What Happened to Bob?

Semantic Data Mining of Context Histories

Michael Wessel[1], Marko Luther[2] and Ralf Möller[3]

[1] Racer Systems GmbH & Co. KG, Hamburg, Germany
[2] DOCOMO Euro-Labs, Munich, Germany
[3] Hamburg University of Technology, Hamburg, Germany

Abstract. We report on event recognition within the life logging application IYOUIT for the automatic creation of static diary-like Blogs. Based on the qualitative context histories produced by IYOUIT, we developed pragmatic event modeling and recognition techniques using technology available today. Our approach combines Description Logics with queries and rules to model event recognizers in terms of context ontologies and Allen's temporal interval algebra. We found the ability to efficiently compute Allen relations between events to be crucial for the performance and scalability of the whole approach. Therefore, we evaluate a set of modeling alternatives and give some practical guidance.

1 Introduction

Social Web applications like Facebook and Twitter allow people to socialize over a distance by sharing notes, photos and other personal information with their buddies or the public. By automating the data gathering via the mobile phone, the IYOUIT[4] system is realizing a sophisticated life logging platform [1]. The mobile application gathers context data by the multitude of sensors available on modern handsets (e.g., GPS, accelerometer,...) and transmits it to components in the network. These components extend the given context streams by making use of external data sources (e.g., mapping locations to weather information) and internal computations (e.g., detecting important places). Eventually, context data is transformed into status updates, like *listening to music* or *just arrived home*, which are distributed via the mobile application or any of the connected Web 2.0 services like Facebook.

To increase the level of abstraction, IYOUIT uses knowledge formalized as OWL DL ontologies to classify qualitative context w.r.t. situation concepts [2, 3]. For example, a *business meeting* may be derived based on the people in proximity, their social relations and the actual place. While the derived (static) situations allow for the generation of more meaningful status updates, the navigation in context histories requires an event model based on situation changes. In a previous work, we exploited simple ontology-based event recognition with Description Logic (DL) [4]. The work reported here improves this initial approach by providing additional pragmatic solutions for DL-based event recognition, solving some of the encountered scalability issues.

We are assuming familiarity with basic DL notions and Semantic Web query languages [5]. Those notions will be used without formally introducing them.

[4] `www.iyouit.eu`

2 Situational Reasoning

Previous work introduced the notion *situation* [3] as a vector of context attribute-value pairs describing the circumstances of a person: $(CA_1 : CV_1, \ldots, CA_n : CV_n)$, where CA stands for context attribute, and CV for context value concept. A situation was represented as an ABox individual sit and an assertion such as

$$sit : (\exists CA_1.CV_1) \sqcap \cdots \sqcap (\exists CA_n.CV_n)$$

There is one OWL ontology for each CA[5], structuring the possible CVs in a taxonomy, e.g., for $CA = at\_place$, we might have $CV \in \{home, office, restaurant, \ldots\}$. The CAs are not necessarily functional (e.g., the $near\_by\ CA$).

In order to recognize a situation, defined concepts are exploited, e.g., a *business meeting* could be detected with a defined concept such as $business\_meeting \doteq\ \geq_3 near\_by.colleague \sqcap \exists at\_place.office$. Such concepts are called *recognizer concepts* in the following. Whereas simple situations can be recognized in the ABox by means of recognizer concepts, exploiting the standard *instance realization service*, we prefer a slightly different representation of situations in the ABox, as in

$$\{sit : situation, (sit, val_1) : CA_1, val_1 : CV_1, \ldots, (sit, val_n) : CA_n, val_n : CV_n\}$$

because this *more explicit* representation allows us to exploit a – concerning the *relational expressivity* – much more powerful recognition mechanism, namely *grounded queries and rules*.[6] Due to the explicit individuals, a standard grounded conjunctive query [6–9] is then sufficient to detect those business meetings in which at least three people are involved which mutually dislike each other:

$$\begin{aligned} ans(x) \leftarrow\ & business\_meeting(x), near\_by(x, y), near\_by(y, z), near\_by(z, x), \\ & hates(x, y), hates(y, z), hates(z, x). \end{aligned}$$

Such a query (rule) is called a *recognizer query (rule)*, and it is well-known that it is not possible to recognize such situations by means of concepts in standard DLs or OWL, due to a lack of relational expressivity. These queries (rules) can be written in SPARQL, SWRL, SQWRL [10] ... We are using NRQL [9].

A NRQL rule is basically a NRQL query, but the query head predicate ans is replaced by a so-called *generalized ABox,* which is an ABox whose assertions may reference variables from the body of the rule. The rule conclusion $hateful\_meeting(x)$ would add the concept assertion $sit : hateful\_meeting$ given the binding $x = sit$ satisfies the rule body. Similar effects can be achieved in SWRL or SPARQL (using `construct`). But unlike SWRL or SPARQL, NRQL also allows to introduce new ABox individuals. This feature will be exploited in the following. Even more important in our scenario, NRQL offers some form of *epistemic first-order queries* [11]. Universal closed domain quantification can be expressed by combining the *negation-as-failure (NAF) operator* and the *projection operator*. This expressivity is indispensable here.

