Response to reviewers

Ontologies of Dynamical Systems and Verifiable ontology-based computation: A Haskell-based implementation of a reference tracking system for medical records as a Case Study

Thomas Bittner, Jonathan Bona and Ceusters Werner

---------- REVIEW 1 ----------
OVERALL EVALUATION: -1 (weak reject)
REVIEWER'S CONFIDENCE: 4 (high)
Scientific or technical quality: 3 (good (middle 1/3))
Novelty or innovation: 1 (not innovative)
Presentation: 3 (good)
References: 2 (important references missing - give details in Review section below)
Recommendation for FOIS Best Paper Award: no

This paper presents a notion of ontology-based computation. The idea of characterizing the correctness and completeness of an implementation with respect to an ontology (what the authors refer to as a computation that adheres to the ontology) is an important idea.

The definition of the notion of a computation that adheres to an ontology is a little vague -- what does it mean to say that data is "well-structured with respect to a given ontology"? Section 3 does not help to elucidate these ideas.

> → *The first two paragraphs of section 2.1 were adjusted to specify these ideas more clearly. And section 3.1 gives an illustration in the context of how to incorporate the BfO categories into the program…*

In the definition of verifiable ontology-based computation, what does it mean to say that "most" code is pure?

> → *Since there is not enough space to respond in full, we removed the word 'most' … and replaced it by 'computation internal to the RT system'*

There is one sentence that I find baffling (in Section 4.1, page 12): ``The more expressive the formal language that is required to specify a formal ontology, the less likely it is that general computation can be formulated as automated deduction."
This is patently false. A decidable logic (such as OWL) is less expressive, but by definition cannot define many computable functions, whereas a logic such a FOL is needed to define computable functions and hence formulate computation as automated deduction.

> → *We rephrased the sentence to make its intended meaning more clear*

On page 4, the authors claim: ``To specify such necessary and sufficient conditions in many cases requires highly expressive languages which in general lack automatic decision procedures that are provably correct." Are the authors saying that they are using logics which are more expressive than FOL i.e. are not even semidecidable? If so, what logics are they proposing?

> → *there is no space to address the logical specifics. Haskell is weak second order and semi-decidable. non-termination is addressed by a haskell program being an executable specification of a computation (understood as a function). This could be addressed in extensor in a journal version but not here. We put something of a summary at the end of section \subsection{Dynamic systems, Situation calculus and planing in Artificial Intelligence:*

The paper overlooks such classical papers as Green, "Application of theorem proving to problem solving", IJCAI-69, - Nilsson and Fikes, "STRIPS:A new approach to the application of theorem, proving to problem solving", - Shanahan, Event Calculus Planning Revisited, which specifically use first-order logic to ``work every ontologically possible state and state change must be derivable from some initial state by means of

automated deduction". The notion of ontology-based computation is also reminiscent of the notion of the Ontological Stance. Also, the notion of theory resolution in automated deduction also seems to be related to ontology-based computation, and should be discussed as Related Work. Ignoring this earlier work seriously undermines the significance of the contribution.

→ *we have included a new subsection to address the above ... (\subsection{Dynamic systems, Situation calculus and planing in Artificial Intelligence}). We had to remove the diagram about the ontology to make space for the new section.*

The paper also addresses the role of reasoning with an ontology and dynamic systems. Again, the classical references to the use of theorem proving for AI planning show how this can be done with suitable axiomatizations, but the authors do not propose anything in this direction. Instead, they claim that a functional programming language (namely Haskell) is used as the basis for ontology-based computing.
Overall, the authors do not convincingly demonstrate their claims that a language such as Haskell support computation only if it is logically and ontologically correct.

→ *we added a subsection .*

Typographical Errors:
Page 12: In the context of ontological computing is is natural should be
In the context of ontological computing it is natural

---------- REVIEW 2 ----------
OVERALL EVALUATION: 1 (weak accept)
REVIEWER'S CONFIDENCE: 4 (high)
Scientific or technical quality: 3 (good (middle 1/3))
Novelty or innovation: 2 (similar to other work but still somewhat innovative)
Presentation: 2 (needs minor improvements - give details in the Review section below)
References: 3 (nothing missing but irrelevant references present)
Recommendation for FOIS Best Paper Award: no

