Workshop Notes

The 5th International Workshop on

Acquisition, Representation and Reasoning with Contextualized Knowledge

ARCOE-LogIC 2013

September 15, 2013
La Coruña, Spain

held at the

International Conference on Logic Programming and Nonmonotonic Reasoning

Editors

Michael Fink
Martin Homola
Alessandra Mileo
Ivan Varzinczak

http://www.arcoe.org

Preface

In order to obtain accurate interpretation of knowledge, not just the knowledge itself but also the associated circumstances that constrain its validity and universality need to be considered. Such contextual information influences reasoning when the situation changes and the knowledge has to be reinterpreted, or in cases when multiple pieces of knowledge recorded under different circumstances need to be combined and reasoned with. As such, context bears crucial importance in knowledge representation and in artificial intelligence in general, especially in the course towards real life applications in which dealing with contextual issues is inevitable.

ARCOE-LogIC 2013, the 5th International Workshop on Acquisition, Representation and Reasoning with Contextualized Knowledge, builds on the previous four editions, but as well on the experience from two previous editions of the International Workshop on Logic-based Interpretation of Context (Log-IC Workshop). It brings together researchers from various parts of the knowledge representation field to discuss on the latest developments but also on general directions in the area.

Submissions to ARCOE-LogIC 2013 were peer-reviewed by at least 3 members of the Program Committee. The accepted papers, bound in these Workshop Notes were carefully selected based on their quality, relevance to the workshop topic, and their potential to bring forward interesting ideas to be discussed at the workshop.

Thanks to the invaluable and much appreciated contributions of the authors and the Programme Committee, ARCOE-LogIC 2013 provides participants with an opportunity to position their contributions with respect to one another. Hopefully, this will encourage further cross-pollination and consolidate the constitution of a truly interdisciplinary research-community dedicated to acquisition, representation and reasoning with contextualized knowledge.

(Vienna, Bratislava, Galway, Pretoria – December 2013)

Michael Fink, Martin Homola, Alessandra Mileo, Ivan Varzinczak

Table of Contents

Workshop Organization

ARCOE-13 Co-Chairs

- Michael Fink (Vienna University of Technology, Austria)
- Martin Homola (Comenius University, Slovakia)
- Alessandra Mileo (Digital Enterprise Research Institute, NUI Galway, Ireland)
- Ivan Varzinczak (CSIR Meraka Institute and University of KwaZulu-Natal, South Africa)

ARCOE-13 Steering Committee

- Alan Bundy (University of Edinburgh, UK)
- Thomas Eiter (Vienna University of Technology, Austria)
- Luciano Serafini (Fondazione Bruno Kessler, Italy)

ARCOE-13 Invited Speaker

- Kewen Wang (Griffith University, Australia)

ARCOE-13 Program Committee

- Grigoris Antoniou (FORTH, Greece)
- Christoph Benzmüller (Free University of Berlin, Germany)
- Leopoldo Bertossi (Carleton University, Canada)
- Antonis Bikakis (University College London, UK)
- Patrick Blackburn (Roskilde University, Denmark)
- Loris Bozzato (Fondazione Bruno Kessler, Italy)
- Gerhard Brewka (University of Leipzig, Germany)
- Katarina Britz (CSIR Meraka Institute and University of KwaZulu-Natal, South Africa)
- Alan Bundy (University of Edinburgh, UK)
- Henning Christiansen (University of Roskilde, Denmark)
- Thomas Eiter (Vienna University of Technology, Austria)

- Jérôme Euzenat (INRIA Grenoble Rhône-Alpes, France)
- Michael Fink (Vienna University of Technology, Austria)
- Chiara Ghidini (Fondazione Bruno Kessler, Italy)
- Víctor Gutiérrez Basulto (University of Bremen, Germany)
- Siegfried Handschuh (Digital Enterprise Research Institute, Ireland)
- Martin Homola (Comenius University, Slovakia)
- Szymon Klarman (CSIR Meraka Institute and University of KwaZulu-Natal, South Africa)
- Oliver Kutz (University of Bremen, Germany)
- João Leite (New University of Lisbon, Portugal)
- Wendy MacCaull (St. Francis Xavier University, Canada)
- Vincenzo Malteze (University of Trento, Italy)
- Stefan Mandl (University of Augsburg, Germany)
- Tommie Meyer (CSIR Meraka Institute and University of KwaZulu-Natal, South Africa)
- Alessandra Mileo (Digital Enterprise Research Institute, NUI Galway, Ireland)
- Philip Obermeier (University of Postdam, Germany)
- Valeria de Paiva (University of Birmingham, UK)
- Rafael Peñaloza (TU Dresden, Germany)
- Enrico Pontelli (New Mexico State University, USA)
- Chiaki Sakama (Wakayama University, Japan)
- Torsten Schaub (University of Potsdam, Germany)
- Luciano Serafini (Fondazione Bruno Kessler, Italy)
- Paolo Torroni (University of Bologna, Italy)
- Ivan Varzinczak (CSIR Meraka Institute and University of KwaZulu-Natal, South Africa)
- Kewen Wang (Griffith University, Australia)
- Renata Wassermann (University of São Paulo, Brazil)
- Antoine Zimmermann (École Nationale Supérieure des Mines de Saint-Étienne, France)

Ontology Changes in DL-Lite

Kewen Wang
Griffith University, Brisbane, Australia

Ontologies are not static but may evolve over time. A user may wish to revise/update an existing ontology or she may want to extract a module from a given ontology. It is a challenge to efficiently and effectively manage such changes for ontologies. Also, such changes are insufficiently supported by existing ontology editing and reasoning tools. In this talk, we will present some progress on two ontology change operations, namely, forgetting and revision. We will first discuss major definitions of forgetting for DLs, major properties and algorithms. Then we will introduce a novel syntactic characterization of the semantics for DL-Lite (called features) and then using this new notion, explain how to handle ontology revision in DL-Lite. Finally, some possible issues for future research will be discussed.

Automated Consistency Checking of Expressive Ontologies — Beware of the Wrong Interpretation of Success!

Christoph Benzmüller* and Marco Ziener

Dahlem Center for Intelligent Systems, Freie Universität Berlin, Germany
c.benzmueller@fu-berlin.de, m.ziener@fu-berlin.de

Abstract. There have been attempts to (partially) translate expressive ontologies such as SUMO or Cyc to first-order logic and to use first-order automated theorem provers for detecting errors and inconsistencies. Claims have been made that these translation results are now ready for use within practical applications.
This paper adopts a critical attitude: The fact that a translation of an expressive ontology into a target logic (be it first-order or higher-order or modal, etc.) exists for which satisfiability can be shown does not imply that the approach is ready for application. What might still be missing is a guarantee of the faithfulness of the translation.
The issue is illustrated in connection with a (knowingly) unfaithful translation of SUMO into classical higher-order logic. The focus is on a small, provably satisfiable subset of this translated SUMO ontology. By adding some intuitively compatible and sound ABox axioms the unfaithfulness of the translation can easily be revealed with automated theorem provers. Without the ABox content, however, the problem remains undetected.
We thus argue for the extensive integration of ABox information into automated consistency checking for expressive ontologies, in particular, when logic translations are involved and when faithfulness has not or cannot be formally ensured.

1 Introduction

Expressive formal ontologies such as SUMO [14, 16] or Cyc [11] have been developed to support a wide range of practical applications. Since these might also include safety critical applications detecting errors and inconsistencies in the used ontologies is of utmost importance. For this purpose the use of automated theorem provers and model finders has been proposed.

Respective research efforts have led to multiple ontology revisions and they resulted in revised (partial) first-order views on them. Examples include Adimen-SUMO [1], TPTP-SUMO [15, 17], and first-orderized Cyc [18].

However, such first-orderized ontologies typically adopt very pragmatic attitudes towards some challenge aspects in the source ontologies. These include

* Supported by the German Research Foundation DFG (grant BE2501/9-1).

contextual statements in the tradition of McCarthy's work [12, 13], in which formulas, say `F`, are wrapped into context descriptions, e.g. `(in-context C F)`. These contexts can be of various kinds, including epistemic (e.g. an agent `A` knows or believes `F`: `(knows A F)`), doxastic (e.g. `(believes A F)`), or temporal (e.g. `F` holds during time `T`: `(holdsDuring T F)`).

Pragmatic solutions for handling these aspects are as follows: Adimen-SUMO simply excludes all affected axioms, that is, Adimen-SUMO is highly partial and it will hence answer queries incorrectly for which such contextualized axioms do play a role. TPTP-SUMO and first-orderized Cyc apply ad hoc translations to affected axioms (cf. [15, 18]). The faithfulness of these translations, however, has not been formally demonstrated. In fact, related problems in the TPTP-SUMO ontology have already been pointed out in the past [4].

Faithfulness of a translation α from source logic S to target logic T refers to the following property (Γ is a set of S-formulas, Φ an S-formula, $\models^S$ and $\models^T$ are the semantical entailment relations of S and T):

$$\Gamma \models^S \Phi \quad \text{iff} \quad \alpha(\Gamma) \models^T \alpha(\Phi)$$

Ideally the faithfulness of a logic translation should be proven first and then formally verified resp. formally ensured for the implementation. There is important and promising work in this direction, including the OntoIOp[1] initiative, LoLa [10] and the LATIN[2] project.

However, a formal assurance of the faithfulness (of the implementation) of a logic translation is often non-trivial to achieve in practice, and Adimen-SUMO or TPTP-SUMO are examples where this applies.

Another example is studied in this paper, where classical higher-order logic (HOL) [2, 8] is employed as target logic for expressive ontologies. In [4] two different mappings of SUMO into HOL (respectively, into the THF0 syntax [22] for HOL) have bee given. The first mapping, called THF0-SUMO-I in the remainder, translates contextualized formulas such as `(knows A F)` into HOL terms ($knows_{i\to o\to o} A_i F_o$) were type i denotes the set of all individuals and type o stands for Booleans. Similar to the solution in TPTP-SUMO this mapping implicitly assumes extensional semantics for the contextualized, embedded formulas, and this is where faithfulness breaks. The second mapping in [4] appropriately identifies epistemic, doxastic and temporal context modifiers such as `knows`, `believes` and `holdsDuring` with respective modal logic connectives. That is, the second mapping is actually a mapping into quantified modal logic, which in turn is modeled in [4] as a fragment of HOL (for more details on the embedding of quantified modal logic in HOL we refer to [3]). The intention of second mapping thus has been to guarantee faithfulness. However, no formal proof has been given yet and a respective verification of the implementation appears very difficult to achieve.

[1] http://ontoiop.org
[2] http://trac.omdoc.org/LATIN/

The contributions of this paper are as follows:

First, we report on recent experiments with the THF0-SUMO-I translation. Extending the work in [4], this translation has meanwhile been applied to the entire SUMO ontology, including the mid-level ontology MILO and including all SUMO domain ontologies. Several errors in SUMO have been detected this way which remained undetected so far. However, the unfaithfulness of the THF0-SUMO-I translation could unfortunately not be revealed in these experiments.

As a second contribution this paper therefore investigates the following question: Given an expressive ontology such as SUMO and given some translation into a target formalism for which automated theorem provers and model finders are available (be it first-order logic, higher-order logic or modal logic), how much confidence should we have in it — in particular, with respect to a proper treatment of contextual statements — when the available reasoners for this target logic report satisfiability or, at least, when they cannot find any further inconsistencies in elaborate and comprehensive experiments? We use a simple example to illustrate that actually very little can be said based on such experiments. In fact, it might be misleading and even dangerous if test results are interpreted wrongly.

As a possible solution we propose the extensive integration of ABox information into automated consistency checking for expressive ontologies, in particular, when logic translations are involved and when faithfulness has not or cannot be formally ensured.

The remainder of the paper is structured as follows: In Section 2 we outline THF0-SUMO-I, our (knowingly incorrect) translation of SUMO into HOL. Section 3 presents some of the errors in SUMO that we have detected in our experiments with THF0-SUMO-I. These experiments were facilitated by our framework for elaborate consistency checking of expressive ontologies with HOL provers and model finders. This framework is outlined in Section 4. The main contribution of this paper is then presented in Section 5, where the focus is on a small, provably satisfiable subset of the THF0-SUMO-I ontology. By adding some intuitively compatible and sound ABox axioms, the unfaithfulness of the THF0-SUMO-I translation can now be revealed by HOL reasoners. Without such additional ABox axioms, however, the problem remains undetected. The paper is concluded in Section 6.

2 A naive translation of SUMO to THF0

The first mapping in [4], called THF0-SUMO-I in this paper, was deliberately chosen to simply map embedded formulas in SUMO axioms to HOL terms of type o (Booleans in THF0) and modal modifiers operating on these embedded formulas to functions on Booleans. For example, with this mapping the SUMO axioms

```
(=> (knows ?AGENT ?FORMULA) (believes ?AGENT ?FORMULA))    (1)
(=> (knows ?AGENT ?FORMULA) (truth ?FORMULA True))         (2)
```

are translated into the following THF0 representation:

```
% Type declarations
thf(truth,type,(truth: ($o>$o>$o))).
thf(believes,type,(believes: ($i>$o>$o))).
thf(knows,type,(knows: ($i>$o>$o))).

% Axioms
thf(ax1126,axiom,((! [FORMULA: $o,AGENT: $i]:                    (3)
 ((knows @ AGENT @ FORMULA) => (believes @ AGENT @ FORMULA))))).
thf(ax3303,axiom,((! [FORMULA: $o,AGENT: $i]:                    (4)
 ((knows @ AGENT @ FORMULA) => (truth @ FORMULA @ $true))))).
```

The THF0 formulas (3) and (4) correspond to the SUMO axioms (1) and (2), respectively. `$o` represents the type of Booleans, `$i` the type of individuals, `!` stands for universal quantification; the quantified variables together with their types are given in `[.]`-brackets. `=>` is implication and `@` is the explicit application operator as required in THF0 syntax. The types of the predicates `truth`, `believes`, and `knows` are provided in the type declaration section; `believes` and `knows` take an individual and a formula (i.e., a Boolean value `$true` or `$false`; remember that HOL is extensional) as argument and return a Boolean value, `truth` acts like a binary connective. More information on THF0 is provided in [22] and the THF0-SUMO-I translation is discussed in more detail in [4].

Extending the achievements in [4], we have applied the THF0-SUMO-I translation to produce a translation of the entire SUMO ontology, including the mid-level ontology MILO and all domains ontologies; this THF0-SUMO-I ontology can be download from:

http://www.christoph-benzmueller.de/papers/SUMOMILODOMAINS.thf

For the understanding of this paper the very details of THF0-SUMO-I are not necessarily required. Just remember that modal operators are wrongly mapped to extensional functions on type o (Booleans). Moreover, the instantiation (copying) of axioms for different types was required. The reason is that SUMO contains some axioms which cannot be assigned simple types otherwise. Consider, for example, the SUMO axiom `(instance instance BinaryPredicate)`, which is translated into `thf(ax,axiom,((instance_IIiioIioI @ instance_IiioI @ lBinaryPredicate_i)))`. Here we have now two differently typed copies of `instance`. Further axioms may now have to be formulated for both of these copies, etc.

3 Detecting errors in SUMO

By employing THF0-SUMO-I numerous errors in the SUMO ontology have been detected which remained undetected so far, also by the experiments conducted

with Adimen-SUMO or TPTP-SUMO. Most often these errors were related to unused resp. undeclared variable names; cf. `?HORSE` and `?MOTION` in the following axiom from SUMO domain ontology *Sports.kif*:

```
(=>
  (instance ?HORSEBACK Equitation)
  (exists (?HORSE)
    (and
      (instance ?MOTION HorseRiding)
      (subProcess ?MOTION ?HORSEBACK))))
```

Typically, axioms were concerned which contain embedded formulas, such as the following one in which the last occurrence of *?RES1* is not bound. Those axioms are either not translated by the existing alternative approaches (this is the case for Adimen-SUMO) or the embedded formulas are translated simply as strings (as selectively done in TPTP-SUMO). In both cases typos and semantical errors in the embedded formulas can therefore not be detected by these approaches.

```
(=>
  (and
    (instance ?AGENT Agent)
    (potentialCustomer ?CUST ?AGENT)
    (modalAttribute
      (and
        (instance ?R Reserving)
        (destination ?R ?AGENT)) Necessity)
    (conditionalProbability
      (exists (?RES1)
        (and
          (instance ?RES1 Reservation)
          (reservingEntity ?CUST ?RES1)
          (fulfillingEntity ?AGENT ?RES1)))
      (customer ?CUST ?AGENT) ?NUM1)
    (conditionalProbability
      (not
        (exists (?RES2)
          (and
            (instance ?RES1 Reservation)
            (reservingEntity ?CUST ?RES2)
            (fulfillingEntity ?AGENT ?RES2))))
      (customer ?CUST ?AGENT) ?NUM2))
  (lessThan ?NUM2 ?NUM1))xf
```

Hence, for the automated detection of such errors in expressive ontologies with automated (higher-order or first-order) theorem provers a correct and intuitively appropriate treatment of all concepts in the mapping of the ontology is not necessarily required. Even very flawed mappings could still be useful to some extent.