We call the IYOUIT component which creates an ABox representing the accumulated context data per day the *Day Description Generator (DDG)* and the generated ABox the *Day Description ABox (DDA).* The DDG is an external program outside of

[5] We are using distinct CA ontologies instead of one big ontology for reasons of modularity.

[6] A query can be considered as a special rule whose conclusion consists of a single ans head predicate.

the DL system. Whereas in many cases, the value concept is explicitly asserted via $val_i : CV$, sometimes ABox realization and thus inference is required to recognize the CVs. To facilitate this kind of CV inference, the DDA not only contains situation descriptions, but also user profile data, e.g., the social network of a user, as well as other mostly static information, e.g. home country, typical office hours, etc.

3 Event Recognition with Description Logics

In [4] we shifted the focus from recognizing static situations which are conceptually instantaneous "snapshots in space-time" to so-called *events* which also take the dynamic and temporal aspects of situational changes into account. Events have a temporal duration. Recognized structures can be exploited for the creation of static diary-like day summaries (Blogs), but also for querying and data mining purposes. As such, the famous *Allen's temporal relations* [12] play an important role, not only for query formulation and natural language generation for Blogs, but also as vocabulary for defining event recognizers. By enabling IYOUIT to recognize situational changes rather than just static situations, a deeper level of context awareness can be achieved. For example, an *ordinary office day* event could be defined as a sequence of consecutive events, which stand in the Allen relation "meets": ⟨at home, moving, in office, moving, in restaurant, moving, in office, moving, at home⟩, together with some additional restrictions regarding certain fixed day time intervals, such as "in restaurant *during* noon", etc. Thus, the DDA also contains these day time intervals. An ordinary office day is therefore a complex event aggregating a series of subevents and should thus be recognizable from the types of its subevents and from the temporal relations holding between them. The challenging questions are:

a) How to represent events?
b) How to define event recognizers?
c) Where do events come from?

Note that we are only interested here in technology that builds on existing DL or OWL reasoners (such as RACERPRO) to get a working system with good performance today. Regarding a), the least "ontological commitment" we can make is saying that we want to represent actual events in the ABox which satisfy the following axiom:

$$event \doteq \exists start\_state.state \sqcap \exists end\_state.state$$

We simply state that an event is an "aggregate" which has a start state and an end state, and that $state$s are basically the situations just described, but augmented with some temporal information which allows to determine the temporal order between states. We assume a linear discrete time model, e.g. $(\mathbf{N}, <)$. There are various options for the representation of the temporal relations between the states; these options will be discussed in the next section. It is also assumed that $start\_state < end\_state$ holds, for all events. In addition, *complex events* are composed of (one or several) subevents:

$$complex\_event \doteq event \sqcap \exists has\_part.event$$

Non-complex events are called *simple events.* Two important specializations are *change events* and *constancy events*. The former witnesses a change of some CV value of some CA from $start\_state$ to end_state, e.g., if the value of $CA = at\_place$ changes from

$CV = home$ to $CV = office$, then this change will be reflected by a $home2office$ change event. The witnessing end_state should be the smallest successor w.r.t. the temporal order $<$. Hence, a change event should have a minimal temporal duration. In contrast, a *constancy event* should have maximal duration, and must also be homogeneous [13], i.e., there may be no state in between the start and end state for which the $CA \times CV$ attribute value pair does *not* hold. An example is the $staying\_in\_the\_office$ event, which is a constant and thus *maximal and homogeneous* event. These event properties are very important, since otherwise the definition of more complex events based on subevents becomes infeasible if one never knows in how many "segments" a subevent is split.

By exploiting the epistemic first-order properties of NRQL, maximal and homogeneous events for some $some\_condition$ are recognized by the following query:

$$\begin{aligned} ans(s_1, s_2) \leftarrow\ & state(s_1), state(s_2), future(s_1, s_2), \\ & some\_condition(s_1), some\_condition(s_2), \\ & \backslash\pi(s_1)\ (state(s_0), next(s_0, s_1), some\_condition(s_0)), \\ & \backslash\pi(s_2)\ (state(s_3), next(s_2, s_3), some\_condition(s_3)), \\ & \backslash\pi(s_1, s_2)\ (\ state(s_3), future(s_1, s_3), future(s_3, s_2), \\ & \qquad \backslash some\_condition(s_3)) \end{aligned}$$

We assume that $future$ holds between s_1 and s_2 iff s_1 precedes s_2 on the time line, i.e. $s_1 < s_2$, and $next$ between s_1 and s_2 iff s_2 is the direct successor of s_1 w.r.t. $<$. The $\backslash$ is the NAF operator, and π is the projection operator. The semantics of these operators if formally exemplified in terms of FOPL just below. The $\backslash\pi \ldots$ construction realizes a first-order epistemic closed-domain quantifier. The first $\backslash\pi \ldots$ ensures maximality of the interval to the left, the second occurrence maximality of the interval to the right, and the third occurrence verifies homogeneity, i.e., there is no state s_3 in between s_1 and s_2 for which one cannot prove $some\_condition$, hence, $\backslash some\_condition(s_3)$ shall not hold. Note that replacing $\backslash some\_condition(s_3)$ by $\neg some\_condition(s_3)$ would be too strong, since absence of $some\_condition(s_3)$ does not imply $\neg some\_condition(s_3)$. We refer to [9, 14] for more details.

