Proc. 23rd Int. Workshop on Description Logics (DL2010), CEUR-WS 573, Waterloo, Canada, 2010.

A Multi-Context System Computing Modalities

Tarek R. Besold[1] and Bernhard Schiemann[2]

[1] University of Erlangen-Nuremberg,
Chair of Computer Science 8: Artificial Intelligence,
D-91058 Erlangen, Germany
sntabeso@i8.informatik.uni-erlangen.de
[2] Bernhard.Schiemann@googlemail.com

Abstract. The system description presents the conception and a prototypical implementation of a multi-context system, used for computing and implementing temporal modalities within given data without the use of modal operators. Instead, an external constraint based rule system is used for computing the corresponding temporal relations, making use of the way a multi-context system works for transporting the needed information between contexts and knowledge bases.

1 Point of Departure and Problem Statement

Introducing new modalities should involve no more fuss than introducing a new predicate. [1]

The above quotation, taken from John McCarthy's paper "*Modality, Si! Modal Logic, No!*", addresses a point quite well-known to description logicians and people working with systems for knowledge representation, based on DL. Real world applications often demand for the use of modalities (e.g. to express statements about time), or using McCarthy again:

In particular, human-level AI requires that programs be able to introduce modalities when this is appropriate, e.g. have function taking modalities as values. [1]

Nevertheless, allowing for the unrestricted use of modal operators within a description logic framework (given the not very probable case that this may be possible), major problems concerning decidability, completeness and computability arise. Even if only some modal operators shall be incorporated into a standard DL framework (as e.g. $SHOIN(D)$, corresponding to the widely used OWL DL V1.0 language, or $SROIQ(D)$ for OWL DL V2.0), a large part of the "pleasant" properties of the original formalism gets lost.

Thus, we decided to build a working and applicable implementation of modalities (in the following we will use time as an example modality) without the use of modal operators, thereby following the basic idea lined out in McCarthy's paper. We managed to do so by using concepts from the field of multi-context systems (MCS), mainly based on the results presented in [2] and [3].

2 Attempt at a Solution

The idea underlying our solution to the stated problem is the following: Knowledge bases in DL normally contain definitions of concepts, roles and instances, stated in present tense. For formal reconstruction purposes within many different domains, e.g. cultural heritage, descriptions of instances which are situated in the past would be needed. But modal logical extensions of DL often carry – besides the problem of (un)decidability – new operators (e.g. "Until") with them. In many cases, a description not relying on the use of modal operators would be sufficient. Moreover, the modellers wouldn't have to learn how to use the new operators properly.

In order to enhance the widely used DL $\mathcal{SHOIN}(\mathcal{D})$ (OWL DL V1.0) with these kind of descriptions, in a reference ontology for cultural heritage (ECRM) Allen's time relations ([4]) are added as descriptions (as already demanded by the CIDOC-CRM[3]). Having done so, the descriptions are contained, but a mechanism performing calculations on this modalities is still missing. This mechanism may be introduced by means of an external module, a constraint based rule system (CBRS).

3 Mode of Operation

Given a DL knowledge base, the functionality of our system may basically be sketched as follows:

1. Extend the concrete knowledge base: Collect time describing features within the DL descriptions and state them explicitly.
2. Transfer the collection of time describing features to a constraint based rule system (CBRS), already containing abstract rules which model Allen's time relations.
3. Generate a model by means of the CBRS and afterwards extract the newly calculated, concrete time relations from the model.
4. Inject the extracted concrete time relations into the DL knowledge base.

Thus the result of the external calculations in the CBRS may be used for the proper reasoning within the DL knowledge base. Moreover, this allows – besides of the possibility of the description of the instances in the past – for an automatic sorting of the time descriptions on a timeline given by Allen's relations.

