis, TN, United States

Knowledge graphs are a modern way to store information. However, the knowledge they contain is not static. Instances of various classes may be added or deleted and the semantic relationship between elements might evolve as well. When such changes take place, a knowledge graph might become inconsistent and the knowledge it conveys meaningless. In order to ensure the consistency and coherency of dynamic knowledge graphs, we propose a method to model the transformations that a knowledge graph goes through and to prove that the new transformations do not yield inconsistencies. To do so, we express the knowledge graphs as logically decorated graphs, then we describe the transformations as algorithmic graph transformations and we use a Hoare-like verification process to prove correctness. To demonstrate the proposed method in action, we use examples from Adverse Childhood Experiences (ACEs), which is a public health crisis.

Keywords: program verification, graph transformation, cloning, merging, knowledge graph, adverse childhood experiences

OPEN ACCESS

Edited by:
Mayank Kejriwal,
University of Southern California,
United States

Reviewed by:
Yanghua Xiao,
Fudan University, China
Ivan Lee,
University of South Australia, Australia

***Correspondence:**
Jon Haël Brenas
jon.brenas@bdi.ox.ac.uk
Arash Shaban-Nejad
ashabann@uthsc.edu

Specialty section:
This article was submitted to Data Mining and Management, a section of the journal Frontiers in Big Data

Received: *28 January 2021*
Accepted: *06 April 2021*
Published: *03 May 2021*

Citation:
Brenas JH and Shaban-Nejad A (2021) Proving the Correctness of Knowledge Graph Update: A Scenario From Surveillance of Adverse Childhood Experiences. Front. Big Data 4:660101. doi: 10.3389/fdata.2021.660101

1. INTRODUCTION

Knowledge graphs have become ubiquitous as a framework to represent entities and the relations that connect them. Thanks to the layer of semantic information, it has become possible to add meaning to the entities and relations contained in graphs and to reason about the knowledge they contain. Because graphs contain knowledge, they are expected to change with new information added or removed depending on outside events. In a similar way, the changes in the semantic layers may cause the meaning associated with each entity to change. Through such modifications, a knowledge graph can become inconsistent with the ontology that describes it (Zhang, 2002), rendering the knowledge, and thus the graph, meaningless.

In this paper, we propose a method to tackle this issue. The proposed method aims to identify what are the transformations entailed by adding or removing a piece of information or an axiom to make sure that the knowledge graph remains consistent with the ontology. In our previous works, we used graph transformation to represent and analyze changes in knowledge-based global health surveillance systems (Brenas et al., 2017, 2018; Al-Manir et al., 2018). We represent the ontology in $\mathcal{C}^2$ (Gradel et al., 1997) for convenience. The initial modification of the knowledge graphs will be assumed to be through a query language but we will represent the transformations as graph transformations and use Hoare-like verification methods (Hoare, 1969) to produce the proofs. The

modifications depend only on the structure of each axiom and the action that is performed and not on the actual ontology or graph. It is thus possible to prove that they behave correctly in an abstract way that is independent of the actual knowledge graph that is modified. In particular, this means that the complexity of the verification task is independent of the size of the actual knowledge graph and only depends on the size of the specification that is proven to be correct.

To showcase how the proposed method works, we use examples coming from Adverse Childhood Experiences (ACEs), which is a major public health concern. ACEs are negative events, e.g., abuse, witnessing violence, etc, that have been linked to various negative health outcomes and risky behaviors (Felitti et al., 1998). The ACEs Ontology and knowledge graph have been described in Brenas et al. (2019a,c).

Here, we will use a clinical example. Patients or their parents are interviewed and screened for ACEs. If they suffer from some ACEs, e.g., if they are homeless or live in a place with mold, they are assigned to a social worker. If they suffered from a different set of ACEs, e.g., emotional abuse, they are assigned to a psychologist. Additionally, if they agree to be part of a study, they need to have provided a phone number or an email address and they will be paired with a caseworker. In order to avoid overworking the staff, social workers can only be assigned 10 patients and psychologists five. If the number of patients is higher than the available number of professionals, they are redirected toward a different clinic and it is agreed that a new hire is required.

In Section 2, we will introduce the formal framework that underpins our method. In Section 3, we show the applicability of this framework through some examples. Finally, in Section 4, we discuss the limitations of the method and further work.

2. LOGIC, GRAPH REWRITING, AND VERIFICATION

Each knowledge graph is composed of a graph and an ontology that assigns meaning to its nodes and edges. In the following, we present a fairly informal and succinct introduction to our framework. We hope to manage to give readers an idea of the logical foundation of our method without spending too much time and space on information that would not be relevant to most end-users. Readers interested in a more formal and in-depth description can find it in Brenas et al. (2016a, 2018b).

In order to represent the knowledge graphs, we use logically decorated graphs where both nodes and edges are labeled with formulae that are part of the ontology language.