The semantics of the NRQL query above is given by evaluating the following first-order query over the relational structure $\mathcal{S}_\mathcal{A}$=($\Delta^\mathcal{I}, C^\mathcal{I}, ..., R^\mathcal{I}, ...$), with $\Delta^\mathcal{I} = \mathsf{inds}(\mathcal{A})$, $C^\mathcal{I} = \{\, i \,|\, i \in \mathsf{inds}(\mathcal{A}), \mathcal{A} \models C(i) \,\}$, $R^\mathcal{I} = \{\, (i,j) \,|\, \mathsf{inds}(\mathcal{A}), \mathcal{A} \models R(i,j) \,\}$, for all relevant roles R and all relevant (not only atomic!) concepts C. By construction of $\mathcal{S}_\mathcal{A}$, NAF negation ($\backslash$) can be replaced by classical negation ($\neg$) as follows, and the universal quantifications $\neg\exists \ldots$ stemming from the $\backslash\pi \ldots$ occurrences become apparent:

$$\begin{aligned} \{\, (s_1, s_2) \,|\, \exists s_1, s_2 :\ & state(s_1) \wedge state(s_1) \wedge future(s_1, s_2) \wedge \\ & some\_condition(s_1) \wedge some\_condition(s_2) \wedge \\ & \neg\exists s_0 :\ state(s_0) \wedge next(s_0, s_1)\ \wedge some\_condition(s_0) \wedge \\ & \neg\exists s_3 :\ state(s_3) \wedge next(s_2, s_3)\ \wedge some\_condition(s_3) \wedge \\ & \neg\exists s_3 :\ state(s_3) \wedge future(s_1, s_3), future(s_3, s_2) \wedge \\ & \qquad \neg some\_condition(s_3)\ \} \end{aligned}$$

Events can be recognized by recognizer concepts, queries or rules. Recognizer concepts solely rely on the ABox individual realization service and thus on explicit ABox individuals and relations between them, which need to be, at least to some extent (modulo inference), explicitly given in the ABox in forms of role assertions, since, with the exception of transitive relations, OWL DL offeres no defined roles. In addition to taking the obvious inferred role assertions into account, a NRQL rule can easly add new ABox

(role) assertion, or even introduce new individuals. Moreover, intensional relation are available for queries and rules by means of so-called *defined queries* which are similar to intensional database relations defined as Datalog rules. For example, the binary recognizer rule for the Allen relation *before*

$$before(x, y) \leftarrow event(x), event(y), end\_state(x, s_1), start\_state(y, s_2), future(s_1, s_2)$$

can either be understood as a rule which adds ("materializes") *before* role assertions to the ABox, or as a definition of the query *before* (i.e., a "query macro") which is expanded and replaced by its definition whenever it is referenced in some other query body, e.g., in an event recognizer query or rule.[7] Hence, a query or rule body referencing *before* does not necessarily require explicit *before* role assertions in the ABox, but can work with implicit *before* relations. As a result of the body expansion, the bodies of the referencing rules can get very complex and become demanding for the query optimizer and processor (an exponential blowup is possible). Defined queries must be acyclic in nRQL. If recursion is required, nRQL rules have to be used instead, and a *rule application strategy* has to be provided (see below).

In principle it is possible to define complex event recognizers as defined queries rather than rules. But, as there would be no ABox individual representing the complex event itself, these queries would either be n-ary predicates (returning the n subevents which make up the complex event), or binary predicates such as $ordinary\_office\_day$ (s_1, s_2) on the states s_1 and s_2. The n-ary solution is obviously bad, and the binary solution makes it impossible to refer to the individual subevents – only the states would be available (which are obviously required for the computation of Allen relations between the complex event and the other events). Consequently, also the simple events should be modeled as binary predicates. To sum up, this results in an entirely different modeling and is thus not considered further in this paper. Even Allen relations could no longer be understood as roles holding between events, but would become quadrary defined queries for pairs of endpoints of events.

Summary: In principle, Allen relations as well as the events can be recognized by rules or defined queries. As we want events to be represented explicitly in the ABox, the later option is dissatisfactory for recognizing events. Recognizing Allen relations via defined queries was also already investigated in [4] where we have observed a rather bad performance, which was partly caused by the unavoidable recomputation of relations, and partly by the blowup resulting from unfolding Allen definitions.

Here we decided to materialize Allen relations via rules at the price of larger ABoxes, a decision evaluated further in Section 4. First, however, we want to analyze some potential answers to question c) "Where do events come from?".

3.1 Approach 1: Pre-Constructing Events

The set of states of a given day is finite. Therefore, one might ask whether it is an option to pre-construct all possible events in the DDA and to rely as much as possible on standard ABox individual realization for the classification of events. For n states, there are at most $m_0 = n(n-1)/2$ simple events. Hence, there are at most

[7] This does not work in concepts, e.g., $\exists before. \ldots$ would not be aware of the defined query.