Furthermore, the approach is very flexible, as the means for calculations and computation may be extended: The descriptions in the ontology may be expanded with more relations, given the corresponding bridge rules to the CBRS. Thereby, the limitations of the description are not caused by the restrictions

[3] CIDOC, the Committee on Documentation of the International Council of Museums, is a working group focusing on the documentation requirements and standards of museums, archives, and similar organizations. CIDOC has defined a *Conceptual Reference Model*, the CIDOC-CRM, which provides a formal and extensible ontology for cultural heritage information.

from the *SHOIN*(D) language, but are given according to the configurable external logic. As this logic may be configured and extended stepwise, exactly those modalities needed for a concrete formal reconstruction may be implemented. Hence, in place of making necessary an entire extension of DL, constructed around one modal operator or another, our approach allows for the incremental construction of a TimeDL.

This extension may be transferred to other formal reconstructions wit relative ease, and can also be applied to other DLs. Only the quite simple DL S is needed when describing Allen's temporal relations. Thus re-usability within other DLs, build upon S (as e.g. *SROIQ*, corresponding to OWL DL V2.0) may easily be reached.

Further independence between the extension and the underlying knowledge representation formalism has been obtained.

4 (Very) Short Introduction to Multi-Context Systems

Here we restate the key definitions given in [2]. First, the concept of *logic*[4] is defined in terms of input-output conditions.

Definition 1. *A logic* $L = (\mathbf{KB}_L, \mathbf{BS}_L, \mathbf{ACC}_L)$ *is composed of the following components:*

1. $\mathbf{KB}_L$ *is the set of well-formed knowledge bases of* L*. We assume each element of* $\mathbf{KB}_L$ *is a set.*
2. $\mathbf{BS}_L$ *is the set of possible belief sets,*
3. $\mathbf{ACC}_L : \mathbf{KB}_L \mapsto 2^{\mathbf{BS}_L}$ *is a function describing the "semantics" of the logic by assigning to each element of* $\mathbf{KB}_L$ *a set of acceptable sets of beliefs.*

Given several logics, bridge rules are used to translate between the logics.

Definition 2. *Let* $L = \{L_1, \ldots, L_n\}$ *be a set of logics. An* L_k *-bridge rule over* $L, 1 \leq k \leq n$*, containing* m *conditions, is of the form*

$$s \leftarrow (r_1 : p_1), \ldots, (r_j : p_j), \mathbf{not}(r_{j+1} : p_{j+1}), \ldots, \mathbf{not}(r_m : p_m) \tag{1}$$

where $j \leq m$*,* $1 \leq r_k \leq n$*,* p_k *is an element of some belief set of* L_{r_k} *and* s *is a syntactically valid element of a knowledge base from* $\mathbf{KB}_k$*,*[5] *and for each* $kb \in \mathbf{KB}_k : kb \cup \{s\} \in \mathbf{KB}_k$*.*

A configuration of logics and bridge rules comprises a multi-context system.

[4] As in the following, no information containing satisfiability or inference rules within the corresponding logics will be given, also the denomination "pre-logic" would be justifiable. For the sake of consistency we will keep the original naming calling it a "logic".

[5] In contrast to [2] where a similar constraint concerning the nature of s is imposed only implicitly.

Definition 3. *A multi-context system* $M = (C_1, \ldots, C_n)$ *consists of a collection of contexts* $C_i = (L_i, kb_i, br_i)$*, where* $L_i = (\mathbf{KB}_i, \mathbf{BS}_i, \mathbf{ACC}_i)$ *is a logic,* kb_i *a knowledge base (an element of* $\mathbf{KB}_i$ *), and* br_i *is a set of* L_i*-bridge rules over* $\{L_1, \ldots, L_n\}$*.*

A belief state is the combination of the belief sets of all contexts of the MCS.

Definition 4. *Let* $M = (C_1, \ldots, C_n)$ *be a MCS. A belief state is a sequence* $S = (S_1, \ldots, S_n)$ *such that each* S_i *is an element of* $\mathbf{BS}_i$*.*

We say a bridge rule r of form (1) is applicable in a belief state $S = (S_1, \ldots, S_n)$ iff for $1 \leq i \leq j : p_i \in S_{r_i}$ and for $j+1 \leq k \leq m : p_k \notin S_{r_k}$. A belief state is an equilibrium if the consequences of all applicable bridge rules are given, hence each context has an acceptable belief set given the belief sets of the other contexts.