---------- Review ----------

The article presents a methodology for embedding the ontological invariants defined in a particular ontology into the Haskell programming language type system, with the purpose of guaranteeing the ontological consistency of programs written in Haskell to manipulate data that represent information that respect the chosen ontology.
Although there exist OWL to code generators for languages such as Java, the Haskell translation described in the paper makes significant use the Haskell's expressive type system, which allows compile-time verification of the ontological invariants represented using the type system.
Thus, the work described in the article seems to be relevant as an example of Applied Ontology and as an example of the profitable interaction between two distinct Computer Science areas (Applied Ontology and Programming Language Theory).
It also presents a novel idea, since no typeful translation from OWL-DL ontologies to Haskell programs seems to exist yet.
However, the text requires a few improvements, mostly regarding (easily fixable) lack of preciseness and ambiguity on some parts, as detailed bellow:

1) In page 2, starting on line 5, the phrase “In [5], a methodology, based on...”: the fact that in [5] a methodology was tested seems to be irrelevant, however the fact that in [5] a methodology was applied to the EHRs of 570.000 alluded before should be relevant, I believe. If that is the case, the phrase should probably something like “In [5], a methodology ... was applied to the dataset, ensuring that ...”

→ *The first phrase is relevant because the methodology is based on Referent Tracking (RT) and the method proposed here in this paper aims to implement the RT ideas. It is true we did not say this explicitly at this point, so we corrected that.*

2) In page 2, line 10, in the phrase "The work .... to implement computation in ways ...", what is the meaning of computation here? Any algorithm? A pure (impure) Haskell Function? Logical reasoning? Database queries? All of these?

→ *We changed this phrase into "The work presented here builds further on these efforts by introducing a method to write software for processing data in ways that are provably correct with respect to the semantics specified in the ontology."*

3) Page 2: "For an independent continuant to exist at a time means to be in its current state at that time" has two readings:
"For an independent continuant (C) to exist at a time (T1) means to be in its current state (state(C,T2)) at that time (T1)", where T2 is the present, or
"For an independent continuant (C) to exist at a time (T) means to be in its current state (state(C,T)) at that time (T)". If the first one was intended, does it mean that an independent continuant always exists at the same state? If the second one is intended, isn't it a trivial assertion: for a thing to exist at a moment it of course must exist in the state it is at that moment. Given that the next phrase asserts that continuants do change, I presume that the second interpretation is the one intented.

→ *We rephrased it.*

4) Page 3: "For example, in classical mechanics only processes are permitted that conserve energy (of a closed system) in that they transform states in ways that conserve the total energy." would be clearer as "For example, in classical mechanics, the only admissable processes are those that transform states in ways that conserve the total energy of closed systems."

→ *done*

5) Page 4: Replace "To specify such ... requires highly ..." with "The specification of such ... requires ..."

→ *done*

6) Page 5: Consider replacing "Above the ontological importance of the category of states in conjunction with the role of processes in transforming independent continuants from one state to another state and thereby creating sequences of states was discussed." to
"The ontological importance of the category of states in conjunction with the role of processes in transforming independent continuants from one state to another state was discussed above."

→ *done*

7) Page 5: change "Logically states are functional relations ..." to "In logical terms, states are functional relations ...", or "Logically, states are functional relations ..."

→ *done*

8) Page 6: "Since recursion is the main computational technique, a terminating pure Haskell program counts as an inductive proof of a theorem. That is, testing a program often counts as a proof of computational aspects of a program that cannot directly be checked by type inference.". A provably total (terminating) Haskell function counts as a inductive proof of a theorem. However, how does testing a program (in general) counts as a proof of something? Are you trying to argue the relevance of automatic testing tools such as QuickCheck? Further in the paragraph, you refer to "manual proofs" and "automatic, semi-automatic and manual techniques of proof".
What kinds of proofs (and techniques) are you referring to? Machine-verifiable proofs in general? Proofs encoded through types and functions? Proofs written through a specific theorem proving environment, such as Agda, Coq, Isabelle, etc?

→ *this could be addressed in a journal version. good questions though.*

9) Page 12: Missing a comma before "as defined above" in “In fact, many current ontologies that support ontology- based computing as automated deduction as defined above, can specify many of the logical properties of formal relations only informally in the form of annotations”

→ *done*

10) On page 13 (Conclusions), in “However the function-based nature of Haskell ... may open up ...”, what are you contrasting the fact that “the function-based nature of Haskell ... may open up ...” with? Such a constrast seems to be implied by the use of “However” (which also seems to be missing a comma).
In ”may open up possibilities to integrate ontologies into programs that can only be expressed in full first or higher order languages” I believe you mean “may open up possibilities to integrate (into programs) (ontologies that can only be expressed in full first or higher order languages)” and not “may open up possibilities to (integrate ontologies) into (programs that can only be expressed in full first or higher order languages)”.