$m_1 = \sum_{2 \leq k \leq m_0} \binom{m_0}{k} = 2^{m_0} - m_0 - 1$ complex events that can be constructed from these m_0 simple events. These events can in turn become subevents of more complex events, and so on. For 3 states, we already get 120 events for $m_0 + m_1 + m_2$, and 1329227995784915872903807060280344455 events on the next level. The set of preconstructible events is in fact infinite, but it may be possible to compute an upper bound based on the definitions of the recognizers. Although it seems reasonable to stop at, say, level 3, still the constructed ABoxes are not manageable. With the more realistic assumption of approx. 30 states per day and user in case of IYOUIT, already level 1 becomes infeasible. Even worse, m_i^2 additional Allen relations have to be asserted.

Since all relevant event aggregates are already explicitly present in the ABox, recognizer concepts become possible:

$$home2office \doteq \exists start\_state.\exists at\_place.home \sqcap \exists end\_state.\exists at\_place.office$$

Unfortunately, certain important event properties, such as maximality and homogeneity, cannot be expressed with concepts. But at least for the basic context attribute values that require no reasoning, these properties can directly be assured and asserted by the DDG.

In most cases, recognizer concepts are insufficient for the representation of complex events. The main limitation is the inability of concepts to describe anything else but tree structures (regarding the role successors). For example, it is impossible to specify that *during* an in_office event, a $meeting\_with\_boss$ event occurred directly before ($meets$ relation) an $meeting\_with\_customer$ event. A modeling attempt:

$$\begin{aligned} &stressful\_office\_day \doteq \\ &\quad \exists has\_part.(\, in\_office \sqcap \\ &\qquad \exists during^{-1}.(\, meeting\_with\_boss \sqcap \\ &\qquad\quad \exists meets.meeting\_with\_customer \sqcap \ldots)) \end{aligned}$$

Obviously, this definition has the defect that it cannot be taken for granted that the $meeting\_with\_customer$ event still takes place during the in_office event. Other attempts to define such a concept will have similar defects. Although it is impossible to fix the start state and end state by means of existentials, the temporal duration of the complex event as well as the Allen relations to other (possibly complex) events are in fact known, since the aggregate was constructed by the DDG which asserted $start\_state$ and end_state. This motivates the has_part role. Without the additional individual representing the complex event, start and end state of the aggregate could not be fixed, and consequently, Allen relations could not be computed. Moreover, one of its subevents would have to act as a proxy for the whole aggregate [15]. This seems inadequate.

Although many temporal constellations between subevents cannot be reliably recognized with defined concepts since a lot of false positives will be detected, some complex events can indeed be realized in that way, e.g., those complex events for whose description a tree-like temporal Allen network is sufficient. Another required provision to minimize the amount of false positives is that one has to *restrict the visibility of Allen relation successors to those events which are part of the same aggregate* – otherwise the traversal of an Allen role assertion R via $\exists R.\ldots$ could lead into an event being subevent of some other complex event, yielding another false positive. However, this forces the DDG to create copies of subnetworks in order to isolate ABox individuals in different events and thus increases the ABox size by an additional order of magnitude. Hence, the approach does not seem reasonable.

Again, a query or rule can help to overcome the expressivity problems:

$$\begin{aligned} \mathit{stressful\_office\_day}(x) \leftarrow\ & \mathit{has\_part}(x,p_1), \mathit{has\_part}(x,p_2), \mathit{has\_part}(x,p_3), \\ & \mathit{in\_office}(p_1), \mathit{meeting\_with\_boss}(p_2), \\ & \mathit{meeting\_with\_customer}(p_3), \\ & \mathit{meets}(p_2,p_3), \mathit{during}(p_2,p_1), \mathit{during}(p_3,p_1) \end{aligned}$$

Summary: For complex events, the pre-construction by the DDG causes two problems. First, the number of precomputed events and Allen relations gets enormous, and second, it is not always possible to define reasonable expressive recognizer concepts for complex events. The latter point is not resolvable unless specialized temporal logics are used, or certain OWL extensions are accepted [16]. For simple events some recognizer concepts can be defined. This is an advantage, since rule management in DL systems has not yet reached the same level of maturity as concept management. Since in general reasoning is required in order to detect the temporal extent of some $CA \times CV$ pair, it becomes clear that first-order facilities are required if recognizer queries (rules) shall rely on properties such as homogeneity. DL-safe rules are sufficient by now, since all events have been pre-constructed.