Definition 2.1. *Let $\mathcal{C}$ (resp. $\mathcal{R}$) be a set of node labels (resp. edge labels), a logically decorated graph is a tuple $(N, E, \Phi_N, \Phi_E, s, t)$ where N is a set of nodes, E is a set of edges, $\Phi_N : N \rightarrow \mathcal{P}(\mathcal{C})$ is a node labeling function, $\Phi_E : E \rightarrow \mathcal{P}(\mathcal{R})$ is an edge labeling function, $s : E \rightarrow N$ is a source function and $t : E \rightarrow N$ is a target function. The source function s and the target function t define the orientation of an edge. For instance, edge e connects node $s(e)$ to node $t(e)$.*

The set $\mathcal{C}$ (resp. $\mathcal{R}$) naturally depends on the logic. It is constructed such that $\mathcal{C}$ contains all "unary" (resp. "binary") predicates of the logic and the closure under their constructors, i.e., class (resp. property) assertions.

To make our reasoning easier to understand, we describe axioms from the ontology both as first-order formulae and in a notation closer to Description Logics. In this paper, we focus on the lower spectrum of the expressivity for the axioms but the verification framework that underpins our method works with the full expressiveness of $\mathcal{C}^2$ (Gradel et al., 1997), the two-variable fragment of first-order logic with counting, that can express most axioms in OWL.[1] Informally, $\mathcal{C}^2$ contains all formulas of first order logic that can be written using only two variables, as well as the counting quantifier $\exists^{>n}$ and its negation $\exists^{\leq n}$. As an example, $\exists^{>2} x.\phi(x)$ can be translated in first order logic as $\exists x_0, x_1.\phi(x_0) \wedge \phi(x_1) \wedge x_0 \neq x_1$, i.e., there exist at least two different x such that $\phi(x)$.

To explain how knowledge graphs are updated and modified, we use graph transformations. The most usual way to deal with graph transformations is by using an algebraic approach rooted in category theory (Barendregt et al., 1987). In this paper, we will use a more algorithmic approach that, we argue, is more suited to verification. In order to build transformations, we first define atomic actions that will be composed to form more elaborate actions.

Definition 2.2. *Let $\mathcal{C}$ (resp. $\mathcal{R}$) be a set of node (resp. edge) labels. 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 (resp. it deletes i and all its incoming or outgoing edges).*
- *a node label addition $add_C(i, c)$ (resp. node label deletion $del_C(i, c)$) where i is a node and c is a label in $\mathcal{C}$. 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 r is an edge label in $\mathcal{R}$. It adds the edge e with label r between nodes i and j (resp. removes all edges with source i and target j with label r), i.e., $s(e) = i$, $t(e) = j$ and $\Phi_E(e) = r$ (resp. $\forall e' \in E. s(e') \neq i$ or $t(e') \neq j$ or $\Phi_E(e') \neq r$).*

An action α is a finite sequence of atomic actions.

Actions are not enough to describe all the transformations we want to perform, as they require the knowledge of the exact nodes and edges that are going to be modified. In order to enable the selection of nodes that satisfy a condition, we use rewriting rules and logically decorated graph rewriting systems.

Definition 2.3. *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 $\rightarrow$ α. A logically decorated graph rewriting system is a set of rules.*

[1] Web Ontology Language (OWL) (accessed on January 2021).

FIGURE 1 | A rule that creates a new node and connects it to a previously existing one.

Example 2.1. ***Figure 1*** *contains an example of rule. It selects a node, b, and creates a new node named a ($add_N(a)$) and connects a to b with an edge labeled with lives_with ($add_E(a, b, lives\_with)$).*

A rule is applied when a match for the left-hand side is found in the knowledge graph that we want to modify. We also define strategies that describe the order in which the rules of a logically decorated graph rewriting system are to be applied.

Definition 2.4. *Given a logically decorated graph rewrite system* GRS*, a strategy is a word of the following language defined by s, where ρ is any rule in* GRS*:*

$s :=$	ϵ	*(Empty Strategy)*	ρ	*(Rule)*
	$s \oplus s$	*(Choice)*	$s; s$	*(Composition)*
	s^*	*(Closure)*		

Given two logically decorated graphs G and G′ and a strategy s, we denote by $G \Rightarrow_s G'$ that it is possible to obtain G′ by applying s to G.

Intuitively, ϵ means "do nothing," ρ is an application of the rule, $s \oplus s$ is the application of either of the two strategies (but not both), $s; s$ is the application composition and s^* applies the strategy as many times as possible.

Example 2.2. *For instance, strategy $\rho_0; (\rho_0^* \oplus \rho_1)$ means "apply ρ_0 followed by either as many applications of ρ_0 as possible or one application of ρ_1."*

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 2.5 (Specification). *A* specification *SP is a triple $\{Pre\}(\mathcal{R}, s)\{Post\}$ where Pre (the precondition) and Post (the post-condition) are formulas (of a given logic), $\mathcal{R}$ is a graph rewriting system and s is a strategy.*

Example 2.3. *Let us assume that we want a specification describing part of Example . $\phi_0 \equiv \forall x.Psychologist(x) \Rightarrow \exists^{\leq 5} y.(Patient(y) \wedge assigned\_to(y, x))$ is a formula that states that at most five patients are assigned to each psychologist. Then, $SP_0 \equiv \{\phi_0\}(\{\rho_0\}, \rho_0^*)\{\phi_0\}$ states that if ϕ_0 is true, it will still be true after applying ρ_0 as many times as possible.*