Definition 5. *A belief state* $S = (S_1, \ldots, S_n)$ *of* M *is an equilibrium iff, for* $1 \leq i \leq n$*, the following condition holds:*

$$S_i \in \mathbf{ACC}_i(kb_i \cup \{head(r) | r \in br_i \text{ applicable in } S\}).^6$$

Now, we moreover introduce "bridge rule models" as a possibility to expatiate on the actual reasoning by signalizing explicitly which bridge rules within an MCS are active with respect to a given belief state (for further details vide [5]).

Definition 6. *Let* $\mathbf{Br}$ *be a set of* n *bridge rules of an MCS. A bridge rule model is an assignment* $\mathbf{Br} \mapsto \{0,1\}^n$ *that represents for each bridge rule in* $\mathbf{Br}$ *whether it is active (i.e. the bridge rule is applied, and the element in its head is added to the respective target knowledge base) or not.*

Proposition 1. *For each equilibrium there is exactly one bridge rule model.*

Proof. Given an MCS and an equilibrium belief state. Then for every bridge rule we can decide whether or not its prerequisites are satisfied. Hence, for each position in the bridge rule model we can decide whether the value is 0 or 1.

5 MCS and Museum Data Completion and Consistency

The now presented MCS, implementing the algorithm for finding the equilibria of an MCS from [3], has been developed in cooperation with part of the team of the WissKI research project[7].

[6] Please note that, if r is a bridge rule of the form $a \leftarrow \ldots$, then $head(r) = a$

[7] Research project "WissKI - Wissenschaftliche KommunikationsInfrastruktur", funding provided by the German Research Council (DFG). The project is being carried out at the Chair for Computer Science 8: Artificial Intelligence of the University of Erlangen-Nuremberg, together with collaborators from the GNM Nuremberg and the ZFMK Bonn. For a presentation of the project vide [6] and http://www.wiss-ki.eu.

The WissKI project's main purpose is to extend the the current conception of wiki-style media to a medium of science communication and scientific interaction. Amongst its main goals are therefore to enable semantic content analysis by means of CIDOC-CRM (ISO 21127) (e.g. vide [7] and [8]). One of the tools under development in the project is a semantics enhanced content management system (CMS+S), in which we integrated a special form of an MCS in order to obtain additional temporal reasoning functionality for the semantic analysis.[8]

We accomplished this by using a form of inference fusion: Starting from a DL based initial representation of data, we transfer parts of it (containing temporal information) to another formalism. Then, we use smodels[9] as a reasoner in order to obtain additional results (in concrete for – if possible – establishing an ordering within the expressed time statements), and finally combine the results of the reasoner and the original data into another enhanced DL representation.

The scenario where the MCS comes into play within the CMS+S may be sketched as follows: We are given a CIDOC-CRM conform file (OWL/DL format, vide [10]) containing information from the context of master pieces of goldsmith art (vide [11]), which also contains – in the form of free text – temporal information about persons and events. An already implemented software tool preprocesses the free text for the MCS: It parses the free text parts, identifies, and then returns place names, person names and time specifications (again vide [11]). Our MCS takes the time specifications (represented as time intervals using Turtle file format[10]) as inputs, creates bridge rules for the transport of all the given statements to a representation suitable for reasoning with smodels,[11] and afterwards uses smodels and given constraints on temporal reasoning in order to create a linear ordering of the given time statements (the ordering of the intervals is established according to [4]). If a linear ordering can be established (if this cannot be done the set of time statements is inconsistent), after ordering the time statements, again bridge rules are created to transport the ordering information to a knowledge base in OWL/DL format, using the CIDOC-CRM

[8] The implementation of the CIDOC-CRM the WissKI project is working on is based on a logical formalism equivalent to a $\mathcal{SHOIN}(\mathcal{D})$ DL. Therefore, temporal reasoning is not supported by the used logical formalism itself.

[9] An implementation of the stable model semantics for logic programs, vide e.g. [9].