"The procedure we have used for finding inconsistencies in the ontology can be sketched as follows:

1. *An automated procedure translates a large part of SUMO (and discharges the remaining axioms).*
2. *Then, the whole resulting formula is given (as input) to a theorem prover for automatically finding an inconsistency (that is, without providing a goal).*
3. *When a refutation is found, the theorem prover provides a description of the proof, from which we select the collection of axioms involved in that refutation.*
4. *With the help of theorem provers (for example, for finding minimal inconsistent subcollections of axioms), we identify the source of the inconsistency and repair it.*
5. *Once we repair the problem, the process is repeated from the beginning."*

Fig. 1. Automated error and inconsistency checking for expressive ontologies as has been applied for Adimen-SUMO; the text is copied from [1].

4 A framework for error and inconsistency detection

To enable experiments with THF0-SUMO-I (and with further mappings to HOL) we have developed and deployed a framework for the mechanized inconsistency detection in expressive ontologies with HOL theorem provers and model finders.

For this we have followed the same overall procedure as has e.g. been applied for Adimen-SUMO. This procedure, replicated from [1], is described in Fig. 1.

A main challenge in our work has been to appropriately cluster the THF0-SUMO-I ontology into smaller subsets which can still be processed by the available THF0 theorem provers and model finders.

A naive approach would be to generate all subsets of the powerset of axioms in THF0-SUMO-I and to apply the THF0 reasoners to them; thereby one could start with the unit sets and work towards larger and larger supersets.

Due to the size of the THF0-SUMO-I ontology such an exhaustive approach is practically infeasible. Therefore we have decided to cluster the THF0-SUMO-I ontology first into smaller subontologies. As in related work on ontology clustering [19, 23, 24] the idea is to identify subsets of axioms which are semantically independent to a large degree. Recent progress in ontology clustering is presented in [9]; there partial consistency proofs may even be combined into global consistency proofs.

Our work has focused on the exploitation of type information for ontology clustering. Remember that axioms are copied for differently typed instances of the same SUMO symbols in THF0-SUMO-I. Hence, we conjectured that this type information might serve as an effective differentiation criterion.

The idea is formalized by viewing the ontology as a structure $\mathbf{O}(\mathbf{T}, \mathbf{A}, \mathbf{R})$ where

– **T** is the set of all types used in the ontology (as mentioned before, the THF0-SUMO-I translation actually generates many different types and it often introduces axiom duplicates for different types);
– **A** is the set of all axioms used in the ontology;
– **R** is the set of user defined relations on a combination of the two previous sets or restricted to one of them.

Given a relation $r \in \mathbf{R}$, our framework computes the induced graph. For inconsistency detection we consider all possible subgraphs of this graph. In order to generate a valid THF0 input file an iteration over all the axioms and types of the subgraph is needed together with some removal of duplicates. By default relation r is chosen such that $r(a, a')$ holds (for $a, a' \in \mathbf{A}$) if and only if there exists $t \in \mathbf{T}$ which is used in both a and a'. Respective properties of types and axioms, on which relations r can be defined, are meta data and they are extracted first. This meta data forms an implicit adjacency list which is stored in a database.

The framework is implemented in the Python programming language. It supports the extraction of meta data from a THF0 ontology and it integrates the HOL automated theorem provers Nitpick [6], LEO-II [5] and Satallax [7]. It is capable of correctly interpreting their results by using the SZS ontology [21]. Moreover, it provides optimizations like caching and generator expressions for increasing speed and for limiting memory consumption on subset generation. Furthermore, the framework stores the ontology and the extracted meta data in a database for persistence.

The framework supports three different execution modes: (A) It can be deployed locally allowing the user to check and develop algorithms on his local machine, which he can then deploy to the cluster. (B) It is also possible to use the framework on the TORQUE [20] cluster or (C) on a custom setup by using Celery[3]. When running locally or when using Celery each subgraph is translated into a single job to be executed. Satisfiable subgraphs as well as generated finite models are dismissed in order to save storage space.

On TORQUE the framework uses templates for job generation as well as a packaging mechanism to reduce the overhead caused by distributing the computation onto different machines. Additionally, the framework employs a two step approach when running on TORQUE. If a job consisting of many THF0 problems runs into a timeout, then this timeout will be detected and the undetermined subsets will be written into the database for further examination. If the job is successful the results will be parsed and negative results (pointing to errors and inconsistencies) will be stored in the database. The database is polled on a regular basis for jobs with timeouts and the framework spawns a single instance of this job with a different prover in order to derive a result.

The error detection framework has been extensively applied over several weeks to the THF0-SUMO-I ontology. A large number of proof problems (to be precise, 3.047.128 at the time of writing) has been generated and passed to the above THF0 reasoners. Thereby, the errors as mentioned in Section 3 have been detected.

[3] http://www.celeryproject.org/

Nevertheless, the outcome of the experiments was disappointing to us: Our expectation was that the (known) unfaithfulness of the THF0-SUMO-I translation could eventually be experimentally revealed. Since this was not the case we did conduct some further experiments in order to find out why this was not the case. These additional experiments are summarized and discussed in the next section.

5 Unfaithfulness may be hard to detect

Can the unfaithfulness of a translation of an expressive ontology into a target logic be automatically detected with automated reasoners for this target logic in experiments? Or, alternatively, what does it mean if extensive experiments in this sense stabilize without revealing any further errors or inconsistencies? Can we then trust the translation? What does it imply if we can even formally prove the consistency of the translation result?

Statements as we find them, for example, on the Adimen-SUMO website (cf. http://adimen.si.ehu.es/web/AdimenSUMO) indicate that stabilizing experiments (or satisfiability results) in this sense are in fact applied as a criterion for the trustfulness of a translation:

As a result of this process [meant is the process as described in Fig. 1]*, we obtain a validated and consistent first-order version of the ontology to be used by first-order theorem provers.*

As outlined in Sections 3 and 4 , we have applied a related process to THF0-SUMO-I. Hence, had we not already been aware of the unfaithfulness of the translation, the temptation would have been huge to make a similar statement for the THF0-SUMO-I ontology.

Sure, THF0 theorem provers and model finders are still comparably weak, and one might argue that this is the reason for not detecting the unfaithfulness of the translation (note that showing the satisfiability of the entire THF0-SUMO-I ontology is currently still way beyond the computational capabilities of the higher-order model finders Nitpick, Satallax and LEO-II). However, as we will illustrate next, this is not the crucial point. Even if satisfiability of the entire THF0-SUMO-I ontology could be formally shown, this would still not imply that THF0-SUMO-I can now be safely employed in applications.

For demonstrating this we consider a SUMO toy ontology consisting of exactly the two SUMO axioms (1) and (2) from page 3; this ontology is called **E1** in the remainder. The THF0-translation of **E1** consists of the axioms (3) and (4) as given on page 4 (together with the necessary type declarations). This subontology of THF0-SUMO-I is called **E1***.

When being applied to **E1*** the THF0 reasoners LEO-II, Satallax and Nitpick report satisfiability (LEO-II and Satallax do this within a few milliseconds on standard PCs, and Nitpick needs about 4 seconds).

Next, we add some ABox information to our toy ontology. Thus, we extend **E1** and **E1*** into **E2** and **E2***, respectively. The added facts express that it is not the truth that Bruce is the father of Ben and of Bill (aboxAx1), that Peter

```
% Type Declarations
thf(bill,type,( bill: $i )). thf(ben,type,( ben: $i )).
thf(bruce,type,( bruce: $i )). thf(peter,type,( peter: $i )).
thf(father,type,( father: $i > $i > $o )).
thf(truth,type,( truth: $o > $o > $o )).
thf(believes,type,( believes: $i > $o > $o )).
thf(knows,type,( knows: $i > $o > $o )).

% Axioms
thf(ax1126,axiom,(
    ! [FORMULA: $o,AGENT: $i] :
      ( ( knows @ AGENT @ FORMULA ) => ( believes @ AGENT @ FORMULA ) ) )).

thf(ax3303,axiom,(
    ! [FORMULA: $o,AGENT: $i] :
      ( ( knows @ AGENT @ FORMULA ) => ( truth @ FORMULA @ $true ) ) )).

thf(aboxAx0,axiom,
    ( ( ben != bill ) & ( bruce != ben ) & ( bruce != bill )
    & ( peter != ben ) & ( peter != bill ) & ( peter != bruce ) )).

thf(aboxAx1,axiom,(
    ~ ( truth
      @ ( ( father @ bruce @ ben ) & ( father @ bruce @ bill ) )
      @ $true ) )).

thf(aboxAx2,axiom,
    ( knows @ peter @ ( father @ bruce @ ben ) )).

thf(aboxAx3,axiom,
    ( believes @ peter @ ( father @ bruce @ bill ) )).
```

Fig. 2. The satisfiable toy ontology **E2***. Adding `~(believes peter ~(father bruce bill))` results in an unsatisfiable set of THF0 axioms. This is clearly counter-intuitive.

knows that Bruce is the father of Ben (aboxAx2), and that Peter believes that Bruce is the father of Bill (aboxAx0); in SUMO notation these axioms read as:

```
~(truth ((father bruce ben) & (father bruce bill)) True)      (aboxAx1)
(knows peter (father bruce ben))                              (aboxAx2)
(believes peter (father bruce bill))                          (aboxAx3)
```

Axiom (aboxAx0) additionally states that the constant symbols `peter`, `bruce`, `ben` and `bill` denote mutually different individuals. The detailed content of **E2*** is presented in Fig. 2.

Toy ontology **E2*** is still satisfiable, which is quickly confirmed by LEO-II, Satallax and Nitpick. This is also what we intuitively expect: Peter's believes may well differ from the true facts about the world.

Next, we slightly reformulate the content of axiom (aboxAx3) and state that Peter does not believe that Bruce isn't the father of Bill:

```
~(believes peter ~(father bruce bill))          (aboxAx4)
```

The translation of this SUMO axiom to THF0 is (aboxAx4*):

```
thf(aboxAx4,axiom,
    ( ~ ( believes @ peter @ ( ~ ( father @ bruce @ bill ) ) ) )).
```

According to our intuition and according to the intended semantics of SUMO, axiom (aboxAx4*) should be consistent with **E2***. However, this is not the case: now LEO-II, Satallax and Nitpick do report unsatisfiability. They also report unsatisfiability when (aboxAx3) is being removed from the example. Moreover, LEO-II and Satallax show that the query

```
believes peter ~(father bruce bill))          (query)
```

is implied by **E2***.

The latter results do obviously contradict our intuition on knowledge and belief, and they do also contradict the semantics of these modalities as intended in SUMO. The problem clearly is the unfaithfulness (more precisely, the wrongly assumed fully extensional semantics) of the THF0-SUMO-I translation. However, as our example nicely illustrates: it cannot be expected that the unfaithfulness of such a logic translation can be revealed without adding further ABox information and/or queries.

6 Conclusion

The unfaithfulness of a translation of an expressive ontology into a target logic might not be detectable by automated reasoners for this target logic in experiments. In our example case that was caused by an extensional treatment of intensional notions, and this mismatch could not be revealed by solely studying the translated ontology. However, it has been illustrated that the addition of further ABox information and of user queries may eventually help.

Instead of conducting theorem prover supported (in-)consistency checking on the translations of the terminological content of expressive ontologies only, we instead argue for the additional exploitation of annotated test corpora within experiments. These test corpora should ideally contain substantial, well selected ABox information and related user queries. As we have documented, the latter approach can eventually reveal inconsistencies and unfaithful translations which will remain undetected otherwise.

Allowedly, the faithfulness of a translation should ideally be formally proved before the translation is implemented and deployed, and in this respect the

work as pursuit in the OntoIOp, LoLa and LATIN projects is promising and important. However, such an approach may not be always easily feasible. In fact, neither for Adimen-SUMO nor for TPTP-SUMO such proofs have been provided. Moreover, even if a respective pen and paper proof is given, typically little can be implied from it for the faithfulness of the actual implementation.

Future work includes studying the following question: How can the systematic creation of respective ABox- and user query-enriched test corpora for expressive ontologies be achieved in practice? Can such a process eventually be (partially) automated?

Acknowledgments: We thank Jasmin Blanchette, Chad Brown, Adam Pease and Geoff Sutcliffe for the provers and the tools which we have employed in our work. Moreover, we thank the anonymous reviewers of this paper for their valuable feedback.

References

1. Javier Álvez, Paqui Lucio, and German Rigau. Adimen-sumo: Reengineering an ontology for first-order reasoning. *Int. J. Semantic Web Inf. Syst.*, 8(4):80–116, 2012.
2. Peter Andrews. Churchś type theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2009 edition, 2009.
3. Christoph Benzmüller and Lawrence Paulson. Quantified multimodal logics in simple type theory. *Logica Universalis (Special Issue on Multimodal Logics)*, 7(1):7–20, 2013.
4. Christoph Benzmüller and Adam Pease. Higher-order aspects and context in SUMO. *Journal of Web Semantics (Special Issue on Reasoning with context in the Semantic Web)*, 12-13:104–117, 2012.
5. Christoph Benzmüller, Frank Theiss, Lawrence Paulson, and Arnaud Fietzke. LEO-II - a cooperative automatic theorem prover for higher-order logic (system description). In *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *LNCS*, pages 162–170. Springer, 2008.
6. Jasmin Christian Blanchette and Tobias Nipkow. Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In *Theorem Proving, First International Conference, ITP 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6172 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2010.
7. Chad E. Brown. Satallax: An automatic higher-order prover. In *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*, pages 111–117. Springer, 2012.
8. Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.
9. Oliver Kutz and Till Mossakowski. A modular consistency proof for dolce. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011.

10. Christoph Lange, Till Mossakowski, and Oliver Kutz. Lola: A modular ontology of logics, languages, and translations. In *Proceedings of the 6th International Workshop on Modular Ontologies, Graz, Austria, July 24, 2012*, volume 875 of *CEUR Workshop Proceedings*, 2012.
11. Cynthia Matuszek, John Cabral, Michael Witbrock, and John Deoliveira. An Introduction to the Syntax and Content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 44–49, 2006.
12. John McCarthy. Generality in artificial intelligence. *Communications of the ACM*, 30(12):1030–1035, 1987.
13. John McCarthy. Notes on formalizing context. In *Proceedings of IJCAI'93*, pages 555–562, 1993.
14. Ian Niles and Adam Pease. Towards A Standard Upper Ontology. In C. Welty and B. Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems*, pages 2–9, 2001.
15. A. Pease and G. Sutcliffe. First Order Reasoning on a Large Ontology. In *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories*, number 257 in CEUR Workshop Proceedings, pages 59–69, 2007.
16. Adam Pease. *Ontology — A Practical Guide*. Articulate Software Press, 2011.
17. Adam Pease, Geoff Sutcliffe, Nick Siegel, and Steven Trac. Large theory reasoning with SUMO at CASC. *AI Communications*, 23(2-3):137–144, 2010.
18. Deepak Ramachandran, Pace Reagan, and Keith Goolsbey. First-orderized ResearchCyc: Expressivity and efficiency in a common-sense ontology. In *Papers from the AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications, Pittsburgh, Pennsylvania, USA, 2005*. Technical Report WS-05-01 published by The AAAI Press, Menlo Park, California, July 2005.
19. Domenico Rosaci. An ontology-based two-level clustering for supporting e-commerce agents' activities. In *Proceedings of the 6th international conference on E-Commerce and Web Technologies*, EC-Web'05, pages 31–40, Berlin, Heidelberg, 2005. Springer-Verlag.
20. Garrick Staples. TORQUE - TORQUE resource manager. In *Proceedings of the ACM/IEEE SC2006 Conference on High Performance Networking and Computing, November 11-17, 2006, Tampa, FL, USA*, page 8. ACM Press, 2006.
21. Geoff Sutcliffe. The SZS Ontologies for Automated Reasoning Software. In *Proceedings of the LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants, and The 7th International Workshop on the Implementation of Logics*, number 418 in CEUR Workshop Proceedings, pages 38–49, 2008.
22. Geoff Sutcliffe and Christoph Benzmüller. Automated reasoning in higher-order logic using the TPTP THF infrastructure. *Journal of Formalized Reasoning*, 3(1):1–27, 2010.
23. Donato Malerba Valentina A.M. Tamma, Pepijn R.S. Visser and Dean M. Jones. Computer assisted ontology clustering for knowledge sharing. In *Proceedings of ECML2000/ML net Workshop on Machine Learning in the New Information Age*, page 7583, 2000.
24. Pepijn Visser and Valentina Tamma. An experience with ontology clustering for information integration. In *Proceedings of the IJCAI-99 Workshop on Intelligent Information Integration*, volume 23 of *CEUR Workshop Proceedings*, 1999.