Definition 2.6 (Correctness). *A specification SP is said to be* correct *iff for all graphs G, G′ such that $G \Rightarrow_s G'$ and G is a model of Pre (i.e., Pre is a logical consequence of the labeling of G), then G′ is a model of Post.*

In order to prove the correctness of a specification, we use a Hoare-like approach (Hoare, 1969). 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 post-condition that needs to be achieved, it becomes possible to generate the weakest precondition that ensures that the post-condition 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 post-condition Q. More details can be found in Brenas et al. (2016a). The definitions of these functions are given in **Figures 2**, **3**, 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 2.7 (Substitutions). *To each elementary action a is associated a* substitution*, written $[a]$, such that for all graphs G and formula ϕ, (G is a model of $\phi[a]$) $\Leftrightarrow$ ($G[a]$ is a model of ϕ).*

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 2.8 (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 2.4. *As the strategy in Example 2.3 contains a closure, we annotate it. $\{\phi_0\}(\{\rho_0\}, \rho_0^*\{\phi_0\})\{\phi_0\}$ is a possible annotated specification.*

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 2.9 (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 2.1 (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 is a model of Pre implies G′ is a model of 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 this section. Another possible choice is to only consider tractable logic so that verification becomes achievable in a reasonable timeframe.

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

FIGURE 2 | 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 3 | Verification conditions for strategies.

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. In the following, we will be using $\mathcal{C}^2$.

Theorem 2.2. *(Brenas et al., 2016a) $\mathcal{C}^2$ 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.

3. APPLICATION

When adding or removing knowledge from a knowledge graph, there is a risk to harm its consistency with the ontology that describes its underlying structure. As a result, additional actions may be needed before making any change to a knowledge graph. In this section, we present some of the transformations that can be enacted to make an update. We will then prove that the specifications resulting from the axioms and the transformations under consideration are correct and, thus, the update can take place.

To show some examples, we use the Adverse Childhood Experiences Ontology (ACESO) (Brenas et al., 2019a) as the source for the axioms. The systematic study of adverse childhood experiences and their health outcomes is recent and new data and knowledge are routinely added in this field. As a result, both the knowledge graphs and the associated ontology are likely to change frequently.

Example 3.1. *Let us assume that the change that we want to perform is the addition of a new node, pa, which is labeled with PhysicalAbuse. This modification would be equivalent to performing the action $a_0 \equiv add_N(pa)$; $add_C(pa,$ PhysicalAbuse). ACESO contains the axiom PhysicalAbuse $\subseteq$ Abuse that can be translated in $\mathcal{C}^2$ as $\forall x.PhysicalAbuse(x) \Rightarrow Abuse(x)$. Performing a_0 would make the knowledge graph inconsistent with the ontology as, if $x = pa$, $PhysicalAbuse(x) \wedge \neg Abuse(x)$ is true. Instead, the transformation that is actually performed is $a_1 \equiv add_N(pa)$; $add_C(pa, PhysicalAbuse)$; $add_C(pa, Abuse)$. In that case, the correctness formula is $(\forall x.PhysicalAbuse(x) \Rightarrow Abuse(x)) \Rightarrow (\forall x.(PhysicalAbuse(x) \vee x = pa) \Rightarrow (Abuse(x) \vee x = pa))$ which is obviously valid. The knowledge graph is thus still consistent with the ontology.*

As it is possible to modify the knowledge graph by adding information contained in new nodes, it is also possible to add new edges. In the previous example, only the new node, *pa*, was really affected by the modification of the knowledge graph and thus a simple action was enough to update the knowledge graph to preserve consistency. The next example, on the other hand, requires a rule to be applied.

Example 3.2. *Let us assume that the change that we want to perform is the creation of a new node, a, and a new edge, e, linking a to the already existing node b with an edge labeled lives_with. This modification would be equivalent to applying Rule ρ_0 shown in **Figure 4**. ACESO contains the axiom Symmetric(lives_with) that*

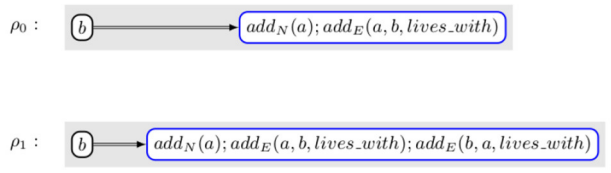

FIGURE 4 | Rules creating a new node *a* and connecting it via property *lives_with* to the node *b*.

can be translated in C^2 *as* $\forall x, y.lives\_with(x, y) \Rightarrow lives\_with(y, x)$. *Applying* ρ_0 *would make the knowledge graph inconsistent with the ontology as, if* $x = a$ *and* $y = b$, $lives\_with(x, y) \wedge \neg lives\_with(y, x)$ *is true. Instead, the transformation that is actually performed is the application of rule* ρ_1 *shown in* ***Figure 4****. In that case, the correctness formula is* $(\forall x, y.lives\_with(x, y) \Rightarrow lives\_with(y, x)) \Rightarrow (\forall x, y.(lives\_with(x, y) \vee (x = a \wedge y = b) \vee (x = b \wedge y = a)) \Rightarrow (lives\_with(y, x) \vee (x = b \wedge y = a) \vee (x = a \wedge y = b))$ *which is obviously valid.*