3.2 Approach 2: Constructing Events on Demand

In contrast to the approach just described, the other extreme is to have no pre-constructed events at all in the DDA, but only states with their temporal relations. As a consequence, the number of recognizer rules gets very large, and rule management becomes an important issue. The first-order capabilities of the NRQL rules are a big advantage in this setting. We already demonstrated how to recognize maximal and homogeneous events. Since only states together with their temporal relations are present in the DDA, non-safe rules which can introduce new event individuals have to be used. NRQL offers the new operator that can be understood as a (FOPL) function symbol. A typical basic event recognizer rule constructing $event$ instances, looks as follows:

$$\begin{aligned} \mathit{start\_state}(\mathit{new}(e,s_1,s_2),s_1), \mathit{end\_state}(\mathit{new}(e,s_1,s_2),s_2), \mathit{event}(\mathit{new}(e,s_1,s_2)) \\ \leftarrow \mathit{state}(s_1), \mathit{state}(s_2), \mathit{future}(s_1,s_2), \ldots \end{aligned}$$

For example, given states $s_1 = i, s_2 = j$, it constructs the ABox assertions $\{e_{i,j} : \mathit{event}, (e_{i,j}, i) : \mathit{start\_state}, (e_{i,j}, j) : \mathit{end\_state}\}$. Non-safe rules are in principle subject to termination problems, since new individuals are introduced (on which rules may fire which introduce individuals, on which rules may fire which introduce individuals, on which ...). In NRQL, this termination problem is delegated to the client program which has to drive the rule application, since NRQL does not offer an automatic rule application strategy. The client program thus has to run a loop of the form "`while (applicable-rules()) {execute-applicable-rules();}`" calling the NRQL rule API functions. Hence, all possible termination problems are caused by the client which runs the loop. Using NRQL's non-monotonic features, even non-safe individual constructing rules can be written in a way which ensures termination.[8] For example, the above rule can disable itself after all constructible events have been constructed by adding the following conjunct to its precondition:

$$\backslash\pi(s_1,s_2)\ (\mathit{event}(x), \mathit{start\_state}(x,s_1), \mathit{end\_state}(x,s_2))$$

[8] Note that this would not be possible in a monotonic rule language such as SWRL.

The need for an external client program driving the rule application can be seen as a drawback. However, recently the MINILISP functional expression language for server-side programming has been added and reached a mature state [17]. It is thus possible to implement a rule application strategy directly on the RACERPRO server. Since MINILISP is termination safe it does not allow unbounded loops. But it allows bounded loops. In case of non-recursive rules, a fixed upper limit on the number of required rule application cycles can be computed from the rule bodies and number of states.

The recognizer rules for complex events look very similar to the $stressful\_office\_day$ example rule already given above. However, its conclusion has to construct a new event individual and also cannot rely on $has\_part(x, p)$ for its parts p in the precondition. Note that for each freshly constructed event, the corresponding Allen relations have to be computed. The universal closed-domain quantifier of NRQL is also important for complex event recognizer rules, since the maximality and the absence of certain other events between two subevents has to be enforced.

From a theoretical perspective, this approach is the cleanest and most powerful one. Only events which have actually been recognized are constructed, in contrast to the previous approach, where also all kinds of meaningless events are pre-constructed. The approach described in this section was also chosen in [4]. As explained above, the bad performance observed was partly caused by Allen relations formulated as defined queries. Having to construct and check homogeneity and maximality even for simple events introduces a big number of universal closed-domain quantifications which are expensive to evaluate. Hence, we are arguing that the DDG should already pre-construct the basic events and take care of these properties whenever it is capable to do so, thus leaving only the hard recognition problems for the reasoner. This approach will be pursued in the following.

Summary: Constructing events with non-safe rules is conceptually very clean. The whole process is driven solely by the ontology (considering rules being part of the ontology). But, this approach does not perform well with current state-of-the-art technology. The number of rules becomes very large and thus issues not well-supported by current DL systems (e.g., rule management) become predominant. Furthermore, Allen relations can no longer be computed in advance by the DDG, but rather have to be recomputed after each rule application cycle.

3.3 Approach 3: The Best of Both Worlds

Having analyzed both extremes, we proceed as follows. The DDG pre-constructs all relevant simple events, and only the complex events as well as those simple events whose recognition requires ontology reasoning are constructed via rules. Note that the Allen relations can already be asserted by the DDG for those pre-constructed events. In addition, Allen relations must be computed dynamically after each rule application cycle.

For every s_1, s_2, CA and CV, the DDG pre-constructs a simple event if and only if $time(s_1) < time(s_2)$, the event is homogeneous, and has a maximal temporal extent w.r.t. the $CA \times CV$ attribute-value pair. In case the recognition of $CA \times CV$ requires reasoning, the event cannot be pre-constructed, and recognition rules (or concepts) have to be written. In the current IYOUIT scenario, recognizer rules are only required for

complex events. The number of simple events is bounded by $|CA||CV|n(n-1)/2$, where n is the number of states. Since all simple events in the DDA are now maximal and homogeneous, it becomes reasonable to attach the CA attributes to the event individuals instead of the states, and use them as primitives instead of the states. For every possible $CA \times CV$ pair, a simple event recognizer concept is defined:

$$event_{CA,CV} \doteq event \sqcap \exists CA.CV.$$

Consequently, the CV taxonomy is automatically inherited to the taxonomy of simple events (something we would not get automatically with recognizer rules, although nRQL is able to compute query and rule subsumption [14]). These auxiliary concepts facilitate the modeling of complex event recognizer rules.

The DDG does not pre-construct change events. Their absence is not really a drawback. They can still be recognized, e.g. the change of CA from CV_1 to CV_2 can be detected with a rule such as

$$\ldots \leftarrow event_{CA,CV_1}(e_1), event_{CA,CV_2}(e_2), meets(e_1, e_2)$$

if required. Still, some code is needed to drive the rule application. The remarks from the previous paragraph apply.