Defeasibility in contextual reasoning with CKR

Loris Bozzato[1], Thomas Eiter[2], and Luciano Serafini[1]

[1] Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy
[2] Institut für Informationssysteme, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria

{bozzato,serafini}@fbk.eu, eiter@kr.tuwien.ac.at

Abstract. Recently, representation of context dependent knowledge in the Semantic Web has been recognized as a relevant issue and a number of logic based solutions have been proposed in this regard: among them, in our previous works we presented the Contextualized Knowledge Repository (CKR) framework.
A CKR knowledge base has a two layered structure, modelled by a global context and a set of local contexts: the global context not only contains the metaknowledge defining the properties of local contexts, but also holds the global (context independent) object knowledge that is shared by all of the local contexts. In many practical cases, however, it is desirable to leave the possibility to "override" the global object knowledge at the local level, by recognizing the axioms that can allow exceptional instances in the local contexts. This clearly requires to add a notion of non monotonicity across the global and the local parts of a CKR.
In this paper we present an extension to the semantics of CKR to introduce such notion of defeasible axioms. By extending a previously proposed datalog translation, we obtain a representation for CKR as a datalog program with negation under answer set semantics. This representation can be exploited as the basis for implementation of query answering for the proposed extension of CKR.

1 Introduction

Representation of context dependent knowledge in the Semantic Web has been recently recognized as a relevant issue that lead to a number of logic based proposals, e.g. [7–9, 13–15, 12]; in particular, the Contextualized Knowledge Repository (CKR) framework [12, 3, 2], with its latest formulation in [4], has been developed at the FBK in Trento.

A CKR knowledge base has a two layered structure: basically, it consists of a global context and a set of local contexts. The global context contains metaknowledge defining the properties of local contexts, but also the global (context independent) object knowledge that is shared by all of the local contexts. Local contexts, on the other hand, contain knowledge that holds under specific circumstances (e.g. an event) and thus represent different independent views of the domain. The global object knowledge is propagated from the global to the local contexts and used as a common part of the system knowledge. In many practical cases, however, it is desirable to leave the possibility to "override" the global object knowledge at the local level, by recognizing those axioms that can allow exceptional instances in their local instantiations.

For example, in the scenario of an event recommendation system, we might want to assert at the global level that *"by default, all of the cheap events are interesting"*,

but then override this inclusion for particular kind of events in the context of a participant. We might also want to express defeasibility on the propagation of information: for instance, in a CKR representing an organization, we might want to express that *"by default, all the employees working the previous year also work in the current year"* and override the axiom in the context of a specific year for employees that finished their working contract at that time. In other words, we want to specify that certain axioms at the global level are *defeasible*, thus they can allow exceptional instances in local contexts, while holding in the general case: this clearly requires to add a notion of non-monotonicity across the global and the local parts of a CKR.

In this work, we present an extension to the CKR semantics of [4] to support such defeasibility for global object knowledge. As we desire to enrich previous work and to have a datalog representation of the extended CKR semantics that extends one for the CKR semantics in [4], we introduce defeasible axioms guided by the approach of inheritance logic programs in [5]. There the idea is that special rules recognize exceptional facts at the local level and others propagate global facts only if they are not proved to be overridden at the local level, which happens if the opposite is derived; in the same vein, we consider instances of axioms that might be overridden at the local level. The semantics for CKR we define is (as desired) representable by a datalog program with negation under answer set semantics; furthermore, a respective translation can be used as the basis to implement query answering over defeasible CKR knowledge bases.

We can thus summarize as follows the contributions of our work:

– After a brief introduction of preliminary definitions in Section 2, we present in Section 3 syntax and semantics of an extension of CKR with defeasible axioms in the global context. Notably, this allow us to introduce for the first time a notion of non-monotonicity across contexts in our contextual framework.
– We then extend in Section 4 the datalog translation for OWL RL based CKR from [4] with rules for the translation of defeasible axioms and for considering local exceptions in the propagation of such knowledge. We express non-monotonicity using answer set semantics, such that instance checking over an CKR with defeasible axioms reduces to cautious inference from all answer set of the translation.

The work reported here is in progress, and an implementation of a prototype reasoner, based on the results of [4] and this paper, is underway.

2 Preliminaries: $\mathcal{SROIQ}$-RL

This work basically builds on the materialization calculus for CKR on OWL RL recently proposed in [4]. The extension of the calculus that we present here is again formulated over the language $\mathcal{SROIQ}$-RL, which represents the restriction of $\mathcal{SROIQ}$ to the OWL RL constructs [11]. The language is obtained by restricting the form of General Concept Inclusion axioms (GCIs) and concept equivalence of $\mathcal{SROIQ}$ to $C \sqsubseteq D$ where C and D are concept expressions, called *left-side concept* and *right-side concept* respectively, and defined by the following grammar:

$$C := A \mid \{a\} \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C_1 \mid \exists R.\{a\} \mid \exists R.\top$$
$$D := A \mid \neg C_1 \mid D_1 \sqcap D_2 \mid \exists R.\{a\} \mid \forall R.D_1 \mid \leqslant nR.C_1 \mid \leqslant nR.\top$$

where A is a concept name, R is role name and $n \in \{0, 1\}$. A *both-side concept* E is a concept expression which is both a left- and right-side concept. TBox axioms can only take the form $C \sqsubseteq D$ or $E \equiv E$. The RBox for $\mathcal{SROIQ}$-RL can contain every role axiom of $\mathcal{SROIQ}$ except $\mathrm{Ref}(R)$. ABox concept assertions can be only stated in the form $D(a)$, where D is a right-side concept.

3 CKR with defeasible axioms

We now introduce CKRs and extend them with primitives to express defeasible axioms. We first present the syntax and then define a model-based semantics for the interpretation of defeasible inheritance from the upper contexts.

A *Contextualized Knowledge Repository (CKR)* is a two layered structure. The upper layer consists of a knowledge base $\mathfrak{G}$, which describes two types of knowledge: (i) the structure and the properties of contexts of the CKR (called *meta-knowledge*), and (ii) the knowledge that is context independent, i.e., that holds in every context (called *global knowledge*). The lower layer is constituted by a set of (local) contexts; each contains (locally valid) facts and can also refer to what holds in other contexts.

Meta-Language. The meta-knowledge of a CKR is expressed by a DL language defined on a meta-vocabulary, containing the elements that define the contextual structure.

Definition 1 (Meta-vocabulary). *A* meta-vocabulary *is a DL vocabulary Γ composed of a set NC_Γ of atomic concepts, a set NR_Γ of atomic roles, and a set NI_Γ of individual constants that are mutually disjoint and contain the following sets of symbols*

1. $\mathbf{N} \subseteq \mathrm{NI}_\Gamma$ *of* context names.
2. $\mathbf{M} \subseteq \mathrm{NI}_\Gamma$ *of* module names.
3. $\mathbf{C} \subseteq \mathrm{NC}_\Gamma$ *of* context classes*, including the class* Ctx.
4. $\mathbf{R} \subseteq \mathrm{NR}_\Gamma$ *of* contextual relations.
5. $\mathbf{A} \subseteq \mathrm{NR}_\Gamma$ *of* contextual attributes.
6. *For every attribute* $\mathsf{A} \in \mathbf{A}$*, a set* $\mathsf{D_A} \subseteq \mathrm{NI}_\Gamma$ *of* attribute values *of* A.

We use the role $\mathsf{mod} \in \mathrm{NR}_\Gamma$ defined on $\mathbf{N} \times \mathbf{M}$ to express associations between contexts and modules. The *meta-language* $\mathcal{L}_\Gamma$ of a CKR is a DL language over Γ with the following syntactic conditions on the application of role restrictions: for every $\bullet \in \{\forall, \exists, \leqslant n, \geqslant n\}$, (i) for a concept $\bullet\mathsf{A.B}$, then B is in the form $\mathsf{B} = \{\mathsf{a}\}$ with $\mathsf{a} \in \mathsf{D_A}$; (ii) for a concept $\bullet\mathsf{mod.B}$, then B is in the form $\mathsf{B} = \{\mathsf{m}\}$ with $\mathsf{m} \in \mathbf{M}$. **Object Language.** The context (in)dependent knowledge of a CKR is expressed via a DL language called *object-language* $\mathcal{L}_\Sigma$ over an object-vocabulary $\Sigma = \mathrm{NC}_\Sigma \uplus \mathrm{NR}_\Sigma \uplus \mathrm{NI}_\Sigma$. Expression in $\mathcal{L}_\Sigma$ will be evaluated locally to each context, i.e., each context can interpret each symbol independently. However, sometimes one wants to constrain the meaning of a symbol in a context with the meaning of a symbol in some other context. For such external references, we extend the object language $\mathcal{L}_\Sigma$ to $\mathcal{L}^e_\Sigma$ with *eval expressions* of the form *eval*(X, C), where X is a concept or role expression of $\mathcal{L}_\Sigma$ and C is a concept expression of $\mathcal{L}_\Gamma$.

Default Axioms. Compared to [4], we extend the types of axioms that can appear in the global object knowledge $\mathfrak{G}$ with defeasible axioms. Given an axiom $\alpha \in \mathcal{L}_\Sigma$, the assertion states that α is a *defeasible axiom* of $\mathfrak{G}$. Intuitively, this statement means that α, at the level of instantiations for individuals, propagates to a local context unless it is contradicted there, and thus an exception to α for individuals is tolerated.

Example 1. A defeasible fact $\mathrm{D}(\mathit{Expensive}(\mathit{concert}))$ might express that a concert is expensive and propagate this to local contexts. At such a context, this might be contradicted by an assertion $\neg\mathit{Expensive}(\mathit{concert})$, which then *overrides* the global assertion. Likewise, such overriding should take place if we have a global axiom $\mathit{Cheap} \sqsubseteq \neg\mathit{Expensive}$ and a local assertion $\mathit{Cheap}(\mathit{concert})$, such that $\neg\mathit{Expensive}(\mathit{concert})$ can be derived at the local context.

Example 2. Beyond facts, from a defeasible GCI axiom $\mathrm{D}(\mathit{Cheap} \sqsubseteq \mathit{Interesting})$ (*"cheap events are interesting"*) and the global assertion $\mathit{Cheap}(\mathit{fbmatch})$ (*"football matches are cheap"*), we may conclude $\mathit{Interesting}(\mathit{fbmatch})$ at a local context; however, an assertion $\neg\mathit{Interesting}(\mathit{fbmatch})$ there would contradict this and should override the instance of the defeasible axiom for $\mathit{fbmatch}$: that is, the fact $\neg\mathit{Cheap} \sqcup \mathit{Interesting}(\mathit{fbmatch})$ may be violated.

The DL language $\mathcal{L}^{\mathrm{D}}_{\Sigma}$ extends $\mathcal{L}_{\Sigma}$ with the set of defeasible axioms in $\mathcal{L}_{\Sigma}$.

CKR Syntax. We are now ready to give our formal definition of Contextualized Knowledge Repository.

Definition 2 (Contextualized Knowledge Repository, CKR). *A* Contextualized Knowledge Repository (CKR) *over a meta-vocabulary Γ and an object vocabulary Σ is a structure $\mathfrak{K} = \langle \mathfrak{G}, \{\mathrm{K}_{\mathsf{m}}\}_{\mathsf{m}\in\mathbf{M}}\rangle$ where:*
– *$\mathfrak{G}$ is a DL knowledge base over $\mathcal{L}_{\Gamma} \cup \mathcal{L}^{\mathrm{D}}_{\Sigma}$, and*
– *K_{m} is a DL knowledge base over $\mathcal{L}^{\mathrm{e}}_{\Sigma}$, for every module name $\mathsf{m} \in \mathbf{M}$.*

In particular, $\mathfrak{K}$ is a $\mathcal{SROIQ}$-RL CKR, if $\mathfrak{G}$ and all K_{m} are knowledge bases over the extended language of $\mathcal{SROIQ}$-RL where eval-expressions can only occur in left-concepts and contain left-concepts or roles. In the sequel, we tacitly focus on such CKR.

We show how the examples in the introduction can be formalized as CKRs.

Example 3. In the first example (inspired by a real application of CKR[3]) we want to define an event recommendation system: we thus represent touristic events and preferences of tourists in order to be able to derive appropriate suggestions. In particular, we want to assert that, in general, all of the Cheap events are $\mathit{Interesting}$; we do so by expressing this as a defeasible axiom in the global context. Furthermore, we propose local markets (market) and football matches ($\mathit{fbmatch}$) as examples of cheap events. On the other hand, we want to reflect that tourists interested in cultural events are not interested in a sportive event like a football match;[4] we express this by negating the interest in $\mathit{fbmatch}$. Thus, our example CKR $\mathfrak{K}_{tour} = \langle \mathfrak{G}, \{\mathrm{K}_{\mathsf{ctourist\_m}}\}\rangle$ has

$$\begin{aligned} \mathfrak{G} : \{\, & \mathrm{D}(\mathit{Cheap} \sqsubseteq \mathit{Interesting}),\ \mathit{Cheap}(\mathit{fbmatch}),\ \mathit{Cheap}(\mathit{market}), \\ & \mathtt{mod}(\mathsf{cultural\_tourist}, \mathsf{ctourist\_m}) \,\}, \\ \mathrm{K}_{\mathsf{ctourist\_m}} : \{\, & \neg\mathit{Interesting}(\mathit{fbmatch}) \,\}. \end{aligned}$$

Note that the negative assertion in the local context represents, as discussed in Example 2, an exception to the defeasible axiom: we want to recognize this "overriding" for the $\mathit{fbmatch}$ instance, but still apply the defeasible inclusion for market.

[3] `http://www.investintrentino.it/News/Trentour-Trentino-platform-for-smart-tourism`

[4] To keep things simple, we omit modeling sportive events by a separate concept.

Example 4. Our next example shows how we can represent a form of defeasible propagation of information across local contexts using *eval* expressions. We want to represent the information about an organization in a CKR, using contexts to represent its situation in different years. We express the rule that every employee working the years before ($WorkingBefore$) also works in the current year ($WorkingNow$) by a defeasible inclusion. In the module associated to 2012, we say that $alice$, bob and $charlie$ were working last year. In the module for 2013, we say (using an *eval* expression) that all of the employees working in 2012 have to be considered in the set of employees working in the past years; moreover, we say that $charlie$ no longer works for the organization. This can be encoded in the CKR $\mathfrak{K}_{org} = \langle \mathfrak{G}, \{\mathrm{K}_{\mathsf{em2012\_m}}, \mathrm{K}_{\mathsf{em2013\_m}}\}\rangle$, where

$$\mathfrak{G} : \left\{ \begin{array}{l} \mathrm{D}(WorkingBefore \sqsubseteq WorkingNow), \\ \mathtt{mod}(\mathsf{employees2012}, \mathsf{em2012\_m}), \mathtt{mod}(\mathsf{employees2013}, \mathsf{em2013\_m}) \end{array} \right\}$$
$$\mathrm{K}_{\mathsf{em2012\_m}} : \{\ WorkingNow(alice),\ WorkingNow(bob),\ WorkingNow(charlie)\ \}$$
$$\mathrm{K}_{\mathsf{em2013\_m}} : \left\{ \begin{array}{l} \mathit{eval}(\ WorkingNow, \{employees2012\}) \sqsubseteq WorkingBefore, \\ \neg\, WorkingNow(charlie) \end{array} \right\}$$

Intuitively, at the local context employees2013, where $WorkingBefore(charlie)$ can be derived, the negative assertion $\neg\, WorkingNow(charlie)$ should override the instance of the inclusion axiom in the global context for $charlie$, as it would lead to the opposite, i.e., $WorkingNow(charlie)$; on the other hand, for $alice$ and bob no overriding should happen and we can derive that they still work for the organization. $\diamond$

Semantics. We now define a model-based semantics of CKRs. The idea is to model exceptions of axiom instances by so called clashing assumptions, which are pairs $\langle \alpha, \mathbf{e} \rangle$ of an axiom and a set of individuals, to the effect that in the evaluation at the local context, the instance of α for $\mathbf{e}$ is disregarded. However, such a clashing assumption must be justified, in the sense that the instance of α for $\mathbf{e}$ must be unsatisfiable at the local context. This is ensured if there are assertions that can be derived which prove this unsatisfiability: we call such assertions clashing sets. Models of a CKR will be then CKR interpretations in [4] extended with clashing assumptions that are all justified.