From the previous two examples, one can observe how the transformations and the axioms interact. As a result, it is possible to define a system of transformation rewriting rules that, given the intended action and the form of an axiom, generates a new transformation of the knowledge graph that can be proved correct independently of the actual knowledge graph and ontology. For instance, the previous examples show that when in presence of an axiom of the form $C \subseteq D$ where C and D are unary predicates (i.e., classes in OWL Lite), an elementary action $add_C(i, C)$ should be replaced with the action $add_N(i, C)$; $add_N(i, D)$ while, in presence of an axiom of the form $Symmetric(R)$ where R is a binary predicate (i.e., a property in OWL Lite), an elementary action $add_N(i)$ should be left unchanged. **Figure 5** shows these two rules plus additional ones.

Example 3.3. *Let us now combine the previous two examples. We assume that we want to add a new person Alice, represented by the new node a, to the knowledge graph. We additionally want to insert the knowledge that Alice is a girl and that she lives with Bob, represented by the already existing node b. The initial rule that would be applied is rule* ρ_2 *from* ***Figure 6****. Provided that the ontology contains the axioms* $Girl \subseteq Child$ *and* $Symmetric(lives\_with)$, *all six rules in* ***Figure 5*** *can be applied. The order in which they are applied does not change the final result, i.e., the rewriting system is convergent, and the final result is Rule* ρ_3 *in* ***Figure 6****. As it is possible to prove that each one of the transformation rewrite rules is correct, their combination is correct as well.*

One of the key advantages of using abstract rules to represent the modification of the transformations is that it becomes possible to use the same rule for multiple transformations. In actual cases, the ontology is likely to contain many more axioms and, in particular, to contain a much more developed hierarchy of classes and properties. However, assuming that the ontology contains additional axioms, e.g., $Girl \subseteq Female$ and $Child \subseteq Person$, the same transformation rewrite rule χ_1 can be applied several times introducing in the action not only $add_C(a, Child)$, as shown in **Figure 6**, but also, $add_C(a, Female)$ and $add_C(a, Person)$ triggered by the first application of the rule. The fact that the rules are iteratively applied does not create a risk of infinite looping provided we assume that there is a mechanism to check that the rule does not already contain the added elementary actions, given that doing the same elementary action twice yields the same result as doing it only once. Similarly, in an actual execution, the rules that do not modify the transformation, e.g., χ_0 would not be part

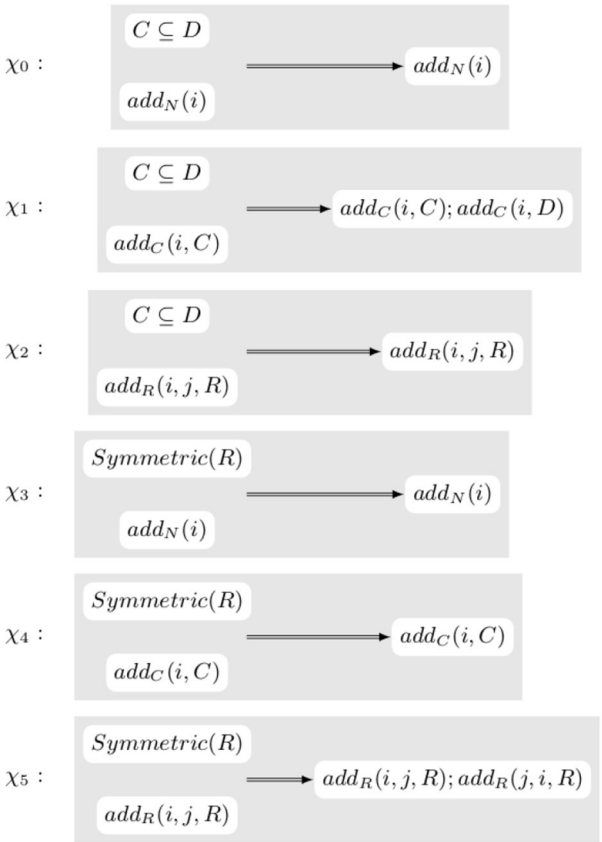

FIGURE 5 | Rules rewriting transformations depending on the form of the axioms in the ontology.

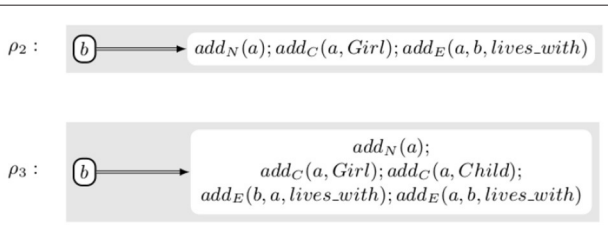

FIGURE 6 | Rules inserting the knowledge that Alice, who did not exist in the knowledge base, is a girl who lives with Bob.