[10] For details concerning the Turtle file format vide e. g. [12]. A typical Turtle triplet encountered in the implementation example contains as first string an identifier, the second string states a property of the corresponding object, and the third string explicitely states a value, related to this property, in XSD time data format: `http://wiss-ki.eu/ns/tmp/gen_e61_2_N65566 http://www8.informatik.uni-erlangen.de/IMMD8/Services/cidoc-crm/erlangen -crm_090330_5_0_1_TQ.owl#has_primitiveTime ''2009-06-22''`

[11] By this not only extracting the information from the initial context and adding it to a context suitable for reasoning with smodels, but also performing a translation between the OWL/DL language and a suitable representation for lparse (vide [13]) input at the same time, as the segmentation of bridge rules in a head part and a body part – without a constraint restricting the formal languages used in both parts to be the same – allows for this kind of use.

time relations[12] to reproduce the ordering found. Afterwards, the results may be merged with the original CIDOC-CRM conform file and another reasoner may be used (e.g. RACER[13] on the given data) in two ways: to check the consistency of the enhanced knowledge base, or for data completion purposes taking into account the newly obtained time information.

The MCS just described is of special form, we call it a "linear multi-context system", as the information is passed through it in a linear way. The basic principle may be sketched as follows: The parser/tagger software tool creates a Turtle format representation of the time statements, this is kb_1 in context C_1. Then, as all information from kb_1 has to be transported to the smodels context's knowledge base $kb_2 \in C_2$, the bridge rules from kb_1 to kb_2 may automatically be created given kb_1. The reasoning part is done in kb_2 when the import has been completed. Afterwards, all obtained information concerning time relations has to be transported to the OWL/DL representation in context C_3 (containing kb_3), the bridge rules may automatically be created, completely covering all corresponding elements of kb_2.[14] When the transport to kb_3 has been completed, the work of the MCS is mainly done, the fusion of kb_3 with the original OWL/DL base in the narrower sense is not part of the MCS. Now the mode of operation of the linear MCS shall be discussed in more detail. Given the Turtle triplets – containing information concerning person names, place names and time statements – the parser/tagger returns as output of his free text analysis, the MCS has to identify and extract the time information by simple structural filtering (due to the chosen output format of the parser/tagger, a strictly syntactical discrimination of the different types of statements is possible, e.g. via the use of regular expressions). The results of this process are written to kb_1, the knowledge base of C_1. Now, the bridge rules for the transport of the time interval information from kb_1 to kb_2 (knowledge base of C_2) have to be created and added to br_2. The bridge rules only have one condition (the element of kb_1 which shall be transported): "$f(a) \leftarrow (1 : a)$", where $a \in kb_1$, $f(a) \in kb_2$ and $f(\cdot)$ a function returning as result a "translation" of a to the smodels formalism. Patterns of the bridge rules requiring generation are given in form of generic bridge rules within

[12] Allen's temporal relations between time intervals are implemented by CIDOC-CRM properties P114 to P120, vide [7].

[13] Vide e.g. [14].

[14] One might be tempted to think about generating all the bridge rules already initially, before starting the MCS procedure, and not dynamically during processing. Unfortunately, this would raise major problems: As no information concerning the ordering within the time statements (computed within kb_2) is ab initio available, in br_3 we would have to create a bridge rule for every possible relation between two time intervals for every tuple of time intervals that may have been transported from kb_1 to kb_2. This would yield thirteen bridge rules for every tuple, out of which only one may in fact be activated in the final equilibrium bridge rule model. Therefore, the number of bridge rule models which (according to [3]) have to be tested for representing the equilibrium of the MCS would be multiplied, substantially worsening the performance of the entire MCS.