Summary: This approach seems promising and combines the best of both worlds. It keeps the number of rules maintainable, allows to employ defined concepts as recognizer for simple events and does not lose expressivity. As for the previous approach, the efficient computation of Allen relations is very important.

4 Computation of Allen Relations

Since the set of states is fixed, the DDG can precompute the temporal order between any two states by means of role assertions using the roles $next$, $equal$ (with $next \doteq prev^{-1}$), and a transitive superrole $future$, $next \dot{\sqsubseteq} future$, $future^+ \doteq future$, $past \doteq future^{-1}$. Note that the explicit relation representation in terms of $next$ and $equal$ requires $n + n(n-1)/2$ role assertions (and even more if also $prev$, $future$ and $past$ are made explicit in case one wanted to get rid of the TBox axioms for the roles). Alternatively, the DDG can attach a filler of the *concrete domain (CD) attribute* $time$ to each state. A nRQL rule can make the $next$ relation explicit, or a defined query could be used:

$$\begin{aligned} next(s_1, s_2) \leftarrow & < (time(s_1), time(s_2)), \\ & \backslash\pi(s_1, s_2)\ (< (time(s_1), time(s_3)), < (time(s_3), time(s_2))) \end{aligned}$$

The $<$-atom is a so-called *CD constraint checking atom*. While the evaluation of this atom requires concrete domain unsatisfiability checks which may be expensive, there is also a cheaper option in nRQL, a so-called *data substrate query* which can check such constraints on a set of *data literals* much faster by model checking some data substrate structure [18].

In case $next$ is materialized, we also automatically get the relations $prev$, $future$, and $past$, due to the role declarations. However, if the rule is used as a defined query, then some more definitions are required for the other roles, e.g. for $future$ we can simply remove the "$\backslash\pi \ldots$" line from the $next$ definition. The implicit and intrinsic relations between states keep the ABox small in number of role assertions, they are thus worth of

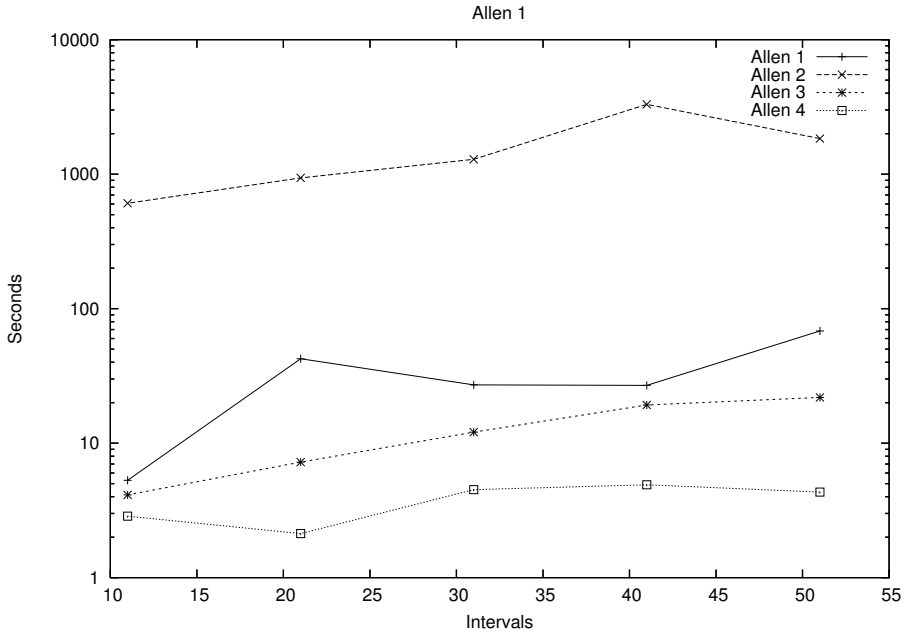

Fig. 1. Benchmark Results

consideration, and should be subject to a benchmark. On the other hand, the explicit relation representation requires bigger ABoxes, but have the benefit to also provide index structures for query answering, and moreover can be used to avoid repeated computations of relations. It is not clear how to proceed without an evaluation of the alternatives. Allen relations are computed from the relations between the states of the events, as already explained in terms of the $before$ relation. The ability to efficiently compute Allen relations is crucial for the performance and scalability of the whole approach. Thus, for the benchmark, we are first evaluating the performance of the Allen computation. A DDA with 50 random intervals is created, which is reduced to 40 intervals by removing 10 intervals, and so on, until only 10 intervals remain. We are considering four different settings for the benchmark:

Allen 1: explicit state relations as $next$ role assertions, 7 Allen rules (one rule per Allen relation and its converse)

Allen 2: implicit state relations via defined query $next$ and CD atom $<$

Allen 3: implicit state relations via defined query $next$ and data substrate atom $<$

Allen 4: implicit state relations and computation of the Allen relations with one rule instead of 7, by means of a MINILISP λ expression which analyzes the values of the $time$ attributes of the states of the events e_1, e_2 and computes the corresponding Allen relation programmatically (with a simple conditional expression $expr$ that we do not explicate here in more detail):

$$\lambda(e_1, e_2) \bullet expr \leftarrow event(e_1), event(e_2)$$

Figure 1 speaks a clear language (note the logarithmic scale). The procedural solution (Allen 4) is by far the fastest with approx. 600 Allen relations per second (ARPS), then comes the data substrate solution with 170 ARPS, then the solution with explicit state relations in the ABox with approx. 66 ARPS, and finally the CD solution with only 1.8 (!) ARPS. Note that the biggest ABox contains 4225 Allen role assertions. The benchmark results make clear that the decision to formulate Allen relations as defined queries based on concrete domains in our previous work [4] was bad.