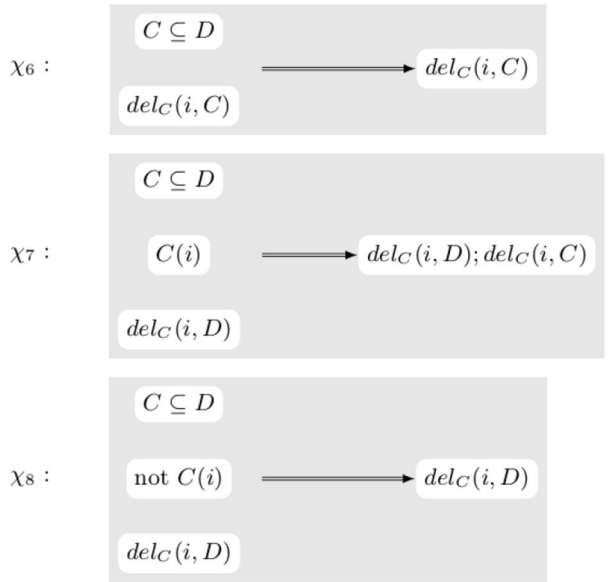

FIGURE 7 | Examples of rules rewriting transformations when deleting the label of node.

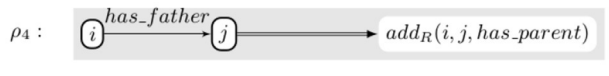

FIGURE 8 | A rule that adds the fact that *has_father* ⊆ *has_parent*.

of the system. We only show them to make our reasoning clearer and easier to understand.

Hitherto, we have only added knowledge to the graph and not removed any. Removing knowledge is not practically much more difficult, at least as long as we keep working with less expressive axioms.

Example 3.4. ***Figure 7*** *contains some transformation rules dealing with node label deletion.* χ_6 *is similar to the rules in* ***Figure 5*** *and doesn't actually modify the transformation. On the other hand, rules* χ_7 *and* χ_8 *are dependent on the current content of the knowledge graph. Indeed, if* $C(i)$ *is true, the axiom* $C \subseteq D$ *is no longer true after the application of* $del_C(i, D)$*. Hence the rule need not only to look at the ontological part of the knowledge graph but also at the graph itself.*

Up to now, we have modified the graph by adding and changing the labeling of the knowledge graph but we have not modified the ontology itself. It is worth pointing that the choices that we made when the transformation presented a risk of inconsistency always lead us to modify the graph and not the ontology. There is, however, no reason to think that when there is a conflict between the ontology and the content of the graph, the ontology is always to be considered correct.

Example 3.5. *In the first example we presented, we chose when adding the fact that pa was a PhysicalAbuse knowing that* $PhysicalAbuse \subseteq Abuse$ *to also add the fact that pa is an Abuse. We could also have decided to remove the axiom and the knowledge graph would have been consistent with the modified ontology.*

As the last example, we will add a new axiom to the ontology. Until now, we have only used the left-hand side of the rules to look at specific instances, either of axioms, elementary actions or individual nodes and we only, perhaps, added a new elementary action to a given rule. When modifying the ontology, a new rule is added that will modify the graph.

Example 3.6. *Let us assume that we want to add the axiom* $has\_father \subseteq has\_parent$ *to the ontology. Rule* ρ_4 *presented in* ***Figure 8*** *is applied to the graph with Strategy* ρ_4^**. The correctness formula, in that case, is, after some simplification,* $(\forall x, y.$ $has\_father(x, y) \Rightarrow has\_parent(x, y)) \Rightarrow (\forall x, y.$ $has\_father(x, y) \Rightarrow has\_parent(x, y))$*, an obvious tautology.*

Example 3.7. *Let us now consider the more elaborate example of Example . Let us define* $\phi_0 \equiv \forall x.Psychologist(x) \Rightarrow \exists^{\leq 5} y.(Patient(y) \wedge assigned\_to(y, x))$*,* $\phi_1 \equiv \forall x.SocialWorker(x) \Rightarrow \exists^{\leq 10} y.(Patient(y) \wedge assigned\_to(y, x))$ *and* $\phi_2 \equiv \forall x.(DefSocialWorker(x) \vee DefPsychologist(x) \vee CaseWorker(x)) \Rightarrow \neg(SocialWorker(x) \vee Psychologist))$*. The precondition and the post-condition will then be* $\phi \equiv \phi_0 \wedge \phi_1 \wedge \phi_2$*.* ϕ_0 *states that psychologists are assigned less than five patients,* ϕ_1 *states that social workers are assigned <10 patients,* ϕ_2*states that default psychologists and social workers and case workers do not count as psychologists or social workers (otherwise, they might not satisfy* ϕ_0 *or* ϕ_1*).*

The set of rules $\mathcal{R}$ *contains the rules shown in* ***Figure 9****.* ρ_5 *looks for a patient suffering from a given type of ACEs, denoted by* $ACEs_0$*, that are not assigned to anyone and for a social worker with nine or fewer patients assigned to them. It then assigns the patient to the social worker.* ρ_6 *does the same for psychologists.* ρ_7 *and* ρ_8 *assign the (possibly) remaining patients to the default options.* ρ_9 *and* ρ_{10} *look for patients with a phone number or an email address, respectively, and assign them to a case worker.* ρ_{11} *and* ρ_{12} *request the hiring of a social worker or a psychologist, respectively, if it had not been requested before and at least one patient is assigned to the default option.*