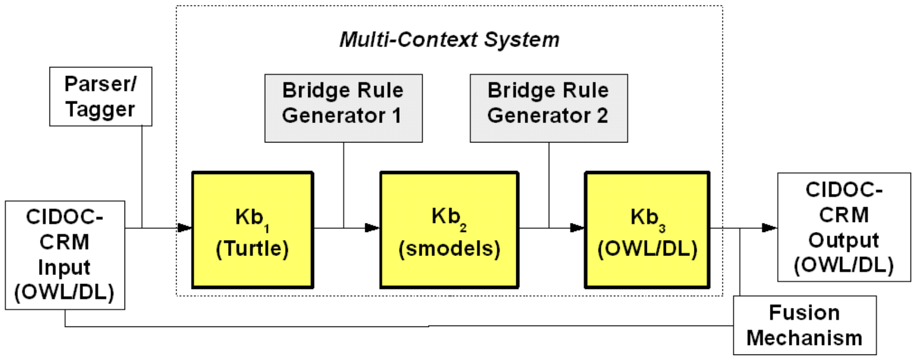

Fig. 1. *A sketch of the design of the linear MCS.*

br_2.[15] For every interval from kb_1, starting time point and ending time point are transported. Having built all bridge rules according to the generic prototypes, we may apply them all at once, as a complete transfer of knowledge from kb_1 to kb_2 shall be performed (the bridge rules have been constructed accordingly). Now, kb_2 is populated with all the temporal information concerning time intervals obtained from the free texts. But kb_2 contains constraints depicting the relations between time intervals that Allen proposed. A reasoner call over kb_2 is performed. The result is again used for bridge rule generation: For each statement indicating the relation between two time intervals, a bridge rule has to be created, transferring this statement to the third context C_3, adding the corresponding CIDOC-CRM statementin OWL/DL format to the knowledge base kb_3. Again, generic bridge rules from br_3 prototypically indicate which bridge rules have to be created.[16] When all of the mentioned bridge rules have been added to br_3, for the same reason as above they may again all be applied at a time. Finally, kb_3 – now containing all the statements concerning the ordering relations amongst the time intervals – is merged with the initial CIDOC-CRM knowledge base (which contains the free texts the parser/tagger originally used), or a single OWL file containing the interval information may be produced.

6 Used Technical Infrastructure

For the implementation of the just sketched MCS we made use of a proof-of-concept MCS Software Framework developed as part of one of the authors'

[15] E.g. $day(X, Y) \leftarrow (1 : day(X, Y))$, indicating that for every element from kb_1, unifyable with $day(X, Y)$ (X an identifier for the temporal entity, Y a numeric value indicating the day's date within the month) a bridge rule shall be generated mapping it to its corresponding element of kb_2.

[16] E.g. "$P114.is\_equal\_in\_time\_to(X, Y) \leftarrow 2 : is\_equal\_in\_time\_to(X, Y)$" ($X$, Y variables for identifiers of time intervals), would create for every fitting element in kb_2, a bridge rule mapping it to its corresponding P114 CIDOC-CRM statement in kb_3.

master's thesis. A detailled overview of the concrete MCS implementation is given in [5]. Therefore, in the following we only want to give an overview of the main technical characteristics of the system used:

As programming language we used SCALA[17] (e. g. vide [15]), a general purpose programming language. The advantages of SCALA, apart from allowing for both object-oriented and functional programming, are its full interoperability with native JAVA code (JAVA may be directly called from SCALA and vice versa), the full byte code compatibility - making possible the full use of existing JAVA libraries or application code - and the possibility to run SCALA programs on the widespread JAVA VM.

Moreover, as previously mentioned, we made use of the lparse/smodels combination as implementation of the stable model semantics for logic programmes, which are both freely available to the scientific community. Both, the lparse front-end and smodels itself, are at some points called as external components by the MCS framework.

For the smodels reasoning, we set up a constraint based implementation of Allen's Interval Algebra, allowing smodels to compute an Allen-like ordering within the time intervals whenever possible (vide Figure 2).

The modality computing MCS has then been integrated into the already existing WissKI software system,[18] placing it in a line with the aforementioned parser/tagger used for performing the free text analysis and production of the MCS's input Turtle triplets. Having computed the linear ordering within the time intervals, the output of the MCS is being merged with the remaining data in OWL/DL format, serving data enrichment and completion purposes.