Definition 3 (CKR interpretation). *A* CKR interpretation *for* $\langle \Gamma, \Sigma \rangle$ *is a structure* $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ *s.t.*

- $\mathcal{M}$ *is a DL interpretation of* $\Gamma \cup \Sigma$ *s.t., for every* $\mathsf{c} \in \mathbf{N}$, $\mathsf{c}^{\mathcal{M}} \in \mathsf{Ctx}^{\mathcal{M}}$ *and, for every* $\mathsf{C} \in \mathbf{C}$, $\mathsf{C}^{\mathcal{M}} \subseteq \mathsf{Ctx}^{\mathcal{M}}$;
- *for every* $x \in \mathsf{Ctx}^{\mathcal{M}}$, $\mathcal{I}(x)$ *is a DL interpretation over* Σ *s.t., for* $a \in \mathrm{NI}_{\Sigma}$ *and any* $y \in \mathsf{Ctx}^{\mathcal{M}}$, $a^{\mathcal{I}(x)} = a^{\mathcal{I}(y)} = a^{\mathcal{M}}$.

The interpretation of ordinary DL expressions on $\mathcal{M}$ and $\mathcal{I}(x)$ in $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ is as usual; *eval* expressions are interpreted as follows: for every $x \in \mathsf{Ctx}^{\mathcal{M}}$,

$$\mathit{eval}(X, \mathsf{C})^{\mathcal{I}(x)} = \bigcup_{\mathsf{e} \in \mathsf{C}^{\mathcal{M}}} X^{\mathcal{I}(\mathsf{e})}$$

According to the previous definition, a CKR interpretation consists of an interpretation for the "upper-layer" (which includes the global knowledge and the meta-knowledge) and an interpretation of the object language for each instance of type context (i.e., for all $x \in \mathsf{Ctx}^{\mathcal{M}}$), providing a semantics of the object-vocabulary in x.

An *instantiation* of an axiom $\alpha \in \mathcal{L}_\Sigma$ with a tuple $\mathbf{e}$ of individuals in NI_Σ, written $\alpha(\mathbf{e})$, is the specialization of α, viewed as its first order translation in an universal sentence $\forall \mathbf{x}.\phi_\alpha(\mathbf{x})$, to $\mathbf{e}$ (i.e., $\phi_\alpha(\mathbf{e})$); accordingly, $\mathbf{e}$ is void for assertions, a single element e for GCIs, and a pair e_1, e_2 of element for role axioms.

A *clashing assumption* is pair $\langle \alpha, \mathbf{e} \rangle$ such that $\alpha(\mathbf{e})$ is an axiom instantiation; intuitively, it represents the assumption that $\alpha(\mathbf{e})$ is not (DL-)satisfiable. A *clashing set* for a $\langle \alpha, \mathbf{e} \rangle$ is a satisfiable set S of ABox assertions such that $S \cup \{\alpha(\mathbf{e})\}$ is unsatisfiable. That is, S is a non-redundant assertional "explanation" for the negation of the instantiation of α with $\mathbf{e}$.

Given a CKR interpretation $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$, we call the structure $\mathfrak{I}_{CAS} = \langle \mathcal{M}, \mathcal{I}, CAS \rangle$ a *CAS-interpretation*, where CAS is a map such that, for every $x \in \Delta^{\mathcal{M}}$, $CAS(x)$ is a set of clashing assumptions for x.

Definition 4 (CAS-model). *Given a CKR $\mathfrak{K} = \langle \mathfrak{G}, \{\mathrm{K}_\mathsf{m}\}_{\mathsf{m} \in \mathbf{M}} \rangle$ and a CAS-interpretation $\mathfrak{I}_{CAS} = \langle \mathcal{M}, \mathcal{I}, CAS \rangle$, we say that $\mathfrak{I}_{CAS}$ is a CAS-model for $\mathfrak{K}$ ($\mathfrak{I}_{CAS} \models \mathfrak{K}$) if*

- *for every $\alpha \in \mathcal{L}_\Sigma \cup \mathcal{L}_\Gamma$ in $\mathfrak{G}$, $\mathcal{M} \models \alpha$;*
- *for every $\mathrm{D}(\alpha) \in \mathfrak{G}$ with $\alpha \in \mathcal{L}_\Sigma$, $\mathcal{M} \models \alpha$;*
- *if $\langle x, y \rangle \in \mathsf{mod}^{\mathcal{M}}$ and $y = \mathsf{m}^{\mathcal{M}}$, then $\mathcal{I}(x) \models \mathrm{K}_\mathsf{m}$;*
- *for every $\alpha \in \mathfrak{G} \cap \mathcal{L}_\Sigma$, for $x \in \mathsf{Ctx}^{\mathcal{M}}$, $\mathcal{I}(x) \models \alpha$, and*
- *for every $\mathrm{D}(\alpha) \in \mathfrak{G}$ with $\alpha \in \mathcal{L}_\Sigma$, $x \in \mathsf{Ctx}^{\mathcal{M}}$, and domain elements $\mathbf{x} \subseteq \Delta^{\mathcal{I}(x)}$, if $\mathbf{x} \neq \mathbf{e}^{\mathcal{M}}$ for every $\langle \alpha, \mathbf{e} \rangle \in CAS(x)$, then $\mathcal{I}(x) \models \alpha(\mathbf{x})$.*

For $\alpha \in \mathcal{L}^{\mathrm{e}}_\Sigma$ and $\mathsf{c} \in \mathbf{N}$, we write $\mathfrak{K} \models_{CAS} \mathsf{c} : \alpha$ if for every CAS-interpretation $\mathfrak{I}_{CAS}$ it holds that $\mathfrak{I}_{CAS} \models \mathfrak{K}$ implies $\mathcal{I}(\mathsf{c}^{\mathcal{M}}) \models \alpha$; for $\alpha \in \mathcal{L}_\Gamma$, we write $\mathfrak{K} \models_{CAS} \alpha$ if for every CAS-interpretation $\mathfrak{I}_{CAS}$ it holds that $\mathfrak{I}_{CAS} \models \mathfrak{K}$ implies $\mathcal{M} \models \alpha$.

We say that a $\mathfrak{I}_{CAS} = \langle \mathcal{M}, \mathcal{I}, CAS \rangle$ model of $\mathfrak{K}$ is *justified* (for $\mathfrak{K}$), if for every $x \in \mathsf{Ctx}^{\mathcal{M}}$ and $\langle \alpha, \mathbf{e} \rangle \in CAS(x)$ there exists a clashing set $S_{x,\langle \alpha, \mathbf{e} \rangle}$ such that for every CAS-model $\mathfrak{I}'_{CAS} = \langle \mathcal{M}', \mathcal{I}', CAS \rangle$ of $\mathfrak{K}$ with $\Delta^{\mathcal{M}} = \Delta^{\mathcal{M}'}$, it holds $\mathcal{I}'(x) \models S_{x,\langle \alpha, \mathbf{e} \rangle}$ for all these $S_{x,\langle \alpha, \mathbf{e} \rangle}$. Informally, justification requires that we have factual evidence that an instantiation of an axiom can not be satisfied, and this evidence is provable. Based on this, we now give the definition of CKR model:

Definition 5 (CKR model). *A CKR interpretation $\mathfrak{I}$ is a* CKR model *of $\mathfrak{K}$ (in symbols, $\mathfrak{I} \models \mathfrak{K}$), if some $\mathfrak{I}_{CAS} = \langle \mathcal{M}, \mathcal{I}, CAS \rangle$ is a CAS-model of $\mathfrak{K}$ that is justified for $\mathfrak{K}$.*

For $\alpha \in \mathcal{L}^{\mathrm{e}}_\Sigma$ and $\mathsf{c} \in \mathbf{N}$, we write $\mathfrak{K} \models \mathsf{c} : \alpha$ if $\mathcal{I}(\mathsf{c}^{\mathcal{M}}) \models \alpha$ for every CKR model $\mathfrak{I}$ of $\mathfrak{K}$; similarly for $\alpha \in \mathcal{L}_\Gamma$, we write $\mathfrak{K} \models \alpha$ if $\mathcal{M} \models \alpha$ for every CKR model $\mathfrak{I}$ of $\mathfrak{K}$.

We can show the following properties:

Proposition 1. *Suppose that $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ and $\mathfrak{I}' = \langle \mathcal{M}', \mathcal{I}' \rangle$ are CKR models of $\mathfrak{K}$ such that $\mathfrak{I}_{CAS} = \langle \mathcal{M}, \mathcal{I}, CAS \rangle$ and $\mathfrak{I}'_{CAS'} = \langle \mathcal{M}', \mathcal{I}', CAS' \rangle$ are justified models for $\mathfrak{K}$. Then $\mathsf{Ctx}^{\mathcal{M}} = \mathsf{Ctx}^{\mathcal{M}'}$ and $CAS'(x) \subseteq CAS(x)$ for every $x \in \mathsf{Ctx}^{\mathcal{M}}$ implies $CAS = CAS'$.*

The previous result shows that, given the justification of the CKR model, there is a notion of minimality on the sets of clashing assumptions related to each context.

Given a CAS-interpretation $\mathfrak{I}_{CAS} = \langle \mathcal{M}, \mathcal{I}, CAS \rangle$, we denote with $\mathfrak{I}^{\mathbf{N}}_{CAS} = \langle \mathcal{M}', \mathcal{I}', CAS' \rangle$ the interpretation in which: (i) $\Delta^{\mathcal{M}'} = \{a^{\mathcal{M}} \mid a \in \mathrm{NI}_\Gamma \cup \mathrm{NI}_\Sigma\}$; (ii) $\mathcal{I}'$ and CAS' are the restrictions of $\mathcal{I}$ and CAS to $\Delta^{\mathcal{M}'}$, respectively.

Proposition 2. *Let $\mathfrak{I}_{CAS}$ be a CAS-model for $\mathfrak{K}$ that is justified for $\mathfrak{K}$. Then, also the CAS-interpretation $\mathfrak{I}^{\mathbf{N}}_{CAS}$ is a CAS-model for $\mathfrak{K}$ and is justified for $\mathfrak{K}$.*

Basically, the result shows that justification for a CKR only depends on the "named contexts" part of the considered CAS-model: intuitively, this allow us to consider such restricted models in the correctness result of the datalog translation presented in the following section.

Example 5. We can now show an example of CKR models satisfying the example CKRs presented in Example 3.

In the case of $\mathfrak{K}_{tour}$, we can consider a model $\mathfrak{I}_{CAS_{tour}} = \langle \mathcal{M}, \mathcal{I}, CAS_{tour} \rangle$ such that $CAS_{tour}(\mathsf{cultural\_tourist}^{\mathcal{M}}) = \{\langle Cheap \sqsubseteq Interesting, \{fbmatch\}\rangle\}$. Note that the interpretation is justified as it is easy to check that $\mathfrak{K}_{tour} \models \mathsf{cultural\_tourist} : \{Cheap(fbmatch), \neg Interesting(fbmatch)\}$, that represents a clashing set for the defeasible axiom. Moreover, for the definition of satisfiability under the assumptions in CAS_{tour}, we obtain that $\mathcal{I}(\mathsf{cultural\_tourist}^{\mathcal{M}}) \models Interesting(market)$.

Similarly, for $\mathfrak{K}_{org}$, we have that the model $\mathfrak{I}_{CAS_{org}} = \langle \mathcal{M}, \mathcal{I}, CAS_{org} \rangle$ with $CAS_{org}(\mathsf{employees2013}^{\mathcal{M}}) = \{\langle WorkingBefore \sqsubseteq WorkingNow, \{charlie\}\rangle\}$ is a CKR model for the example CKR. For the interpretation of *eval* expressions, in every interpretation of $\mathfrak{K}_{tour}$ we have that $\{alice^{\mathcal{I}(x)}, bob^{\mathcal{I}(x)}, charlie^{\mathcal{I}(x)}\} \sqsubseteq WorkingBefore^{\mathcal{I}(x)}$, where $x = \mathsf{employees2013}$. Thus the justification of the model can be easily seen as $\mathfrak{K}_{org} \models \mathsf{employees2013} : S$ for $S = \{WorkingBefore(charlie), \neg WorkingNow(charlie)\}$, which represents a clashing set for the defeasible axiom on $charlie$. On the other hand, for the satisfiability under the assumptions in CAS_{org}, we obtain that $\mathcal{I}(\mathsf{employees}^{\mathcal{M}}) \models WorkingNow(alice)$ and $\mathcal{I}(\mathsf{employees}^{\mathcal{M}}) \models WorkingNow(bob)$. $\diamond$

As clashing assumptions in CAS maps are ground instances of axioms, they refer merely to named individuals. We remark that using standard names for the domain elements, one could permit clashing assumptions for all elements; the results in Propositions 1 and 2 carry over to this setting.

4 CKR translation to general programs

We revise the datalog translation for $\mathcal{SROIQ}$-RL CKR from [4] with rules for the detection of axiom overriding and defeasible propagation of global knowledge. To simplify the presentation of rules, we introduce a normal form for the considered axioms. We say that a CKR $\mathfrak{K} = \langle \mathfrak{G}, \{\mathrm{K}_{\mathsf{m}}\}_{\mathsf{m} \in \mathbf{M}} \rangle$ is in *normal form* if:

- $\mathfrak{G}$ contains axioms in $\mathcal{L}_\Gamma$ of the form of Table 1 or in the form $\mathsf{C} \sqsubseteq \exists \mathsf{mod}.\{\mathsf{m}\}$, $\mathsf{C} \sqsubseteq \exists \mathsf{A}.\{\mathsf{d_A}\}$ for $A, B, C \in \mathbf{C}$, $R, S, T \in \mathbf{R}$, $a, b \in \mathbf{N}$, $\mathsf{m} \in \mathbf{M}$, $\mathsf{A} \in \mathbf{A}$ and $\mathsf{d_A} \in \mathsf{D_A}$.
- $\mathfrak{G}$ and every K_{m} contain axioms in $\mathcal{L}_\Sigma$ of the form of Table 1 and every K_{m} contain axioms in $\mathcal{L}^e_\Sigma$ of the form $eval(A, \mathsf{C}) \sqsubseteq B$, $eval(R, \mathsf{C}) \sqsubseteq T$ for $A, B, C \in \mathrm{NC}$, $a, b \in \mathrm{NI}$, $R, S, T \in \mathrm{NR}$ and $\mathsf{C} \in \mathbf{C}$.
- $\mathfrak{G}$ contains defeasible axioms $\mathrm{D}(\alpha) \in \mathcal{L}^{\mathrm{D}}_\Sigma$ with α of the form of Table 1.

It can be seen that for named interpretations, i.e., of the form $\mathfrak{I}^{\mathbf{N}}_{CAS}$, every CKR can be rewritten into an equivalent one in normal form (using new symbols).

$A(a)$	$R(a,b)$	$\neg A(b)$	$\neg R(a,b)$	$a = b$	$a \neq b$
$A \sqsubseteq B$	$\{a\} \sqsubseteq B$	$A \sqsubseteq \neg B$	$A \sqcap B \sqsubseteq C$		
$\exists R.A \sqsubseteq B$	$A \sqsubseteq \exists R.\{a\}$	$A \sqsubseteq \forall R.B$	$A \sqsubseteq \leqslant 1 R.B$		
$R \sqsubseteq T$	$R \circ S \sqsubseteq T$	$\mathrm{Dis}(R,S)$	$\mathrm{Inv}(R,S)$	$\mathrm{Irr}(R)$	

Table 1. Normal form axioms

In this version of the translation, the definition of program has now to be adapted to the new form of the rules (admitting negative literals) and to the answer set semantics of the resulting logic program.

Syntax. A *signature* is a tuple $\langle \mathbf{C}, \mathbf{P} \rangle$ of a finite set $\mathbf{C}$ of *constants* and a finite set $\mathbf{P}$ of *predicates*. We assume a set $\mathbf{V}$ of *variables*; the elements of $\mathbf{C} \cup \mathbf{V}$ are *terms*. An *atom* is of the form $p(t_1, \ldots, t_n)$ where $p \in \mathbf{P}$ and $t_1, \ldots, t_n$, are terms. A *literal* l is either a positive literal p or a negative literal $\neg p$ with p an atom and $\neg$ the symbol of strong negation. Literals of the form p, $\neg p$ are *complementary*.

A rule r is an expression of the form

$$a \leftarrow b_1, \ldots, b_k, \texttt{not}\ b_{k+1}, \ldots, \texttt{not}\ b_m.$$

where $a, b_1, \ldots, b_m$ are literals and `not` is the negation as failure symbol. We denote with $Head(r)$ the head a of rule r and with $Body^+(r)$ and $Body^-(r)$ the positive $(b_1, \ldots, b_k)$ and negative $(b_{k+1}, \ldots, b_m)$ literals in the body of the rule[5]. A *fact* H is a ground rule with empty body (we then omit $\leftarrow$). A *program* P is a finite set of rules. A *ground substitution* σ for $\langle \mathbf{C}, \mathbf{P} \rangle$ is a function $\sigma : \mathbf{V} \rightarrow \mathbf{C}$; (ground) substitutions on atoms and *ground instances* of atoms are as usual.