It is also important to know how much time is spend for Allen computation if a new event is introduced into the existing network. We thus added 5 random intervals into an existing network of 25 intervals and measured the times required for Allen relation computation: Allen 1 = 12.4s, Allen 2 = 612s, Allen 3 = 5.7s, Allen 4 = 1.5s. The bigger the network gets, the more time is required.

5 Conclusion

While lots of temporal logics have been designed [19–24] which could have been applied in this scenario, few of them have been implemented, less have implementations with good performance, and thus, none of these can be used for practical applications today. In contrast, DL systems have made tremendous progress. Still, realizing DL-based event recognition which exhibits good performance is a highly demanding task. There are no simple answers to most of the modeling questions. With nowadays quite complex DL and Semantic Web technology, there are often various realization and representation options. Even for experts, the consequences of certain design decisions can be very hard to oversee. Without performing benchmarks and evaluations, no solid ground can be reached in applying this technology to real world application problems. We argue that case studies like ours are important since they conserve and convey a lot of "how to" knowledge which may prevent users from reinventing the wheel, and make modeling alternatives explicit which is equally important. Although we have used RACERPRO in this work, we argue that the discussed problems and solutions generalize.

In future work, alternative (but implemented) approaches should be checked out, e.g., instead of using unsafe rules, recent progress regarding multimedia and image interpretation with horn rules in an abduction framework shall be exploited [25, 26] where new individuals and corresponding assertions are abducted rather than constructed (a technology brought to a mature state in the BOEMIE[9] EU project). Moreover, in the spirit of the classical event recognition system NAOS [27], a new temporal representation and querying engine, called *time net*, has been integrated into RACERPRO that offers an alternative way to define complex event recognizers. The feasibility of these options for IYOUIT should be evaluated.

Regarding IYOUIT, there are open challenges. To exploit events for the offering of context dependent services on a mobile phone, the events have to be recognized incrementally and online, not offline [28]. As such, the work on incremental plan recognition is relevant [29]. Ideally, this must happen immediately as in [30]. Also there is work in progress regarding the handling of events spanning multiple days, e.g., an oversea business trip. This is challenging since events get "split" at day boundaries and have to be re-merged, etc.

[9] `www.boemie.org`

References

1. Böhm, S., Koolwaaij, J., Luther, M., Souville, B., Wagner, M., Wibbels, M.: Introducing IYOUIT. In: Proceedings of the International Semantic Web Conference (ISWC'08). Volume 5318 of LNCS., Springer, Heidelberg (October 2008)
2. Turhan, A.Y., Springer, T., Berger, M.: Pushing Doors for Modeling Contexts with OWL DL a Case Study. In: Proceedings of the 4th annual International Conference on Pervasive Computing and Communications Workshops, IEEE Computer Society (2006)
3. Luther, M., Fukazawa, Y., Wagner, M., Kurakake, S.: Situational Reasoning for Task-oriented Mobile Service Recommendation. The Knowledge Engineering Review **23**(1) (March 2008) 7–19 Special Issue on Contexts and Ontologies: Theory, Practice and Applications.
4. Wessel, M., Luther, M., Wagner, M.: The Difference a Day Makes – Recognizing Important Events in Daily Context Logs. In: Proceedings of the CONTEXT'07 Workshop on Contexts and Ontologies. Volume 298 of CEUR Workshop Proceedings., CEUR-WS.org (2007)
5. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook – Theory, Implemenation and Applications. Cambridge University Press (2003)
6. Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Description Logic Framework for Information Integration. In Cohn, A.G., Schubert, L.K., Shapiro, S.C., eds.: Proceedings of 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Morgan Kaufmann (1998)
7. Möller, R., Wessel, M.: Terminological default reasoning about spatial information: A first step. In Freksa, C., Mark, D.M., eds.: Proceedings of the International Conference on Spatial Information Theory (COSIT '99). Volume 1661 of LNCS., Springer, Heidelberg (1999)
8. Horrocks, I., Tessaris, S.: Querying the Semantic Web: A Formal Approach. In Horrocks, I., Hendler, J., eds.: Proceedings of the 13th International Semantic Web Conference (ISWC 2002). Volume 2342 of LNCS., Springer, Heidelberg (2002)
9. Wessel, M., Möller, R.: A High Performance Semantic Web Query Answering Engine. In: Proceedings of the International Workshop on Description Logics (DL'05). Volume 147 of CEUR Workshop Proceedings., CEUR-WS.org (2005)
10. O'Connor, M.J., Shankar, R., Musen, M.A., Das, A.K., Nyulas, C.: The SWRLAPI: A Development Environment for Working with SWRL Rules. In: Proceedings of the International Workshop on OWL: Experiences and Directions (OWLED 2008). (2008)
11. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: EQL-Lite: Effective First-Order Query Processing in Description Logics. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI '07). (2007)
12. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. Communications of the ACM **26**(11) (1983) 832–843
13. Artale, A., Bettini, C., Franconi, E.: Homogeneous Concepts in a Temporal Description Logic. In Baader, F., Lenzerini, M., Nutt, W., Patel-Schneider, P., eds.: Proceedings of the International Workshop on Description Logics (DL'94). Number DFKI-D-94-10, DFKI (1994)
14. Haarslev, V., Möller, R., Wessel, M.: RacerPro User's Guide and Reference Manual Version 1.9.0. Technical report, Racer Systems GmbH & Co. KG (May 2005)
15. Haarslev, V., Wessel, M.: GenEd – An Editor with Generic Semantics for Formal Reasoning about Visual Notations. In: Proceedings of the IEEE Symposium on Visual Languages, IEEE Computer Society (1996)
16. Motik, B., Grau, B.C., Sattler, U.: The Representation of Structured Objects in DLs using Description Graphs. In Baader, F., Lutz, C., Motik, B., eds.: Proceedings of the 21st International Workshop on Description Logics (DL'08). Volume 353., CEUR-WS.org (2008)