7 Why Use an MCS – Enhanced Scenarios

Up to now, critics of this approach may question why to use an MCS for this purpose, because as seen from some angles a more or less elaborately written script might provide almost the same functionality. But the scenario just shown is only the first step of evolution of this kind when using an MCS: We conceive by far more complex systems, not being linear, but containing at least one feedback loop from C_3 to C_1, e.g. testing the fusion of C_3 and the original OWL/DL base for consistency, and modifying the knowledge base $kb_1 \in C_1$ if any inconsistency is detected. Performing a run of the MCS excluding in a systematical manner elements from kb_1 from being promoted to kb_2, we might possibly identify the causes for the inconsistency.

[17] We compiled the MCS framework implementation with "`Scala version 2.7.5 final (Java Hotspot(TM) Client VM, Java 1.6.0_15)`".

[18] For more details again vide e.g. http://www.wiss-ki.eu.

```
                  ...
    month_leq(X,Y) :- month(X,A), month(Y,B), A ≤ B, X ≠ Y.
    month_leq(X,Y) :- not month(X,_), month(Y,B), starting_time_primitive(_,X).
    month_leq(Y,X) :- not month(X,_), month(Y,B), ending_time_primitive(_,X).
     month_eq(X,Y) :- month(X,A), month(Y,B), A = B, X ≠ Y.
                  ...
       before(X,Y) :- year_leq(X,Y), not year_eq(X,Y).
       before(X,Y) :- year_eq(X,Y), month_leq(X,Y), not month_eq(X,Y).
       before(X,Y) :- year_eq(X,Y), month_eq(X,Y), day_leq(X,Y),
                        not day_eq(X,Y).
                  ...
       before(X,Y) :- year_eq(X,Y), month_eq(X,Y), day_eq(X,Y),
                        hour_eq(X,Y), minute_eq(X,Y), second_eq(X,Y),
                        millisecond_leq(X,Y), not millisecond_eq(X,Y).
 time_primitive(Y) :- starting_time_primitive(X,Y), year(Y,Z).
 time_primitive(Y) :- ending_time_primitive(X,Y), year(Y,Z).
        equal(X,Y) :- not before(X,Y), not before(Y,X), time_primitive(X),
                        time_primitive(Y), X ≠ Y.
   inconsistent(X) :- starting_time_primitive(X,A), ending_time_primitive(X,B),
                        before(B,A).
                   :- inconsistent(X).
     finishes(X,Y) :- X ≠ Y, starting_time_primitive(X,A),
                        ending_time_primitive(X,B), starting_time_primitive(Y,C),
                        ending_time_primitive(Y,D), equal(B,D), before(C,A),
                        not inconsistent(X), not inconsistent(Y).
is_finished_by(X,Y) :- X ≠ Y, starting_time_primitive(X,A),
                        ending_time_primitive(X,B), starting_time_primitive(Y,C),
                        ending_time_primitive(Y,D), equal(B,D), before(A,C),
                        not inconsistent(X), not inconsistent(Y).
                  ...
 occurs_during(X,Y) :- X ≠ Y, starting_time_primitive(X,A),
                        ending_time_primitive(X,B), starting_time_primitive(Y,C),
                        ending_time_primitive(Y,D), before(C,A), before(B,D),
                        not inconsistent(X), not inconsistent(Y).
                  ...
```