Semantics. Given a program P, we define the *universe* U_P of P as the set of all constants occurring in P and the *base* B_P of P as the set of all the ground *literals* constructible from the predicates in P and the constants in U_P. An *interpretation* $I \subseteq B_P$ is a consistent subset of B_P (i.e., not containing complimentary literals). We say that a literal l is *true* in I iff $l \in I$ and *false* otherwise.

Given a rule $r \in ground(P)$, the body of r is true in I if: (1) every literal in $Body^+(r)$ is true w.r.t. I, and (2) every literal in $Body^-(r)$ is false w.r.t. I. A rule r is *satisfied* in I if either the head of r is true in I or the body of r is not true in I.

An interpretation I for P is a model for P ($I \models P$), if every rule in $ground(P)$ is satisfied in I; it is a *minimal model* for P, if no proper subset $I' \subset I$ is a model for P.

Given an interpretation I for P, the (Gelfond-Lifschitz) *reduct* of P w.r.t. I, denoted by $G_I(P)$, is the set of rules obtained from $ground(P)$ by (i) removing every rule r such that $Body^-(r) \nsubseteq I$. (ii) removing the NAF part from the bodies of the remaining rules. Then I is an *answer set* of P, if I is a minimal model of the positive version $pos(G_I(P))$ of $G_I(P)$ (i.e. the positive program obtained by considering each negative literal $\neg p(t_1, \ldots, t_n)$ as a positive one with predicate symbol $\neg p$).

The following property is well-known.

Lemma 1. *If M is an answer set for P, then M is a minimal model of P.*

[5] In the rules, we might write $(\neg)p$ to denote that the rule holds both for the positive and negative literal associated to p. This will be used only as a shortcut to simplify the presentation of rules.

Using this interpretation for our programs, we say that a literal $H \in B_P$ is a *consequence* of P and we write $P \models H$ iff for every answer set M of P we have that $M \models H$.

Translation. We now are ready to present our translation. As in [4], we basically instantiate and adapt the materialization calculus in [10] to meet the structure of CKR.

The translation is composed by the following sets: the *input translations* I_{glob}, I_{loc}, I_{D}, I_{rl}, the *deduction rules* P_{loc}, P_{D}, P_{rl}, and *output translation* O, such that:

- every input translation I and output translation O are partial functions (defined over axioms in normal form) while deduction rules P are sets of datalog rules;
- given an axiom or signature symbol α (and $\mathsf{c} \in \mathbf{N}$), each $I(\alpha)$ (or $I(\alpha, \mathsf{c})$) is either undefined or a set of datalog facts or rules;
- given an axiom α and $\mathsf{c} \in \mathbf{N}$, $O(\alpha, \mathsf{c})$ is either undefined or a single datalog fact;

We extend the definition of input translations to knowledge bases (set of axioms) S with their signature Σ, with $I(S) = \bigcup_{\alpha \in S} I(\alpha) \cup \bigcup_{s \in \Sigma, I(s) \text{ defined}} I(s)$ (similarly $I(S, \mathsf{c}) = \bigcup_{\alpha \in S} I(\alpha, \mathsf{c}) \cup \bigcup_{s \in \Sigma, I(s) \text{ defined}} I(s, \mathsf{c})$).

We briefly present the form of the different sets of translation and deduction rules involved in the translation process: the tables containing the complete set of rules can be found in the Appendix.

The set of rules in $I_{rl}(S, c)$ define the rules to translate $\mathcal{SROIQ}$-RL axioms and signature: for example, we have the following rule for concept inclusions: $A \sqsubseteq B \mapsto \{\mathtt{subClass}(A, B, c)\}$. The set of rules P_{rl} are the corresponding deduction rules for axioms in $\mathcal{SROIQ}$-RL: for example, for atomic concept inclusions we have the rule

$$\mathtt{instd}(x, z, c) \leftarrow \mathtt{subClass}(y, z, c), \mathtt{instd}(x, y, c).$$

Global input rules of $I_{glob}(\mathfrak{G})$, basically encode the interpretation of Ctx in the global context. Similarly, local input rules $I_{loc}(\mathrm{K}_{\mathsf{m}}, \mathsf{c})$ and local deduction rules P_{loc} provide the translation and rules for elements of the local object language, in particular for $eval$ expression: e.g., for inclusion of concepts with a left $eval$ expression we have the input rule $eval(A, \mathsf{C}) \sqsubseteq B \mapsto \{\mathtt{subEval}(A, \mathsf{C}, B, \mathsf{c})\}$ and the corresponding deduction rule

$$\mathtt{instd}(x, b, c) \leftarrow \mathtt{subEval}(a, c_1, b, c), \mathtt{instd}(c', c_1, \mathsf{gm}), \mathtt{instd}(x, a, c').$$

The input rules in I_{D} provide the translation of defeasible axioms in the global context: given a defeasible axiom $\mathrm{D}(\alpha)$, $I_{rl}(\alpha, \mathsf{gk})$ is applied and a rule defining when the axiom is locally overridden is added to the program. For example, if $\mathrm{D}(A \sqsubseteq B) \in \mathfrak{G}$, the fact $\mathtt{subClass}(A, B, c).$ is added to the program for the global context together with the corresponding overriding rule:

$$\mathtt{ovr}(\mathtt{subClass}, x, A, B, c) \leftarrow \neg\mathtt{instd}(x, B, c), \mathtt{instd}(x, A, c), \mathtt{prec}(c, \mathsf{g}).$$

The deduction rules in P_{D}, on the other hand, provide the definition of the defeasible propagation of axioms from the global context to the local contexts. For example, the following rule propagates an atomic concept inclusion axiom: if the inclusion axiom is in the program of the global context and can be applied to a local instance, this can be applied if the instance is not recognized as an exception:

$$\texttt{instd}(x, z, c) \leftarrow \texttt{subClass}(y, z, g), \texttt{instd}(x, y, c), \\ \texttt{prec}(c, g), \texttt{not ovr}(\texttt{subClass}, x, y, z, c).$$

Finally, the set of output rules $O(\alpha, \mathsf{c})$ provide the translation of ABox assertions that can be verified by applying the rules of the final program. For example, the case for atomic concept assertions in a given context c is given as $A(a) \mapsto \{\texttt{instd}(a, A, \mathsf{c})\}$.

Given a CKR $\mathfrak{K} = \langle \mathfrak{G}, \{\mathrm{K}_\mathsf{m}\}_{\mathsf{m} \in \mathbf{M}} \rangle$, the translation to its datalog program follows these steps:

1. the *global program* for $\mathfrak{G}$ is translated as:

$$PG(\mathfrak{G}) = I_{glob}(\mathfrak{G}_\Gamma) \cup I_{\mathrm{D}}(\mathfrak{G}_\Gamma) \cup I_{rl}(\mathfrak{G}_\Gamma, \mathsf{gm}) \cup I_{rl}(\mathfrak{G}_\Sigma, \mathsf{gk}) \cup P_{rl}$$

   with gm, gk new context names, $\mathfrak{G}_\Gamma = \{\alpha \in \mathfrak{G} \mid \alpha \in \mathcal{L}^{\mathrm{D}}_\Gamma\}$ and $\mathfrak{G}_\Sigma = \{\alpha \in \mathfrak{G} \mid \alpha \in \mathcal{L}_\Sigma\}$.
2. We define the set of contexts

$$\mathbf{N}_\mathfrak{G} = \{\mathsf{c} \in \mathbf{N} \mid PG(\mathfrak{G}) \models \texttt{instd}(\mathsf{c}, \mathsf{Ctx}, \mathsf{gm})\}$$

   For every $\mathsf{c} \in \mathbf{N}_\mathfrak{G}$, we define its associated knowledge base as:

$$\mathrm{K}_\mathsf{c} = \bigcup \{\mathrm{K}_\mathsf{m} \in \mathfrak{K} \mid PG(\mathfrak{G}) \models \texttt{tripled}(\mathsf{c}, \mathsf{mod}, \mathsf{m}, \mathsf{gm})\}$$

3. We define each *local program* for c as:

$$PC(\mathsf{c}) := P_{loc} \cup P_{\mathrm{D}} \cup I_{loc}(\mathrm{K}_\mathsf{c}, \mathsf{c}) \cup I_{rl}(\mathrm{K}_\mathsf{c}, \mathsf{c}) \cup \{\texttt{prec}(\mathsf{c}, \mathsf{gk}).\}$$

4. The ***CKR program*** is then defined as:

$$PK(\mathfrak{K}) = PG(\mathfrak{G}) \cup \textstyle\bigcup_{\mathsf{c} \in \mathbf{N}_\mathfrak{G}} PC(\mathsf{c})$$

We say that $\mathfrak{G}$ *entails* an axiom $\alpha \in \mathcal{L}_\Gamma$ (denoted $\mathfrak{G} \models\!\!\!\!\!\!\sim_P \alpha$) if $PG(\mathfrak{G})$ and $O(\alpha, \mathsf{gm})$ are defined and $PG(\mathfrak{G}) \models O(\alpha, \mathsf{gm})$. Similarly, $\mathfrak{G}$ entails an axiom $\alpha \in \mathcal{L}_\Sigma$ (denoted $\mathfrak{G} \models\!\!\!\!\!\!\sim_P \alpha$) if $PG(\mathfrak{G})$ and $O(\alpha, \mathsf{gk})$ are defined and $PG(\mathfrak{G}) \models O(\alpha, \mathsf{gk})$. We say that $\mathfrak{K}$ *entails* an axiom $\alpha \in \mathcal{L}^e_\Sigma$ in a context $\mathsf{c} \in \mathbf{N}$ (denoted $\mathfrak{K} \models\!\!\!\!\!\!\sim_P \mathsf{c} : \alpha$), if the elements of $PK(\mathfrak{K})$ and $O(\alpha, \mathsf{c})$ are defined and $PK(\mathfrak{K}) \models O(\alpha, \mathsf{c})$.

Correctness of the translation. In the following we show the correctness of the translation with respect to the problem of instance checking in the presented semantics for CKR.

Let $CAS_\mathbf{N}$ be a map associating every $\mathsf{c} \in \mathbf{N}$ to a set of clashing assumptions. Moreover, given a CKR interpretation $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$, we denote with $CAS^\mathcal{M}_\mathbf{N}$ a corresponding map from elements of the domain of $\mathcal{M}$, i.e. such that if $CAS_\mathbf{N}(\mathsf{c}) = S$, then $CAS^\mathcal{M}_\mathbf{N}(\mathsf{c}^\mathcal{M}) = S$. We define the set:

$$OVR(CAS_\mathbf{N}) = \{\texttt{ovr}(p(\mathbf{e})) \mid \langle \alpha, \mathbf{e} \rangle \in CAS_\mathbf{N}(\mathsf{c}),\ I_{rl}(\alpha, \mathsf{c}) = p\}$$

Given a CKR $\mathfrak{K}$ and its associated CKR program $PK(\mathfrak{K})$, the *OVR-reduct* of $PK(\mathfrak{K})$ (denoted by $G_{OVR}(PK(\mathfrak{K}))$) is the set of rules obtained from $ground(PK(\mathfrak{K}))$ by: (i) removing every rule r such that every occurrence of instances of predicate $\texttt{ovr}$ appears in $OVR(CAS_\mathbf{N})$. (ii) removing the NAF part (corresponding to instances of $\texttt{ovr}$) from the bodies of the remaining rules. We can show that the following property holds:

Lemma 2. $G_{OVR}(PK(\mathfrak{K})) \models O(\alpha, \mathsf{c})$ *iff* $\mathfrak{K} \models_{CAS^{\mathcal{M}}_{\mathbf{N}}} \mathsf{c} : \alpha$.

We can prove the lemma by establishing the following propositions

Proposition 3 (cf. [4]). *For every* α, $ground(PG(\mathfrak{G})) \models O(\alpha, \mathsf{g})$ *iff* $\mathfrak{G} \models \alpha$.

Proposition 4. *For every* α,

1. *(CAS-soundness) If* $G_{OVR}(PK(\mathfrak{K})) \models O(\alpha, \mathsf{c})$, *then* $\mathfrak{K} \models_{CAS^{\mathcal{M}}_{\mathbf{N}}} \mathsf{c} : \alpha$.
2. *(CAS-completeness) If* $\mathfrak{K} \models_{CAS^{\mathcal{M}}_{\mathbf{N}}} \mathsf{c} : \alpha$, *then* $G_{OVR}(PK(\mathfrak{K})) \models O(\alpha, \mathsf{c})$.

The completeness of the translation procedure with respect to CKR semantics can be proved using the following results. Let $S|_p$ be the restriction of an answer set S to the set of facts for predicate p.

Lemma 3. *Given a CAS-model* $\mathfrak{I}_{CAS^{\mathcal{M}}_{\mathbf{N}}} = \langle \mathcal{M}, \mathcal{I}, CAS^{\mathcal{M}}_{\mathbf{N}} \rangle$ *for* $\mathfrak{K}$ *that is justified with* $\mathfrak{K}$, *there exists an answer set* S *of* $PK(\mathfrak{K})$ *such that* $S|_{\mathtt{ovr}} = OVR(CAS_{\mathbf{N}})$.

Lemma 4. *For every answer set* S *of* $PK(\mathfrak{K})$ *and the map* $CAS_S(c) = \{\langle \alpha, \mathbf{e} \rangle \mid I_{rl}(\alpha, c) = p, \mathtt{ovr}(p(\mathbf{e})) \in S\}$, *there exists a CAS-model* $\mathfrak{I}_{CAS^{\mathcal{M}}_S} = \langle \mathcal{M}, \mathcal{I}, CAS^{\mathcal{M}}_S \rangle$ *for* $\mathfrak{K}$ *that is justified for* $\mathfrak{K}$.

The completeness result directly follows from previous results.

Theorem 1. $PK(\mathfrak{K}) \models O(\alpha, \mathsf{c})$ *iff* $\mathfrak{K} \models \mathsf{c} : \alpha$.

5 Conclusion

We presented an extension to the Contextualized Knowledge Repository (CKR) framework introducing a notion of defeasibility of axioms across contexts. We then presented a datalog translation for the extended semantics, based on the materialization calculus for instance checking in [4]. In the translation, non-monotonicity is expressed using answer set semantics, such that instance checking over OWL RL based CKR reduces to cautious inference from all answer sets of the translation.

An implementation of the presented translation in a prototype is ongoing. It basically builds on the *DReW* DL datalog rewriter [16], and extends the basic translation of OWL RL ontologies to the two layered structure of CKR. Given as input an ontology representing the global context and ontologies representing the knowledge modules, the translation produces a datalog program (compliant to DLV syntax) representing the input CKR. More specifically, it encodes the rules and the two layered translation process presented in Section 4: after the translation of the global context, the set of contexts and their associations to modules are derived from the global program through interaction with the DLV solver; with the local knowledge bases defined, the programs for the local contexts are then computed. We aim at extending the current prototype to access external data sources; moreover, we want to compare the query performance with the available implementation of CKR based on SPARQL forward rules [4].

An interesting direction for future investigation is the comparison of the proposed approach to the known approaches to integrate notions of defeasibility and defaults

in description logics. Notable examples of such approaches include representations of typicality in DLs concepts [6] and works employing circumscription in description logics [1]. It is interesting to see whether such notions of defeasibility can be reduced to our representation, thus leading to an implementation of such approaches.

Another natural continuation to the presented work is to allow defeasible axioms across local contexts, possibly along an explicit order relation between contexts (as the *coverage* relation [12]), or across knowledge modules, allowing overriding in specific instances of context classes associated to such modules.