17. Kaplunova, A., Möller, R., Wessel, M.: Leveraging the Expressivity of Grounded Conjunctive Query Languages. In: Proceedings of the International Workshop on Scalable Semantic Web Systems (SSWS'07). Volume 4806 of LNCS., Springer, Heidelberg (2007)
18. Wessel, M., Möller, R.: Flexible Software Architectures for Ontology-Based Information Systems. Journal of Applied Logic **7**(1) (2009)
19. Artale, A., Franconi, E.: A Temporal Description Logic for Reasoning about Actions and Plans. J. Artif. Intell. Res. (JAIR) **9** (1998) 463–506
20. Grathwohl, M., de Bertrand de Beuvron, F., Rousselot, F.: A New Application for Description Logics: Disaster Management. In Lambrix, P., Borgida, A., Lenzerini, M., Möller, R., Patel-Schneider, P.F., eds.: Proceedings of the International Workshop on Description Logics (DL'99). Volume 22 of CEUR Workshop Proceedings., CEUR-WS.org (1999)
21. Artale, A., Franconi, E.: A Survey of Temporal Extensions of Description Logics. Annals of Mathematics and Artificial Intelligence **30**(1-4) (2001) 171–210
22. Artale, A., Lutz, C.: A Correspondence between Temporal Description Logics. Journal of Applied Non-Classical Logics **14**(1-2) (2004) 209–233
23. Artale, A., Parent, C., Spaccapietra, S.: Evolving Objects in Temporal Information Systems. Annals of Mathematics and Artificial Intelligence **50**(1-2) (2007) 5–38
24. Artale, A., Lutz, C., Toman, D.: A Description Logic of Change. In Veloso, M.M., ed.: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), AAAI Press (2007)
25. Neumann, B., R., M.: On Scene Interpretation with Description Logics. In Christensen, H., Nagel, H.H., eds.: Cognitive Vision Systems: Sampling the Spectrum of Approaches. Volume 3948 of LNCS., Springer, Heidelberg (2006)
26. Espinosa Peraldi, S., Kaya, A., Melzer, S., Möller, R., Wessel, M.: Towards a Foundation for Knowledge Management: Multimedia Interpretation as Abduction. In Calvanese, D., Franconi, E., Haarslev, V., eds.: Proceeding of the International Workshop on Description Logics (DL'07). Volume 250 of CEUR Workshop Proceedings., CEUR-WS.org (2007)
27. Mohnhaupt, M., Neumann, B.: Understanding Object Motion: Recognition, Learning and spatiotemporal Reasoning. Robotics and Autonomous Systems **8**(1-2) (1991) 65–91
28. Luther, M., Böhm, S.: Situation-Aware Mobility: An Application for Stream Reasoning. In Della Valle, E., Ceri, S., Fensel, D., van Harmelen, F., Studer, R., eds.: Proceedings of the International Workshop on Stream Reasoning. Volume 466 of CEUR Workshop Proceedings. (2009)
29. Artale, A., Franconi, E.: Hierarchical Plans in a Description Logic of Time and Action. In Horrocks, I., Sattler, U., Wolter, F., eds.: Proceedings of the International Workshop on Description Logics (DL'95). Volume 147 of CEUR Workshop Proceedings. (1995)
30. Andre, E., Herzog, G., Rist, T.: On the Simultaneous Interpretation of Real World Image Sequences and their Natural Language Description: The System Soccer. In Kodratoff, Y., ed.: Proceedings of the 8th European Conference on Artificial Intelligence (ECAI'88), Pitmann Publishing, London (1988)