Fig. 2. *Parts of a constraint base for computing Allen's time interval relations.*

```
**Input**:
$tagger\_output : List[String]$ (containing strings with data triplets in Turtle format),
$MCS = (C_1, C_2, C_3)$ ($kb_2 : List[String]$ containing the smodels constraint rules
for temporal reasoning, $br_2 : List[String]$ and $br_3 : List[String]$ each containing
generic bridge rules,
$kb_1 : List[String] = kb_3 : List[String] = br_1 : List[String] = \emptyset$).
**Output**:
$MCS\_output : List[String]$ (containing the relations between time intervals).
$br\_model : List[List[Int]] \leftarrow \{\}$
$kb\_buffer : List[String] \leftarrow \{\}$
$kb_1 \leftarrow extractTimeInformation(tagger\_output)$
for $generic\_br \in br_2$ do
    for $element \in kb_1$ do
        if $matchesPattern(element, generic\_br)$ then
            $br_2 \leftarrow br_2 \cup instantiateBR(element, generic\_br)$
            $br\_model(C_2) \leftarrow br\_model(C_2) \cup \{\{1\}\}$
    $removeFrom(generic\_br,\ br_2)$
$setAllValuesToZero(br\_model(C_1))$, $setAllValuesToZero(br\_model(C_3))$
$kb\_buffer \leftarrow extractKB(findEquilibria(MCS, br\_model), kb_2)$
for $generic\_br \in br_3$ do
    for $element \in kb\_buffer$ do
        if $matchesPattern(element, generic\_br)$ then
            $br_3 \leftarrow br_2 \cup instantiateBR(element, generic\_br)$
            $br\_model(C_3) \leftarrow br\_model(C_3) \cup \{\{1\}\}$
    $removeFrom(generic\_br,\ br_3)$
$setAllValuesToZero(br\_model(C_1))$, $setAllValuesToZero(br\_model(C_2))$
$kb\_buffer \leftarrow extractKB(findEquilibria(MCS, br\_model), kb_3)$
$MCS\_output \leftarrow addHeaderEtc(kb\_buffer)$
return $MCS\_output$
```

Algorithm 1: *The linear MCS for the CMS+S.*

Example 1. Given an MCS $M = (C_1, C_2, C_3)$, where C_1 and C_3 are DL contexts, and C_2 is a temporal logic context.[19] Moreover, initially $kb_1 = \{david,\ goliath,\ abraham,\ lifetime(abraham, 150-200),\ lifetime(david, 175-225),\ lifetime(goliath, 205-250),\ is\_father\_of(abraham, david),\ is\_son\_of(david, goliath)\}$.[20]

Now, assuming that the "$is\_father\_of(X,Y)$" and the "$is\_son\_of(X,Y)$" relations have properly been modelled (i. e. also stating conditions on the relation between the lifetimes of father and son, e.g. that the lifetime of the son may not

[19] For the sake of readability and to avoid unnecessary complications, in the example we will not use the Turtle and OWL syntax, but a more intuitive and easily accessible notation.

[20] $is\_father\_of(X,Y)$ stating that X is the father of Y, and $is\_son\_of(X,Y)$ analogously stating that X is the son of Y.

begin before the lifetime of the father begins), performing a run of the MCS – analogously to the description of the linear MCS above – we would obtain an inconsistency in the belief set corresponding to C_3, as for the lifetimes of "*david*" and "*goliath*" – according to Allen's relations between time intervals – "$lifetime(david, 175-225)$ $overlaps$ $lifetime(goliath, 205-250)$" would be stated. This contradicts the fact that "*david*" is declared a son of "*goliath*".

Excluding the "$is\_son\_of(david, goliath)$" statement from kb_1, we would obtain a consistent belief state, stating an equilibrium of the MCS. Thus, "$is\_son\_of(david, goliath)$" has been identified as possibly causing an inconsistency within the data and should be reviewed.[21]

A related application would be a series of MCS calls for data completion purposes, after each complete call using the newly obtained data for another run of the MCS, until no further augmentation of information may be obtained.

Another but far more sophisticated way of using this type of MCS would be an application in combination with the symbolic sub-symbolic integration proposed in [3]: A "modality + sub-symbolic MCS" could e.g. be used to combine person recognition systems (e.g. based on neural networks) at airports, train stations and street cameras with databases for train, flight and bus timetables etc., making possible tracing, tracking and verification of a person's movement pattern on a wide-area basis.

8 Future Prospects and Conclusion

To the best of our knowledge, the implemented functionality is quite innovative and enriches the CMS+S system with a very attractive feature: the possibility to also perform temporal reasoning. Normally, the underlying description logic by itself does not offer this possibility, but a special temporal extension to the DL must be used. With our approach, the original DL may stay untouched, and moreover the applied "temporal logic" must not be fixed, but may be extended or modified according to individual needs and thus becomes highly modularisable, as additional inference rules or entire logic formalisms may be added to the temporal logic context without modifying the description logic parts of the MCS.