→ *Rephased.*

---------- REVIEW 3 ----------
OVERALL EVALUATION: 1 (weak accept)
REVIEWER'S CONFIDENCE: 4 (high)
Scientific or technical quality: 2 (fair (bottom 1/3))
Novelty or innovation: 2 (similar to other work but still somewhat innovative)
Presentation: 3 (good)
References: 4 (excellent references)
Recommendation for FOIS Best Paper Award: no

---------- Review ----------
The paper describes how the functional programming language Haskell can be used to develop applications that reason using ontologies. Although the paper is motivated by the authors' referent-tracking approach to management of temporally oriented information in biomedicine, Haskell is not actually applied in this application. Indeed, a weakness of the paper is that there is no discussion of the actual use of Haskell to build any functioning application. Rather, the paper enumerates desirable features of Haskell for the engineering of verifiable ontology-oriented applications and discusses how the language would support unique identifiers for universals and particulars. There are no real results reported.
The presentation in the paper is very good, although the English is not always idiomatic. The abstract emphasizes the need for referent tracking in EHR applications, although there is no evidence that the authors actually have used Haskell to develop such software. As Haskell will be unfamiliar to many readers, it at least deserves an explanation in the abstract. The introduction to the paper should make it clear that the manuscript is not actually about referent tracking.

→ *Has been addressed*

---------- REVIEW 4 ----------
OVERALL EVALUATION: 1 (weak accept)
REVIEWER'S CONFIDENCE: 3 (medium)
Scientific or technical quality: 4 (very good (upper 1/3))
Novelty or innovation: 3 (innovative)
Presentation: 1 (needs major improvements - give details in the Review section below)
References: 3 (nothing missing but irrelevant references present)
Recommendation for FOIS Best Paper Award: no

---------- Review ----------

This paper presents a notion of ontology-based computing. The basic observation is that within a significant part of the ontology community ontology-based computing is identified with automatic theorem proving. The authors introduce a wider notion of ontology-based computing, which requires the input data, the transformations, and the output to conform to a given ontology. An example is given, which shows how the subclass hierarchy of an ontology may be used as type-constraints in Haskell.
The idea is interesting and the individual pieces of the paper are well-written. However, there is a significant mismatch between the title, the abstract, and the introduction on one hand and the rest of the paper. The title/abstract/introduction suggest that this paper is about a health information system that was tested with 570 000 electronic health records. The introduction discusses further subjects like a top-level ontology of a dynamic system (based on BFO), domain specific constraints on the possible states of a system, and referent tracking. All of these subjects are, as far as I can tell, quite irrelevant to the main part of the paper. No health information system is discussed. Even the examples in section 3 are not really specific to the health domain, they mainly involve notions from BFO and some temperature related entities. -- Because of the disconnect between the introductory part and the main part of the paper, readers likely will get wrong expectations. Further, the explanation and motivation of the real topic of the paper (namely, why we need the notion of ontology-based computing) is too short. (Section 4.1 provides some of the missing motivation, but that's in the discussion section.)

→ *We agree. We therefore made the following changes: (1) We removed the medical records topic from the title, (2) we changed abstract and introduction to make more clear how the various elements – Haskell, referent tracking and electronic healthcare records – fit together in the work we report on.*

Section 2.1: clause (a) of the definition of "pure ontology-based computing" is unclear. What is meant by "specification"?

→ *we explained this*

Section 2.1: the definitions need additional explanation. For example, why is it important that the program is syntactically and semantically well-formed in a way that can be verified *before* the program runs for the first time? Why is run-time verification not sufficient? And what does semantically well-formedness mean?

→ *done*

Section 3: as somebody who is not familiar with Haskell, I found it hard to understand some of the examples. Of course, the paper is not the place to provide an introduction to the Haskell syntax, but readability could be improved if the content of the Haskell code would be paraphrased in English.

→ *there is just no space left to do this ...*

Section 4: The paper does not address the limitations of the approach. Can every OWL axiom be turned into a Haskell type constraint? What about more expressive languages?

→ *good question for a journal version. type level inference may not be enough. manual proofs may be needed to show that existential axioms are satisfied. We added a sentence in the paper.*

Section 4: The paper discusses the relationship to OWL-DL and whether ontology-based computing is possible to implement in imperative programming languages. I was surprised that logical programming was not addressed. Obviously, ontologies have been used within logical programming long before the term "ontology" (in the computational sense) was introduced.

→ *again no space to go into that here ...*