We chose to apply the strategy $s \equiv (\rho_5 \oplus \rho_6)^*; (\rho_7 \oplus \rho_8)^*; (\rho_9 \oplus \rho_{10})^n; \rho_{11}^*; \rho_{12}^*$*. In order to be able to perform verification, all closures need to be annotated. We annotate all of them with* ϕ*, that is all invariants are the same as the pre- and post-conditions.* $(\rho_9 \oplus \rho_{10})^n$ *denotes n-iterations of the strategy* $\rho_9 \oplus \rho_{10}$*. It is a proper strategy for any given n and the actual choice of n has no impact on the correctness of the specification so we use this construct to let an unspecified number of patients register for the study.* ρ_{11} *and* ρ_{12} *can only be applied at most once each. More elaborate strategy constructors can be defined to require that strategies are applied at most once, for instance, but we did not introduce them in this paper to keep things simpler. The annotated specification is thus* $s' \equiv \{\phi\}(\mathcal{R}, (\rho_5 \oplus \rho_6)^*\{\phi\}; (\rho_7 \oplus \rho_8)^*\{\phi\}; (\rho_9 \oplus \rho_{10})^n; \rho_{11}^*\{\phi\}; \rho_{12}^*\{\phi\})\{\phi\}$*.*

Giving the full proof that the specification is correct would be long and tedious. Instead, let us give an informal description of the

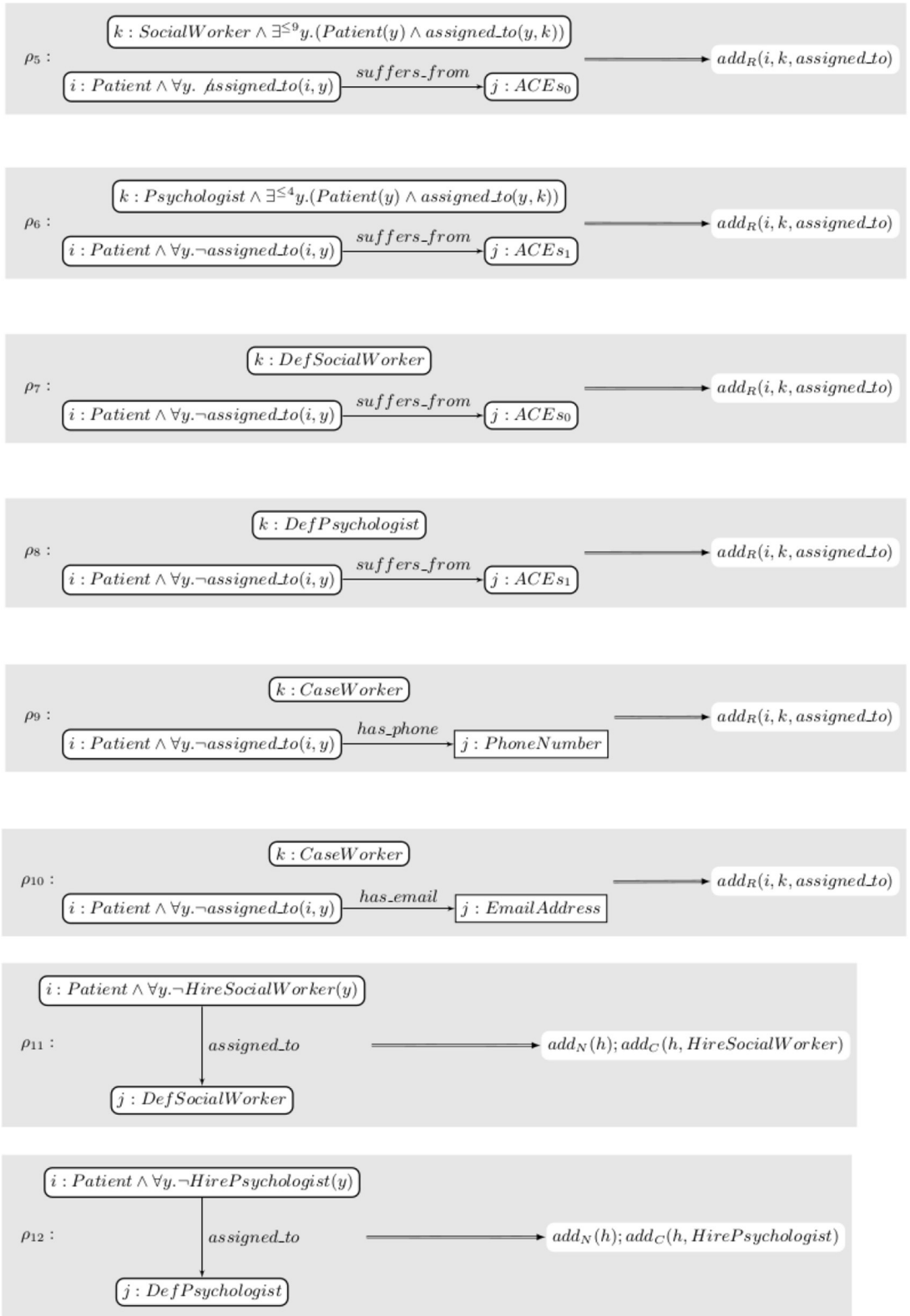

FIGURE 9 | The rules used in Example 3.7.