As next step we see an examination of the extendability of the used concept to other modalities then time, e.g. place or possibility.

Moreover, the implementation of the enhanced functionality sketched in Sect. 7, offering functionality for the diagnosis of possibly implicit inconsistencies in the DL knowledge base, as well as for data completion purposes, is one of the main topics on our agenda.

[21] Also the lifetimes of "*david*" and "*goliath*" would be possible reasons for the inconsistency. Which proposal for the source of error to start with in the reviewing process has to be decided application specific, or even some kind of "hypothetical reasoning" handling alternate versions of the knowledge base may be performed (using the corresponding bridge rule models as identifiers and basis of the alternate possibilities of knowledge base evolution).

9 Acknowledgements

We want to thank G. Goerz, S. Mandl and M. Scholz for their cooperation and for the valuable discussions concerning the topics adressed in this paper, as well as the anonymous reviewers for their helpful comments and advice.

References

1. McCarthy, J.: Modality, Si! Modal Logic, No! Studia Logica **59**(1) (July 1997)
2. Brewka, G., Eiter, T.: Equilibria in Heterogeneous Nonmonotonic Multi-Context Systems. In: Proceedings of the National Conference on Artificial Intelligence. Volume 22., Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2007) 385–390
3. Besold, T.R., Mandl, S.: Integrating Logical and Sub-Symbolic Contexts of Reasoning. In Filipa, J., Fred, A., Sharp, B., eds.: Proceedings of ICAART 2010 - Second International Conference on Agents and Artifcial Intelligence, INSTICC Press (2010) . For a full version of the paper vide also *http://www8.informatik.uni-erlangen.de/inf8/Publications/bridging_mcs_original.pdf.*
4. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11) (1983) 832–843
5. Besold, T.R.: Theory and Implementation of Multi-Context Systems Containing Logical and Sub-Symbolic Contexts of Reasoning. Master's thesis, Department of Mathematics & Department of Computer Science 8: Artificial Intelligence, FAU Erlangen-Nuremberg (2009) . The full thesis is available under http://www.opus.ub.uni-erlangen.de/opus/volltexte/2010/1587/.
6. Lampe, K.H., Krause, S., Hohmann, G., Schiemann, B.: Wissen vernetzt: Vom Wandel der Dokumentation in Museen der Natur- und Kulturgeschichte. KI - Künstliche Intelligenz **4/09, Special Issue on "Cultural Heritage and A.I."** (2009)
7. Crofts, N., Doerr, M., Gill, T., Stead, S., Stiff, M.: Definition of the CIDOC Conceptual Reference Model (Version 4.2.4). (March 2008)
8. Doerr, M.: The CIDOC Conceptual Reference Model: an ontological approach to semantic interoperability of metadata. AI Magazine **Vol. 24(3)** (2003) 75–92
9. Simons, P., Niemelá, I., Soininen, T.: Extending and implementing the stable model semantics. Artif. Intell. **138**(1-2) (2002) 181–234
10. Goerz, G., Oischinger, M., Schiemann, B.: An Implementation of the CIDOC Conceptual Reference Model (4.2.4) in OWL-DL. In: Proceedings of the 2008 Annual Conference of CIDOC - The Digital Curation of Cultural Heritage. (2008)
11. Goerz, G., Scholz, M.: Content Analysis of Museum Documentation with a Transdisciplinary Approach. In: Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education. (2009)
12. Beckett, D., Burners-Lee, T.: Turtle - Terse RDF Triple Language. http://www.w3.org/TeamSubmission/turtle/ (January 2008)
13. Syrjnen, T.: Lparse 1.0 User's Manual. (2000)
14. Haarslev, V., Möller, R.: Description of the RACER System and its Applications. In: Proceedings International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August. (2001) 131–141
15. Odersky, M.: Report on the Programming Language Scala. http://lampwww.epfl.ch/scala/ (2002)