References

1. Bonatti, P.A., Lutz, C., Wolter, F.: Description logics with circumscription. In: KR. pp. 400–410 (2006)
2. Bozzato, L., Ghidini, C., Serafini, L.: Comparing contextual and flat representations of knowledge – a concrete case about football data (2013), accepted at K-Cap 2013.
3. Bozzato, L., Homola, M., Serafini, L.: Towards More Effective Tableaux Reasoning for CKR. In: DL2012. CEUR-WP, vol. 824, pp. 114–124. CEUR-WS.org (2012)
4. Bozzato, L., Serafini, L.: Materialization Calculus for Contexts in the Semantic Web. In: DL2013. CEUR-WP, CEUR-WS.org (2013)
5. Buccafurri, F., Faber, W., Leone, N.: Disjunctive logic programs with inheritance. In: ICLP. pp. 79–93 (1999)
6. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A non-monotonic description logic for reasoning about typicality. Artif. Intell. 195, 165–202 (2013)
7. Khriyenko, O., Terziyan, V.: A framework for context sensitive metadata description. IJSMO 1(2), 154–164 (2006)
8. Klarman, S., Gutiérrez-Basulto, V.: $\mathcal{ALC}_{\mathcal{ALC}}$: a context description logic. In: JELIA (2010)
9. Klarman, S.: Reasoning with Contexts in Description Logics. Ph.D. thesis, Free University of Amsterdam (2013)
10. Krötzsch, M.: Efficient inferencing for owl el. In: JELIA 2010. Lecture Notes in Computer Science, vol. 6341, pp. 234–246. Springer (2010)
11. Motik, B., Fokoue, A., Horrocks, I., Wu, Z., Lutz, C., Grau, B.C.: OWL 2 Web Ontology Language Profiles. W3C recommendation, W3C (Oct 2009), http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/
12. Serafini, L., Homola, M.: Contextualized knowledge repositories for the semantic web. J. of Web Semantics 12 (2012)
13. Straccia, U., Lopes, N., Lukácsy, G., Polleres, A.: A general framework for representing and reasoning with annotated semantic web data. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), Special Track on Artificial Intelligence and the Web (July 2010)
14. Tanca, L.: Context-Based Data Tailoring for Mobile Users. In: BTW 2007 Workshops. pp. 282–295 (2007)
15. Udrea, O., Recupero, D., Subrahmanian, V.S.: Annotated RDF. ACM Trans. Comput. Log. 11(2), 1–41 (2010)
16. Xiao, G., Heymans, S., Eiter, T.: DReW: a reasoner for datalog-rewritable description logics and dl-programs. In: BuRO 2010 (2010)

A Appendix: rules tables

Global input rules $I_{glob}(\mathfrak{G})$

(igl-subctx1) $\mathsf{C} \in \mathbf{C} \mapsto \{\mathtt{subClass}(\mathsf{C}, \mathsf{Ctx}, \mathsf{gm})\}$
(igl-subctx2) $\mathsf{c} \in \mathbf{N} \mapsto \{\mathtt{insta}(\mathsf{c}, \mathsf{Ctx}, \mathsf{gm})\}$

Local input rules $I_{loc}(\mathrm{K}_\mathsf{m}, \mathsf{c})$

(ilc-subevalat) $eval(A, \mathsf{C}) \sqsubseteq B \mapsto \{\mathtt{subEval}(A, \mathsf{C}, B, \mathsf{c})\}$
(ilc-subevalr) $eval(R, \mathsf{C}) \sqsubseteq T \mapsto \{\mathtt{subEvalR}(R, \mathsf{C}, T, \mathsf{c})\}$

Local deduction rules P_{loc}

(plc-subevalat) $\mathtt{instd}(x, b, c) \leftarrow \mathtt{subEval}(a, c_1, b, c), \mathtt{instd}(c', c_1, \mathsf{gm}), \mathtt{instd}(x, a, c')$.
(plc-subevalr) $\mathtt{tripled}(x, t, y, c) \leftarrow \mathtt{subEvalR}(r, c_1, t, c), \mathtt{instd}(c', c_1, \mathsf{gm}), \mathtt{tripled}(x, r, y, c')$.
(plc-eq) $\mathtt{eq}(x, y, c) \leftarrow \mathtt{nom}(x, c), \mathtt{eq}(x, y, c')$.

Output translation $O(\alpha, \mathsf{c})$

(o-concept) $A(a) \mapsto \{\mathtt{instd}(a, A, \mathsf{c})\}$
(o-role) $R(a, b) \mapsto \{\mathtt{tripled}(a, R, b, \mathsf{c})\}$
(o-nconcept) $\neg A(a) \mapsto \{\neg\mathtt{instd}(a, A, \mathsf{c})\}$
(o-nrole) $\neg R(a, b) \mapsto \{\neg\mathtt{tripled}(a, R, b, \mathsf{c})\}$

Table 2. K_{ic} translation and deduction rules

For $\mathrm{D}(\alpha) \in S$, apply $I_{rl}(\alpha)$ and the corresponding rule in the following:

(ovr-inst1)	$\mathrm{D}(A(a))$	$\mapsto \{\, \mathtt{ovr}(\mathtt{insta}, a, A, c) \leftarrow \neg\mathtt{instd}(a, A, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-inst2)	$\mathrm{D}(\neg A(a))$	$\mapsto \{\, \mathtt{ovr}(\neg\mathtt{insta}, a, A, c) \leftarrow \mathtt{instd}(a, A, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-triple1)	$\mathrm{D}(R(a, b))$	$\mapsto \{\, \mathtt{ovr}(\mathtt{triplea}, a, R, b, c) \leftarrow \neg\mathtt{tripled}(a, R, b, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-triple2)	$\mathrm{D}(\neg R(a, b))$	$\mapsto \{\, \mathtt{ovr}(\neg\mathtt{triplea}, a, R, b, c) \leftarrow \mathtt{tripled}(a, R, b, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-inst3)	$\mathrm{D}(\{a\} \sqsubseteq B)$	$\mapsto \{\, \mathtt{ovr}(\mathtt{insta}, a, B, c) \leftarrow \neg\mathtt{instd}(a, B, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-subc)	$\mathrm{D}(A \sqsubseteq B)$	$\mapsto \{\, \mathtt{ovr}(\mathtt{subClass}, x, A, B, c) \leftarrow \neg\mathtt{instd}(x, B, c), \mathtt{instd}(x, A, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-not)	$\mathrm{D}(A \sqsubseteq \neg B)$	$\mapsto \{\, \mathtt{ovr}(\mathtt{supNot}, x, A, B, c) \leftarrow \mathtt{instd}(x, B, c), \mathtt{instd}(x, A, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-cnj)	$\mathrm{D}(A_1 \sqcap A_2 \sqsubseteq \neg B)$	$\mapsto \{\, \mathtt{ovr}(\mathtt{subConj}, x, A_1, A_2, B, c) \leftarrow \neg\mathtt{instd}(x, B, c), \mathtt{instd}(x, A_1, c), \mathtt{instd}(x, A_2, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-subex)	$\mathrm{D}(\exists R.A \sqsubseteq B)$	$\mapsto \{\, \mathtt{ovr}(\mathtt{subEx}, x, R, A, B, c) \leftarrow \neg\mathtt{instd}(x, B, c), \mathtt{tripled}(x, R, y, c), \mathtt{instd}(y, A, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-supex)	$\mathrm{D}(A \sqsubseteq \exists R.\{a\})$	$\mapsto \{\, \mathtt{ovr}(\mathtt{supEx}, x, A, R, a, c) \leftarrow \neg\mathtt{tripled}(x, R, a, c), \mathtt{instd}(x, A, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-forall)	$\mathrm{D}(A \sqsubseteq \forall R.B)$	$\mapsto \{\, \mathtt{ovr}(\mathtt{supForall}, x, y, A, R, B, c) \leftarrow \neg\mathtt{instd}(y, B, c), \mathtt{instd}(x, A, c), \mathtt{tripled}(x, R, y, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-leqone)	$\mathrm{D}(A \sqsubseteq \leqslant 1R.B)$	$\mapsto \{\, \mathtt{ovr}(\mathtt{supLeqOne}, x, y, z, A, R, B, c) \leftarrow \neg\mathtt{eq}(z, y, c), \mathtt{instd}(x, A, c), \mathtt{tripled}(x, R, y, c), \mathtt{tripled}(x, R, z, c), \mathtt{instd}(y, B, c), \mathtt{instd}(z, B, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-subr)	$\mathrm{D}(R \sqsubseteq S)$	$\mapsto \{\, \mathtt{ovr}(\mathtt{subRole}, x, y, R, S, c) \leftarrow \neg\mathtt{tripled}(x, S, y, c), \mathtt{tripled}(x, R, y, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-subrc)	$\mathrm{D}(R \circ S \sqsubseteq T)$	$\mapsto \{\, \mathtt{ovr}(\mathtt{subRChain}, x, y, z, R, S, T, c) \leftarrow \neg\mathtt{tripled}(x, T, z, c), \mathtt{tripled}(x, R, y, c), \mathtt{tripled}(y, S, z, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-dis)	$\mathrm{D}(\mathrm{Dis}(R, S))$	$\mapsto \{\, \mathtt{ovr}(\mathtt{dis}, x, y, R, S, c) \leftarrow \mathtt{tripled}(x, S, y, c), \mathtt{tripled}(x, R, y, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-inv)	$\mathrm{D}(\mathrm{Inv}(R, S))$	$\mapsto \{\, \mathtt{ovr}(\mathtt{inv}, x, y, R, S, c) \leftarrow \neg\mathtt{tripled}(x, S, y, c), \mathtt{tripled}(x, R, y, c), \mathtt{prec}(c, \mathsf{g}).$ $\mathtt{ovr}(\mathtt{inv}, x, y, R, S, c) \leftarrow \neg\mathtt{tripled}(x, R, y, c), \mathtt{tripled}(x, S, y, c), \mathtt{prec}(c, \mathsf{g}).\,\}$
(ovr-irr)	$\mathrm{D}(\mathrm{Irr}(R))$	$\mapsto \{\, \mathtt{ovr}(\mathtt{irr}, x, R, c) \leftarrow \mathtt{tripled}(x, R, x, c), \mathtt{prec}(c, \mathsf{g}).\,\}$

Table 3. Input rules $I_{\mathrm{D}}(S)$ for defeasible axioms

(prop-inst)	$(\neg)\texttt{instd}(x, z, c)$	$\leftarrow (\neg)\texttt{insta}(x, z, g), \texttt{prec}(c, g), \texttt{not ovr}((\neg)\texttt{insta}, x, z, c).$
(prop-triple)	$(\neg)\texttt{tripled}(x, r, y, c)$	$\leftarrow (\neg)\texttt{triplea}(x, r, y, g), \texttt{prec}(c, g), \texttt{not ovr}((\neg)\texttt{triplea}, x, r, y, c).$
(prop-subc)	$\texttt{instd}(x, z, c)$	$\leftarrow \texttt{subClass}(y, z, g), \texttt{instd}(x, y, c), \texttt{prec}(c, g), \texttt{not ovr}(\texttt{subClass}, x, y, z, c).$
(prop-not)	$\neg\texttt{instd}(x, z, c)$	$\leftarrow \texttt{supNot}(y, z, g), \texttt{instd}(x, y, c), \texttt{prec}(c, g), \texttt{not ovr}(\texttt{supNot}, x, y, z, c).$
(prop-cnj)	$\texttt{instd}(x, z, c)$	$\leftarrow \texttt{subConj}(y_1, y_2, z, g), \texttt{instd}(x, y_1, c), \texttt{instd}(x, y_2, c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{subConj}, x, y_1, y_2, z, c).$
(prop-subex)	$\texttt{instd}(x, z, c)$	$\leftarrow \texttt{subEx}(v, y, z, g), \texttt{tripled}(x, v, x', c), \texttt{instd}(x', y, c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{subEx}, x, v, y, z, c).$
(prop-supex)	$\texttt{tripled}(x, r, x', c)$	$\leftarrow \texttt{supEx}(y, r, x', g), \texttt{instd}(x, y, c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{supEx}, x, y, r, x', c).$
(prop-forall)	$\texttt{instd}(y, z', c)$	$\leftarrow \texttt{supForall}(z, r, z', g), \texttt{instd}(x, z, c), \texttt{tripled}(x, r, y, c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{supForall}, x, y, z, r, z', c).$
(prop-leqone)	$\texttt{eq}(x_1, x_2, c)$	$\leftarrow \texttt{supLeqOne}(z, r, z', g), \texttt{instd}(x, z, c), \texttt{tripled}(x, r, x_1, c),$ $\texttt{instd}(x_1, z', c), \texttt{tripled}(x, r, x_2, c), \texttt{instd}(x_2, z', c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{supLeqOne}, x, x_1, x_2, z, r, z', c).$
(prop-subr)	$\texttt{tripled}(x, w, x', c)$	$\leftarrow \texttt{subRole}(v, w, g), \texttt{tripled}(x, v, x', c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{subRole}, x, y, v, w, c).$
(prop-subrc)	$\texttt{tripled}(x, w, z, c)$	$\leftarrow \texttt{subRChain}(u, v, w, g), \texttt{tripled}(x, u, y, c), \texttt{tripled}(y, v, z, c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{subRChain}, x, y, z, u, v, w, c).$
(prop-dis)	$\neg\texttt{tripled}(x, v, y, c)$	$\leftarrow \texttt{dis}(u, v, g), \texttt{tripled}(x, u, y, c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{dis}, x, y, u, v, c).$
(prop-inv1)	$\texttt{tripled}(y, v, x, c)$	$\leftarrow \texttt{inv}(u, v, g), \texttt{tripled}(x, u, y, c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{inv}, x, y, u, v, c).$
(prop-inv2)	$\texttt{tripled}(y, u, x, c)$	$\leftarrow \texttt{inv}(u, v, g), \texttt{tripled}(x, v, y, c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{inv}, x, y, u, v, c).$
(prop-irr)	$\neg\texttt{tripled}(x, u, x, c)$	$\leftarrow \texttt{irr}(u, g), \texttt{nom}(x, c),$ $\texttt{prec}(c, g), \texttt{not ovr}(\texttt{irr}, x, u, c).$

Table 4. Deduction rules P_{D} for defeasible axioms

RL input translation $I_{rl}(S, c)$

(irl-nom)	$a \in \text{NI} \mapsto \{\texttt{nom}(a, c)\}$	(irl-not)	$A \sqsubseteq \neg B \mapsto \{\texttt{supNot}(A, B, c)\}$
(irl-cls)	$A \in \text{NC} \mapsto \{\texttt{cls}(A, c)\}$	(irl-subcnj)	$A_1 \sqcap A_2 \sqsubseteq B \mapsto \{\texttt{subConj}(A_1, A_2, B, c)\}$
(irl-rol)	$R \in \text{NR} \mapsto \{\texttt{rol}(R, c)\}$	(irl-subex)	$\exists R.A \sqsubseteq B \mapsto \{\texttt{subEx}(R, A, B, c)\}$
(irl-inst1)	$A(a) \mapsto \{\texttt{insta}(a, A, c)\}$	(irl-supex)	$A \sqsubseteq \exists R.\{a\} \mapsto \{\texttt{supEx}(A, R, a, c)\}$
(irl-inst2)	$\neg A(a) \mapsto \{\neg\texttt{insta}(a, A, c)\}$	(irl-forall)	$A \sqsubseteq \forall R.B \mapsto \{\texttt{supForall}(A, R, B, c)\}$
(irl-triple)	$R(a, b) \mapsto \{\texttt{triplea}(a, R, b, c)\}$	(irl-leqone)	$A \sqsubseteq \leqslant 1 R.B \mapsto \{\texttt{supLeqOne}(A, R, B, c)\}$
(irl-ntriple)	$\neg R(a, b) \mapsto \{\neg\texttt{triplea}(a, R, b, c)\}$	(irl-subr)	$R \sqsubseteq S \mapsto \{\texttt{subRole}(R, S, c)\}$
(irl-eq)	$a = b \mapsto \{\texttt{eq}(a, b, c)\}$	(irl-subrc)	$R \circ S \sqsubseteq T \mapsto \{\texttt{subRChain}(R, S, T, c)\}$
(irl-neq)	$a \neq b \mapsto \{\neg\texttt{eq}(a, b, c)\}$	(irl-dis)	$\text{Dis}(R, S) \mapsto \{\texttt{dis}(R, S, c)\}$
(irl-inst3)	$\{a\} \sqsubseteq B \mapsto \{\texttt{insta}(a, B, c)\}$	(irl-inv)	$\text{Inv}(R, S) \mapsto \{\texttt{inv}(R, S, c)\}$
(irl-subc)	$A \sqsubseteq B \mapsto \{\texttt{subClass}(A, B, c)\}$	(irl-irr)	$\text{Irr}(R) \mapsto \{\texttt{irr}(R, c)\}$
(irl-top)	$\top(a) \mapsto \{\texttt{insta}(a, \texttt{top}, c)\}$		
(irl-bot)	$\bot(a) \mapsto \{\texttt{insta}(a, \texttt{bot}, c)\}$		