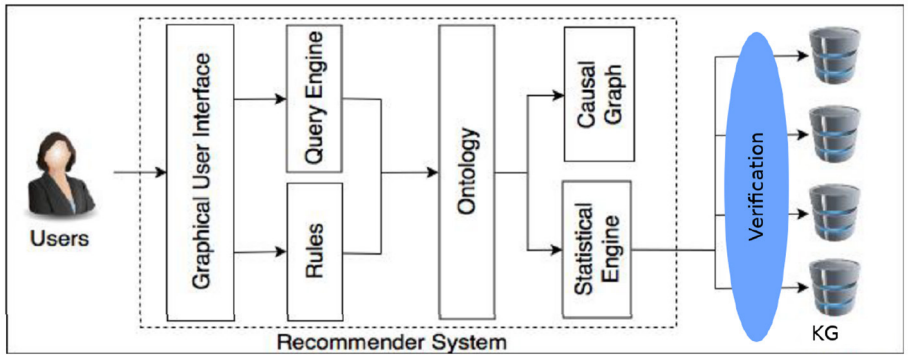

FIGURE 10 | A schema of the recommender system (Brenas et al., 2019b).

proof. Neither ρ_{11} nor ρ_{12} affect any of the predicates in ϕ thus one can ignore them. ρ_9 and ρ_{10} only modify assigned_to(x, y) for y a case worker. From ϕ_2, one gets that being a case worker excludes the possibility of being either a social worker or a psychologist which means that if ϕ is true after $(\rho_5 \oplus \rho_6)^*; (\rho_7 \oplus \rho_8)^*$, it will be true at the end of the execution. Similarly, ρ_7 and ρ_8 only modify the assignment to the default social worker and psychologist that, according to ϕ_2, are not social workers or psychologists. Thus, one needs to prove that $\{\phi\}(\mathcal{R}, (\rho_5 \oplus \rho_6)^*\{\phi\})\{\phi\}$ is correct. ρ_5 only assigns a patient to a social worker that has nine or less and thus, after the assignment, they will only have ten or less. The same argument works for psychologists albeit with a different number of edges. The specification s' is thus correct.

4. CONCLUSION

In this paper, we demonstrated the feasibility of using graph transformations for managing changes and modifications in knowledge graphs while maintaining their consistency. We used logically decorated graphs to represent the knowledge graph, graph transformations for the modifications and a Hoare-like approach to verification to prove the correctness of the transformations.

Verification of graph transformations is an active field of computer science. Our approach is based on the method of Brenas et al. (2016b). Several other approaches exists including methods akin to model checking (Rensink et al., 2004) or the use of graph programming languages, such as GP2 (Wulandari and Plump, 2020). These methods have not been applied to knowledge graphs as far as we know.

This is only the first step toward a comprehensive solution. An important limitation to the application of our method is the fact that not all currently existing knowledge graphs are equipped with ontologies. In the absence of an ontology, the whole problem is moot as there is nothing to be consistent with. That said, it is possible to start with an empty ontology and to use our method to provably make sure that a knowledge graph becomes consistent with it by defining rules that modify it.

We have presented a few transformation rewriting rules and we are working to formulate a formal framework for their definitions. Similarly, we have only looked at the easiest of the interactions between transformations and axioms.

In particular, the rules that we have presented allow automating the process in the situation that we have presented but it is not hard to find examples where such automation is not feasible. When increasing the expressivity of the axioms, there are many situations where a non-trivial choice has to be made to decide which part of the knowledge is responsible for the inconsistency, and thus should be removed. Algorithms exist to do repairs (Bienvenu et al., 2016) that go much further than we did and integrating these algorithms into our method seems promising. That said, the verification process generates a counter model, which is a knowledge graph that invalidates the specification. This counter model can be used to decide the source of the error.

Moreover, we have only presented logics with very little expressivity in this work to make it easier to follow and to avoid problems that arise at higher expressivity. However, it has been proved that the verification that underpins this method works for the whole expressivity of $\mathcal{C}^2$ (Brenas et al., 2018a). Moreover, OWL allows defining more complex axioms on properties, e.g., transitivity, that cannot be expressed in $\mathcal{C}^2$ but can be expressed in other decidable logics, such as the Guarded Fragment of first-order logic (Andréka et al., 1998) for which the same results apply (Brenas et al., 2018b). Extending the expressivity to full first order logic and beyond is much more difficult as satisfiability in first-order logic is undecidable and, as a result, so is our verification process. On the other hand, it is possible to restrict the specifications to less expressive logics. Finding the conditions on the logic and the change that can be performed to reduce the complexity of the verification process is under active study. The importance of the complexity problem is somewhat reduced by the fact that the complexity of the verification task is dependent on the size of the specification and not on the size of the data it represents.

Among the possible applications of the research presented in this paper is the conception of a recommender system to assist in the detection, prevention and treatment of ACEs (Brenas et al., 2019b). **Figure 10** shows a schema of

the recommender system's components. The recommender needs to be able to update its knowledge about the circumstances and the health of the patients in a verifiably correct manner.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

JB and AS-N contributed equally to the research and writing of this paper.

FUNDING

The funding for this study has been provided by the University of Tennessee Health Science Center.

REFERENCES

Al-Manir, M. S., Brenas, J. H., Baker, C. J., and Shaban-Nejad, A. (2018). A surveillance infrastructure for malaria analytics: provisioning data access and preservation of interoperability. *JMIR Public Health Surveill.* 4:e10218. doi: 10.2196/10218

Andréka, H., Németi, I., and van Benthem, J. (1998). Modal languages and bounded fragments of predicate logic. *J. Philos. Logic* 27, 217–274. doi: 10.1023/A:1004275029985

Barendregt, H. P., van Eekelen, M. C. J. D., Glauert, J. R. W., Kennaway, R., Plasmeijer, M. J., and Sleep, M. R. (1987). "Term graph rewriting," in *PARLE, Parallel Architectures and Languages Europe, Volume II: Parallel Languages, June 15–19, 1987, Proceedings* (Eindhoven), 141–158. doi: 10.1007/3-540-17945-3_8

Bienvenu, M., Bourgaux, C., and Goasdoué, F. (2016). "Explaining inconsistency-tolerant query answering over description logic knowledge bases," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12–17, 2016* (Phoenix, AZ), 900–906.

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

Brenas, J. H., Echahed, R., and Strecker, M. (2016a). "Ensuring correctness of model transformations while remaining decidable," in *Theoretical Aspects of Computing–ICTAC 2016–13th International Colloquium, October 24–31, 2016, Proceedings* (Taipei), 315–332. doi: 10.1007/978-3-319-46750-4_18

Brenas, J. H., Echahed, R., and Strecker, M. (2016b). "Proving correctness of logically decorated graph rewriting systems," in *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, June 22–26, 2016* (Porto), 14:1–14:15.

Brenas, J. H., Echahed, R., and Strecker, M. (2018a). On the verification of logically decorated graph transformations. *CoRR* abs/1803.02776.

Brenas, J. H., Echahed, R., and Strecker, M. (2018b). "Verifying graph transformations with guarded logics," in *2018 International Symposium on Theoretical Aspects of Software Engineering, TASE 2018, August 29–31, 2018* (Guangzhou), 124–131. doi: 10.1109/TASE.2018.00024

Brenas, J. H., Shin, E. K., and Shaban-Nejad, A. (2019a). Adverse childhood experiences ontology for mental health surveillance, research, and evaluation: Advanced knowledge representation and semantic web techniques. *JMIR Mental Health* 6:e13498. doi: 10.2196/preprints.13498

Brenas, J. H., Shin, E. K., and Shaban-Nejad, A. (2019b). A hybrid recommender system to guide assessment and surveillance of adverse childhood experiences. *Stud. Health Technol. Inform.* 262, 332–335. doi: 10.5210/ojphi.v11i1.9694

Brenas, J. H., Shin, E. K., and Shaban-Nejad, A. (2019c). An ontological framework to improve surveillance of adverse childhood experiences (ACES). *Stud. Health Technol. Inform.* 258, 31–35. doi: 10.3233/978-1-61499-959-1-31

Brenas, J. H., Strecker, M., Echahed, R., and Shaban-Nejad, A. (2018). Applied graph transformation and verification with use cases in malaria surveillance. *IEEE Access* 6, 64728–64741. doi: 10.1109/ACCESS.2018.2878311

Felitti, V., Anda, R., Nordenberg, D., Williamson, D., Spitz, A., Edwards, V., et al. (1998). Relationship of childhood abuse and household dysfunction to many of the leading causes of death in adults: The adverse childhood experiences (ACE) study. *Am. J. Prev. Med.* 14, 245–258. doi: 10.1016/S0749-3797(98)00017-8

Gradel, E., Otto, M., and Rosen, E. (1997). "Two-variable logic with counting is decidable," in *Proceedings of 12th IEEE Symposium on Logic in Computer Science LICS '97* (Warschau). doi: 10.1109/LICS.1997.614957

Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Commun. ACM* 12, 576–580. doi: 10.1145/363235.363259

Rensink, A., Schmidt, Á., and Varró, D. (2004). "Model checking graph transformations: a comparison of two approaches," in *Graph Transformations*, eds H. Ehrig, G. Engels, F. Parisi-Presicce, and G. Rozenberg (Berlin; Heidelberg: Springer Berlin Heidelberg), 226–241. doi: 10.1007/978-3-540-30203-2_17

Wulandari, G. S., and Plump, D. (2020). "Verifying graph programs with first-order logic," in *Proceedings of the Eleventh International Workshop on Graph Computation Models, Online-Workshop, 24th June 2020, Volume 330 of Electronic Proceedings in Theoretical Computer Science*, eds B. Hoffmann and M. Minas (Open Publishing Association), 181–200. doi: 10.4204/EPTCS.330.11

Zhang, L. (2002). *Knowledge graph theory and structural parsing* (Ph.D. thesis), University of Twente, Twente University Press, Enschede

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.