RL deduction rules P_{rl}

(prl-instd)	$(\neg)\texttt{instd}(x, z, c)$	$\leftarrow (\neg)\texttt{insta}(x, z, c).$
(prl-tripled)	$(\neg)\texttt{tripled}(x, r, y, c)$	$\leftarrow (\neg)\texttt{tripled}(x, r, y, c).$
(prl-eq1)	$\texttt{eq}(x, x, c)$	$\leftarrow \texttt{nom}(x, c).$
(prl-eq2)	$\texttt{eq}(y, x, c)$	$\leftarrow \texttt{eq}(x, y, c).$
(prl-eq3)	$(\neg)\texttt{instd}(y, z, c)$	$\leftarrow \texttt{eq}(x, y, c), (\neg)\texttt{instd}(x, z, c).$
(prl-eq4)	$(\neg)\texttt{tripled}(y, u, z, c)$	$\leftarrow \texttt{eq}(x, y, c), (\neg)\texttt{tripled}(x, u, z, c).$
(prl-eq5)	$(\neg)\texttt{tripled}(z, u, y, c)$	$\leftarrow \texttt{eq}(x, y, c), (\neg)\texttt{tripled}(z, u, x, c).$
(prl-eq6)	$\texttt{eq}(x, z, c)$	$\leftarrow \texttt{eq}(x, y, c), \texttt{eq}(y, z, c).$
(prl-top)	$\texttt{instd}(x, \texttt{top}, c)$	$\leftarrow \texttt{instd}(x, z, c).$
(prl-subc)	$\texttt{instd}(x, z, c)$	$\leftarrow \texttt{subClass}(y, z, c), \texttt{instd}(x, y, c).$
(prl-not)	$\neg\texttt{instd}(x, z, c)$	$\leftarrow \texttt{supNot}(y, z, c), \texttt{instd}(x, y, c).$
(prl-subcnj)	$\texttt{instd}(x, z, c)$	$\leftarrow \texttt{subConj}(y_1, y_2, z, c), \texttt{instd}(x, y_1, c), \texttt{instd}(x, y_2, c).$
(prl-subex)	$\texttt{instd}(x, z, c)$	$\leftarrow \texttt{subEx}(v, y, z, c), \texttt{tripled}(x, v, x', c), \texttt{instd}(x', y, c).$
(prl-supex)	$\texttt{tripled}(x, r, x', c)$	$\leftarrow \texttt{supEx}(y, r, x', c), \texttt{instd}(x, y, c).$
(prl-supforall)	$\texttt{instd}(y, z', c)$	$\leftarrow \texttt{supForall}(z, r, z', c), \texttt{instd}(x, z, c), \texttt{tripled}(x, r, y, c).$
(prl-leqone)	$\texttt{eq}(x_1, x_2, c)$	$\leftarrow \texttt{supLeqOne}(z, r, z', c), \texttt{instd}(x, z, c), \texttt{tripled}(x, r, x_1, c),$ $\texttt{instd}(x_1, z', c), \texttt{tripled}(x, r, x_2, c), \texttt{instd}(x_2, z', c).$
(prl-subr)	$\texttt{tripled}(x, w, x', c)$	$\leftarrow \texttt{subRole}(v, w, c), \texttt{tripled}(x, v, x', c).$
(prl-subrc)	$\texttt{tripled}(x, w, z, c)$	$\leftarrow \texttt{subRChain}(u, v, w, c), \texttt{tripled}(x, u, y, c), \texttt{tripled}(y, v, z, c).$
(prl-dis)	$\neg\texttt{tripled}(x, v, y, c)$	$\leftarrow \texttt{dis}(u, v, c), \texttt{tripled}(x, u, y, c).$
(prl-inv1)	$\texttt{tripled}(y, v, x, c)$	$\leftarrow \texttt{inv}(u, v, c), \texttt{tripled}(x, u, y, c).$
(prl-inv2)	$\texttt{tripled}(y, u, x, c)$	$\leftarrow \texttt{inv}(u, v, c), \texttt{tripled}(x, v, y, c).$
(prl-irr)	$\neg\texttt{tripled}(x, u, x, c)$	$\leftarrow \texttt{irr}(u, c), \texttt{nom}(x, c).$

Table 5. RL input and deduction rules

User Preference Learning for Context-Aware Situation Identification

Judie Attard, Simon Scerri, Ismael Rivera, and Siegfried Handschuh

Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland
firstname.lastname@deri.org

Abstract. Current smart devices, such as laptops and mobile phones, are generating relevant context information, but in a disconnected fashion. This brings a limitation to a smart device's capability of becoming aware of the user's current context. The unification of such information can thus be exploited in order to recognize recurring situations, and provide context-aware suggestions/functions to the user. With the aim of enhancing such a system, in this paper we extend an ontology-based graph matching technique which continuously compares a person's current context to a number of previously stored situations. We extend this technique with a feedback loop which learns a user's preferences during the lifetime of the system, with the aim of improving recurring situation identification.

1 Introduction

Information extracted from last-generation personal devices has the potential of indicating the owner's habitual tasks, activities and actions. Yet, however rich the information, it must be unified and interpreted in order to provide context-aware features to the user. Apart from lacking the unification of such data, personal devices are also lacking in models required to represent core context-aware concepts. This deficiency results in any context-aware techniques to be system-specific; with limited re-use and/or extension outside the native environment. Yet another vulnerability of current context-aware devices is the fact that user situation representations are static, while in fact a user's situations will change over time.

In their daily life, users of modern technology and smart devices (representative of most users in the future) encounter repeated situations which necessitate repeated intervention, for example changing device mode or availability when entering a workplace, forwarding emails from a particular sender to a specific contact, or revoking any privacy-sensitive information from being shared in an area when untrusted/unknown persons are in the vicinity. To enable such situation-based recommendation, our first aim is to develop a context-aware system. This aim is also shared by the di.me[1] project, which seeks to provide context-aware warnings, suggestions and automation. Our objective is thus the automatic recognition of recurring situations by continuously monitoring a user's presence and comparing their dynamic contextual information to a set of manually-stored situations. This requires the unification and interpretation of context information. Although the term 'context information' is open to many interpretations,

[1] http://dime-project.eu/

we stick to Dey's definition, i.e., any kind of information that can be used to characterise the situation of an entity [4]. In this case, the entity refers to any person using today's technology, devices and online services. The di.me approach builds upon an interoperable knowledge representation format for representing, amongst others, context knowledge and context-driven rules. This representation format is provided by an ontology framework which enables the collection and integration of a user's activity context streams from various personal devices.

The DCON[2] ontology, which we use to represent a user's activity context, is integrated with the framework, thus extending the *Personal Information Model* (PIM) representation with personal 'context' information.

Context extraction techniques are employed by the semantic lifting process within the di.me system to populate the *Personal Information Model* (PIM) with all kinds of extracted personal information. This includes presence knowledge, for which two sources for mining can be identified:

- device usage & sensors - context information relayed by device-embedded sensors, user attention monitoring, etc.;
- social sharing activities - context information provided directly or indirectly by the user through their Social Web activities.

Through the user's intervention (saving situations, places of interest, etc.), the information extracted from user devices can then be interpreted into higher-level abstractions. For example, a wireless access point can be linked to particular geographical locations. This information can be used to infer a user's current location. Similarly, bluetooth connections can be useful for determining which people (owning the bluetooth device) are in the vicinity, once they are already known to the PIM. Another example is to infer the current user activity from their location, such as when the user is in a location marked to be a restaurant, we can infer the user is eating.

Our second objective is to adapt our situation detection mechanism using user-feedback, with the aim of improving our situation representations. If a situation is proposed to the user to be recurring, the user is allowed to accept or reject the suggested situation. This small effort, which will be minimally intrusive, will have a low impact on the system usability. Yet, this improvement should result in a more accurate situation recognition component. This is done by adapting the stored situations using the given user feedback, along with the current user context.

In the rest of the paper, we first compare related work in Section 2, before providing the details of our technique in Section 3. In Section 4 we then report and discuss the results of its evaluation. Concluding remarks and future work plans are included within the Conclusion in Section 5.

2 Related Work

Although there have been many efforts in the area of context-aware systems, we hereby compare our approach to only the ones that are most similar in terms of both final objectives and candidate techniques. [14], [2] and [8] are approaches comparable to ours

[2] `http://www.semanticdesktop.org/ontologies/dcon/`

in the sense that their main goal is to match a person's current context to a set of states. While they all extract the user context from sensors worn on the user's body, our proposed context-aware system targets more realistic context sensors in order to interpret a person's different 'states'. As opposed to the above techniques, we do not implement machine learning or classification techniques to achieve our aim of identifying a user's current state. Rather, we implement ontology-based instance matching, since our context is represented as instances of the DCON ontology.

Since both live context and situation representations are available as unified DCON instances, i.e., as RDF graphs, we turn our attention to graph-based ontology matching techniques. Our context representations include both instance/data and schema/structure levels [10]. Given that the graph structure in our case has been standardised by DCON, approaches which attempt to discover common substructures between graphs are irrelevant for our task. Therefore, the success of our context matching technique depends on how able it is to compare two graph instances at the data/content level.

Kirsh-Pinheiro et al. in [7] propose a graph-based approach where service descriptions and requests are interpreted as graphs, with the aim of matching contextual service descriptions using similarity measures, allowing inexact matches. Service description and requirements are compared using two kinds of similarity measures. Similarly to [7], our context representations are graph-based, however our similarity calculation is somewhat different. Firstly, the process is reversed; we start by matching our context representation graphs as a whole, instead of starting matching nodes individually. In this way, only nodes belonging to DCON ontology elements of the same type (e.g. temperature, place) are compared locally. Secondly, we perform constraint-based matching on the ontology elements according to their type (e.g. type person) to determine their similarity. Constraint-based matching is driven by the ranges specified for each DCON context attribute (e.g., wifi signal's can be between 0 and 5, inclusive). The similarity results for all the ontology elements compared between both context representations are then combined to achieve a global similarity value, similar to the approach taken by Kirsh-Pinheiro et al.

The approaches of [5] and [13] are similar to our approach in many ways. We implement two different similarity measures for three distinct layers. We first compare the ontological type of each entity, enabling us to match entities of the same type, e.g person, then we compare the entities' values, where simple and complex data types are compared using different similarity metrics for each type. Another similarity of our approach to the work in [5] and [13] is the use of weights in the calculation of the global similarity to bias the results. Since we consider some entities to be more defining of the user context than other, we assign higher weights to such entities, thus enabling us to bias the result.

SemMF [9] is a semantic matching framework which enables the calculation of semantic similarity between RDF graph representations. It allows for taxonomic and non-taxonomic concept matching techniques to be implemented on particular elements, which can also be assigned varying weights. This made SemMF an instant candidate for our context matching requirements, although a number of limitations were instantly identified. These limitations were tackled in [1].

The idealistic vision of context aware systems is to serve users non-obtrusively in their every-day environment, whilst constantly monitoring context changes and adapting their behavior and functionality with the aim of maximizing the user benefit. While many approaches target this vision, the self-adapting feature of such systems are far from trivial. Here we compare systems which propose a solution to this feature.

While having a similar user-sensitive weighting approach to [6] and [16], in contrast we adopt a semantic approach where a user's context is represented as an RDF graph. Our weighting system is based on the idea that each context element (feature) has a value indicating its significance in defining a particular instance, as opposed to defining all instances. For example, consider a user who saves two situations, namely 'Working' and 'At home', in a short span of time. In this case, location, as opposed to temperature, is more defining of the saved situations. These weights are then adjusted in the feedback loop based on users' feedback for each instance; where context elements identified to be important in characterising a situation are given higher weights, and vice versa. Although the techniques implemented in [6] and [16] adopt a weighting system which uses user feedback to update feature weights, they do not allow different weights for the same context element in different instances. On the other hand, techniques similar to the ones reviewed in [15] require training, usually needing a large amount of pre-existing training data. This makes them unsuitable for our purpose since we cannot have any data before the di.me userware is being used, and also due to its personalisable nature. This means that data generated by a particular user cannot be used to train the system for generic use. In contrast, the learning process starts on first use, and continues during the userware's lifetime. Thus, an alternative and more adequate learning function has been designed and implemented, as explained in Section 3.

3 Approach

In this section we describe the techniques underlying our situation recognition component, focusing mainly on the semi-automatic situation adjustment technique. The lifetime of the system starts with the personal user devices; context data is extracted, stored, and interpreted in order to represent the current user context. Through the users' intervention, specific snapshots of the user context are stored as situations. These represent particular scenarios which are of relevance to the user, such as 'Eating', 'Shopping', 'At the Pub' and 'At Work'. The aim of the context matching process is to identify recurring situations. Thus, the current user live context is compared with all stored situations, in order to determine the situations which best match with the live context. Those situations which have the highest similarity score are then suggested to the user to be potential recurring situations. The feedback loop then enables the user to confirm or reject these suggestions. This feedback is then used to adapt the stored situations.

3.1 Context Extraction

Before going into the details of our context representation schema, we first explain our general approach towards handling and interpreting context information. Specifically, we differ between three types of context information; namely raw context, interpreted context, and situations [11]. Raw context is retrieved from a number of various

personal devices (e.g. computers and smartphones) and services (e.g. social networks and weather services) which make up a personal sensor network. Interpreted context gives us additional user activity context from the retrieved sensor data. In di.me, examples of interpreted context include person(s) proximity detection via network connections, place types based on the user's own tags and publicly available maps (e.g. 'home', 'workplace', 'restaurant', 'sports centre', 'airport'). More sophisticated techniques are used to infer user activities (e.g. working, eating, travelling) from the discovered place type (e.g. restaurant: 'eating'), information extraction techniques to analyse social network status messages (e.g. "On my way to the Airport": 'travelling'), and known associations between certain applications/files being used or edited on either of the devices (e.g. application Eclipse in foreground: 'working'). In addition, discrete data retrieved from sensors (e.g. 23°C, 15:35pm, 54dB) can also be transformed into predefined fuzzy classifications (e.g. temperature: 'warm', time period: 'late afternoon', noise level: 'quiet').

Our approach focuses on two context representations; the *dcon:LiveContext* and the *dcon:Situation* representations. Although both a live context and a situation are composed of a combination of raw and interpreted context elements, the former consists of a single and continuously updated context snapshot which is bound to a specific point in time. This represents the user's *transitory* context. In contrast, a situation can refer to multiple live context snapshots, each occurring at different times. Thus, the relationship between a situation and a live context is that the former can be 'characterised' through a series of positive and negative examples, each corresponding to live context observations taken at different times. Examples of situations are 'Working', 'Travelling' and 'At the Pub'.

3.2 Context Representation

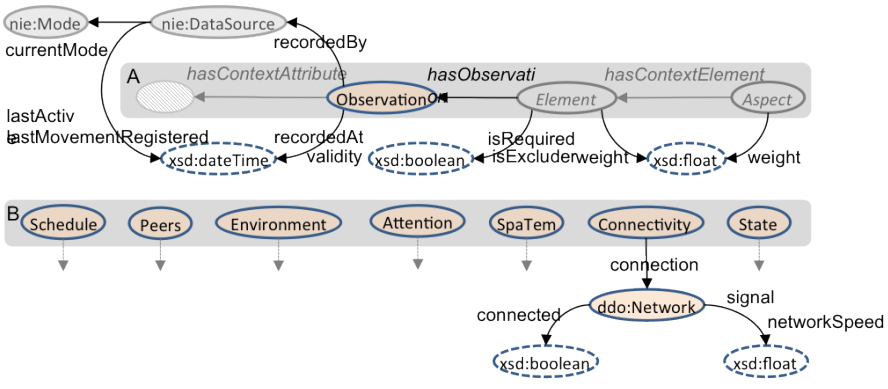

Fig. 1. The DCON Context Ontology

The di.me Context Ontology (DCON) enables a comprehensive representation of an entity's context, as acquired from device and user activity sensors [11]. DCON context instances are each stored as an RDF named graph [3]. DCON separates between three different layers of context abstractions, based on a model provided by Schwarz [12]. As shown in Fig. 1 (grey box 'A'), information corresponding to a *Context* is categorised under different kinds of *Aspects*, each of which consists of various *Elements*, which in turn posses various context *Attributes*. In addition, since we retrieve context information from multiple devices simultaneously, this might result in multiple *Observations* for the same context element (e.g., a wifi is detected by both a person's laptop and their smartphone). A formal definition of DCON is available in [1].

In the latest DCON, the number of Aspects (grey box 'B') is 7, each of which groups a number of context elements:

1. **Schedule** - groups known events and tasks as its elements. Tasks may be current or upcoming;
2. **Peers** - groups nearby contacts and groups of contacts as saved by the user in the PIM;
3. **Environment** - various sensor data including temperature, brightness and noise, and a number of weather conditions;
4. **Attention** - user device activities including active files and applications;
5. **Spatial/temporal data** - data indicating the location and time period of the current context, including speed, direction, altitude and time period;
6. **Connectivity** - any networks detected from the used devices;
7. **States** - current user availability and any current physical activities.

Each of the embedded elements, which in the current DCON number 20, can be further described using domain ontologies, such as the Presence Ontology (DPO)[3] and the Information Element Ontology set (NIE)[4], to store details such as a file's size, a contact's address, a wifi's mac address, etc. The DPO also provides a fuzzy abstraction to the relevant context elements, e.g., according to the brightness level and current temperature, the brightness and temperature are taken to be 'Indoor Light' and 'Warm' respectively. Context-dependent information about these elements, however, is only attached through DCON vocabulary. Therefore, no external ontology vocabulary, beside the element type itself, is taken into consideration by our context matching algorithm. Instead, DCON provides a list of context attributes (numbering 39 in the latest version) that are indicative of an entity's context (e.g. *dcon:connected* indicates a nearby wifi's connection status, *dcon:distance* indicates the physical distance to nearby locations, etc.). Aspect *Connectivity* is shown in Fig. 1, along with its elements and attributes, as an example. As explained above, these context attributes will not be attached to the element itself, but to its observations. This is because different devices may experience different context attributes for an element, such as when a person is nearer to your laptop, than to your phone. Apart from the element-specific context attributes, DCON also provides three generic ones which particularly aid the matching process. As shown in Fig. 1-A, *dcon:recordedBy* refers to the device through which an observation has been recorded,

[3] `http://www.semanticdesktop.org/ontologies/dpo/`
[4] `http://www.semanticdesktop.org/ontologies/nie/`

whereas *dcon:recordedAt* indicates the observation time. In addition, the *dcon:validity* attribute predefines a span of time during which an observation is assumed to remain relevant (e.g. active applications context changes faster than the outside temperature), and is intended for use by context-aware systems like di.me to indicate the required observation update frequency.

Apart from the specific types of context aspects, elements and their attributes, DCON also provides generic vocabulary that can be used to optimise the context matching task (top-left corner of Fig. 1). The *dcon:weight* property can be used to bias certain aspects and/or elements in relation to a situation (e.g. the Spatial-Temporal aspect as a whole is more indicative of a situation, or the presence of a specific person reduces the likelihood of a situation being active). In addition, DCON also provides two special markers that define which elements are necessary for recognising a particular context/situation (*dcon:isRequired*), and the inverse, i.e. its presence would right away eliminate a particular situation from being matched (*dcon:isExcluder*). For example, if a situation 'Eating' has location element 'kitchen' marked as dcon:isRequired, then for that situation to recur, 'kitchen' must exist in the current user life context. These properties have the effect of pre-filtering the candidate situations that are to be matched.

Live Context and Situations are both subclasses of the generic *dcon:Context* class. As they consequently have the same structure, this simplifies our context matching task. However, a *dcon:Situation* represents a recurrent context abstraction that can be characterised by one or more past *dcon:LiveContext* snapshots, each of which can be either a negative or positive example (*dcon:positiveInstance* or *dcon:negativeInstance*) of the *dcon:Situation*. Thus when a situation is correctly identified to be currently recurring, the current user live context is taken to be a positive example of the situation in question. The opposite is true when a situation is incorrectly identified to be currently recurring. In this way, situations are gradually 'trained' in order to achieve a better representation. Positive and negative examples can be persisted as instances of *duho:ContextLog*. The User History Ontology (DUHO)[5] enables the logging of past user information, including context.

3.3 Identifying Recurring Situations

In order to enable the automatic recognition of recurring situations, we require a mechanism for matching the continuously updated live context against any number of stored situations. The job of the context matching process is to compare the live context graph contents to all stored situations in order to find the highest degree of similarity between the contained elements. In order to compute the degree of similarity between two context graphs, we adapt and extend the SemMF [9] matching framework to be able to take as input two DCON instances and compare their contents at the three schema levels. We adopt a top-down approach, whereby we start by comparing each available context aspect (e.g. Connectivity) in the live context, against the same aspect in the candidate situation graph. For each aspect, we then obtain all available elements (e.g. detected network connections) from both graphs. A confusion matrix is created to match elements of the same *hasContextElement* sub-property (e.g. wifi connections). Although we are

[5] `http://www.semanticdesktop.org/ontologies/duho/`

able to identify common elements occurring in both graphs (e.g. same wifi network), we also compare non-equivalent elements to gauge their similarity. To this end, we consider the available context attributes for each element, and execute various constraint-based matchers depending on the attribute type and their specified range.

The implemented similarity measure also employs a weighting system which is used to give prominence to context information that is either considered to be more representative of the situation (positive bias), or to decrease the likelihood of that situation having been reactivated (negative bias). Aspects, elements and attributes can all be assigned a numeric weight, although the idea for weight assignment in the di.me system differs for each of the three levels. When persisting a situation, the current di.me prototype enables users to (optionally) manually assign weights to detected context elements, in order to mark those which they consider to be most representative (high weight) of the situation they are saving. For example, a user might consider his location to be more important in defining a situation 'Working' than the current time, since they might work different hours but always in the same location. In addition, they can also mark (either at the creation stage or later) those elements which would normally exclude the situation from being a candidate for the automatic situation recognition, as well as those which are required. Thus Element Weights w_e have a range of: $-1 \leq w_e \leq 1$, where a negative weighting indicates that a context element reduces the likelihood that a situation containing that element is suggested to be recurring. Aspect Weights w_a are all equalised at the start of the situation lifecycle, with a range of $0 \leq w_a \leq 1$. Attributes also have weights, which however are pre-defined by the schema. Attribute weights determine the internal ratio of each attribute for a specific context element, such that some are given more prominence (e.g. a nearby wifi's signal is more indicative of a user's situation than a wifi's connection speed). Thus Attribute Weights w_t have a range of: $0 \leq w_t \leq 1$ such that $\sum_{n=1}^{g}(w_t) = 1$, where g is the number of attributes compared for the element in question, $g \leq v$, and v is the number of possible attributes for an element.

Thus, the context aspect, element and attribute matching process described above, coupled with any corresponding weights, enable us to define a similarity metric that calculates the overall graph similarity between a live context and one or more candidate situations. Based on the resulting matching scores from the context matching algorithm, the di.me UI will then notify the user of possible recurring situations, by suggesting the top three matching situations when compared to the current user *dcon:LiveContext*. These results are then used, alongst user feedback, as an input to the semi-automatic situation adjustment feature.

3.4 Semi-automatic Situation Adjustment

In order to adjust and better characterize represented situations, we couple the context matching algorithm with a feedback loop which enables the situation recognition technique to self-adapt over time. This functionality requires interaction with the user, as a form of gradual training. The user will be able to interact with the UI to say whether any of the suggested situations are really taking place or otherwise. If the user confirms the suggestion, the current user *dcon:LiveContext* instance is taken to be a positive example (*dcon:positiveInstance*) of the situation. On the other hand, if the user

rejects the suggestion, the *dcon:LiveContext* instance is stored as a negative example (*dcon:negativeInstance*). These examples will be used to automatically improve situation representations.

Whenever a new (positive or negative) context example for a situation is logged, the element weights will automatically and gradually be adjusted to reflect the new example. Context examples improve the situation representation in two ways:

1. by extending the situation with newly observed elements and aspects, having positive/negative initial weights, depending on the nature of the context example (positive/negative);
2. by modifying the weights of existing situation elements that are also observed in the context example, such that they are reduced or increased appropriately given the nature of the example.

As an example, consider a person being in a social dinner with colleagues. Previously, they have saved a number of situations, including "@BusinessDinner" and "@BusinessLunch". Detecting a number of common elements and their attributes, the context matching component will present both situations to the user, who then confirms that "@BusinessDinner" is representative of their current situation, whereas the other suggestion is rejected. This user feedback will result in the current live context to be stored as a *dcon:positiveInstance* and a *dcon:negativeInstance*, of the confirmed and rejected situations respectively. Both situations are then adjusted so as to better reflect this new knowledge. Here, the system will identify previously unobserved context elements, such as a *dpo:TimePeriod* instance that represents the evening. This element will be introduced under the Spatial-Temporal context aspect (*dcon:currentTime*) for both situations, with initial positive weights for the "@BusinessDinner" situation, and negative weights for "@BusinessLunch". Secondly, the weights of existing items will be adjusted. For example, whereas in previous occurrences of the confirmed situation person 'John Doe' was always observed, in the latest positive example the same element is not part of the Peers aspect. From this, one can interpret that this person is not very important for the characterisation of this situation, and therefore its weight is automatically reduced.

$$w' = 2 * \frac{2}{\pi} * arctan(w) \tag{1}$$

Formula 1 is the tangential formula used for the training process, where w and w' are the original and the new weight respectively. The formula is shown as the dotted line in Figure 2. This formula is used for positive weights in positive live context instances, as well as for negative weights in negative live context instances. This formula thus increases positive weights while decreasing negative weights. On the other hand, the dashed line represents an adaptation of Formula 1, where the new weight w'' is now $w - (w' - w)$, giving us increased negative weights in positive live context instances and reduced positive weights in negative instances. This formula is used because of its curve, being steep where the value of w is small, and getting gentler where w nears 1. This is ideal for our training process as we require weights to initially change in a fast manner, then slowing down as they reach the limits defined for w.

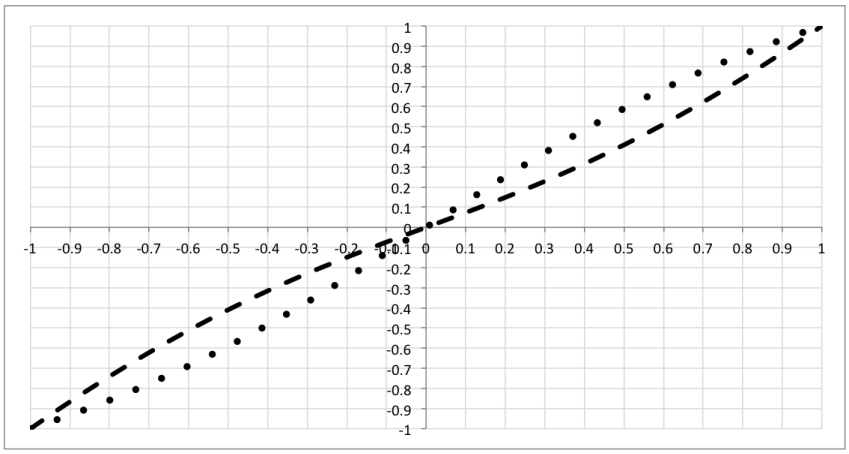

Fig. 2. The Weight Modifying Functions

4 Evaluation

In this section we describe and discuss the results of an evaluation of our semi-automatic situation adjustment technique, led as a continuation to the context matching technique experiments in [1]. For the former evaluation, we use the same live context and situation instances used in the latter experiments, where evaluators 1 to 7 each identified 3, 2, 3, 2, 1, 1 and 2 situations, and a corresponding 19, 16, 23, 16, 8, 8 and 14 live contexts respectively, resulting in 62 distinct live contexts[6]. The aim of this experiment is to determine whether the similarity results change after the training process, i.e. a higher similarity for matching instances, lower similarity for non-matching instances. Due to the unavailability of the UI component which executes the feedback loop, we simulate user feedback by using the live contexts identified by the evaluators, which are initially marked as being a positive or a negative instance of a situation. We execute our experiment in three steps:

1. Comparing a situation with a live context;
2. Training the situation with the other positive and negative live context instances;
3. Comparing the modified situation with the same live context as used in step 1.

For each situation $s_{original}$ identified by each evaluator, we thus selected two live contexts; a positive instance lc_p and a negative instance lc_n of $s_{original}$. We then ran the context matching technique, comparing $s_{original}$ with lc_p and lc_n. This gave us the 'original' similarity values sim_o. We proceeded to simulate the feedback loop, by using all the other live contexts **not** used in the initial matching process, i.e. for $s_{original}$ we excluded lc_p and lc_n. During this process, the situation is modified in order to reflect

[6] This data is available online at `http://smile.deri.ie/Context_Matching_Evaluation`

the live context instances, modifying weights and adding elements accordingly, as explained in Section 3.4. After the training process is completed, we then compare the modified situation $s_{modified}$ to lc_p and lc_n, giving us the new similarity values sim_m.

To determine the effectivity of the semi-automatic situation adjustment technique, we compare the sim_o and sim_m, for each situation - live context couple. While for positive live context instances, the similarity values should be higher, for negative live context instances the similarity values should now be lower. For the former, sim_m gave us 11 higher values and 3 lower ones. This means that for 11 situations, the live contexts matched better after the training process. For the latter, sim_m gave us 7 lower values and 7 higher ones, meaning that while for 7 situations the negative live context instances matched less, another 7 live contexts matched better.

To further gauge the improvement of the results we then calculate the average difference between sim_o and sim_m, which gave us 0.006524831 for positive live context instances and 0.001781389 for negative live context instances. This shows that while all the situation - live context couples have higher similarity values, those for positive live context instances have a larger change. On the other hand, the negative live context instances have a small change in the similarity value.

Overall, the training process ensued in better similarity values, thus indicating that the situations are better characterised after the training process. Considering the small amount of live contexts per user we could use as training data, the change in similarity results was considerable. This indicates that during the life time of the system, the situations can be characterised better due to the constant increase in training data, and thus the recurring execution of the feedback loop.

5 Conclusion

In this paper we presented an extension to the ontology-based situation recognition technique proposed in [1]. For the semi-automatic situation adjustment we exploit user feedback to characterise situations in order to enhance the matching process. When the system identifies potential situations to be recurring, the latter are suggested to the user. The user then accepts or rejects each suggestion. For each situation, the current user life context is used to update the situation in question; adding any new elements and updating weights according to if the situation is accepted or rejected. We use a tangential equation in order to increase or decrease weights in a rapid curve when not near the defined limits, then more slowly as the weights reach limits. We then proceeded to evaluate the efficacy of the proposed training process, concluding that overall, keeping in mind the small amount of data used to train the system, the matching process fared better after the training process. Future work will see us focusing our efforts on optimising the formula used for the weight-adjustment performed by the feedback loop, in order to speed up and refine the situation adjustment.

Acknowledgment

This work is supported in part by the European Commission under the Seventh Framework Program FP7/2007-2013 (*digital.me* – ICT-257787) and in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (*Líon-2*).

References

1. J. Attard, S. Scerri, I. Rivera, and S. Handschuh. Ontology-based situation recognition for context-aware systems. In *I-SEMANTICS*, 2013.
2. M. Blum, A. Pentland, and G. Tröster. Insense: Interest-based life logging. *IEEE MultiMedia*, 13(4):40–48, 2006.
3. J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *Proceedings of the 14th international conference on World Wide Web*, page 613, New York, New York, USA, 2005. ACM Press.
4. A. K. Dey. Understanding and using context. Technical report, Future Computing Environments Group, Georgia Institute of Technology, Atlanta, GA, USA, 2001.
5. M. Ehrig, P. Haase, M. Hefke, and N. Stojanovic. Similarity for ontologies - a comprehensive framework. In *ECIS*, pages 1509–1518, 2005.
6. K. Kakousis, N. Paspallis, and G. A. Papadopoulos. Optimizing the utility function-based self-adaptive behavior of context-aware systems using user feedback. In *OTM Conferences (1)*, pages 657–674, 2008.
7. M. Kirsch-pinheiro, Y. Vanrompay, and Y. Berbers. Context-aware service selection using graph matching. In *Proceedings of the CEUR Workhop*, 2008.
8. A. Krause, A. Smailagic, and D. P. Siewiorek. Context-aware mobile computing: Learning context-dependent personal preferences from a wearable sensor array. *IEEE Transactions on Mobile Computing*, 5:113–127, 2006.
9. R. Oldakowski and C. Bizer. SemMF: A Framework for Calculating Semantic Similarity of Objects Represented as RDF Graphs. *Poster at the 4th International Semantic Web Conference (ISWC 2005)*, 2005.
10. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB JOURNAL*, 10:2001, 2001.
11. S. Scerri, J. Attard, I. Rivera, M. Valla, and S. Handschuh. Dcon: Interoperable context representation for pervasive environments. In *In Proceedings of the Activity Context Representation Workshop at AAAI 2012*, 2012.
12. S. Schwarz. A context model for personal knowledge management applications. In *Modeling and Retrieval of Context, Second International Workshop, MRC 2005, Edinburgh, UK, July 31 - August 1, 2005, Revised Selected Papers*, volume 3946 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2006.
13. R. Shkundina and S. Schwarz. A similarity measure for task contexts. In *In Proceedings of the Workshop Similarities - Processes - Workflows in conjunction with the 6th International Conference on Case-Based Reasoning*, 2005.
14. D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F. L. Wong. Sensay: A context-aware mobile phone. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, ISWC '03, pages 248–, Washington, DC, USA, 2003. IEEE Computer Society.
15. J. Ye, S. Dobson, and S. McKeever. Review: Situation identification techniques in pervasive computing: A review. *Pervasive Mob. Comput.*, 8(1):36–66, Feb. 2012.
16. Z. Zhang and Q. Yang. Feature weight maintenance in case bases using introspective learning. *J. Intell. Inf. Syst.*, 16(2):95–116, 2001.

ARCOE-LogIC 2013 Workshop Notes