Update of time-invalid information in Knowledge Bases through Mobile Agents

Conference or Workshop Item

Update of time-invalid information in Knowledge Bases through Mobile Agents

Ilaria Tiddi, Enrico Daga, Emanuele Bastianelli, Mathieu d'Aquin[1]

Abstract— **In this paper, we investigate the use of a mobile, autonomous agent to update knowledge bases containing statements that lose validity with time. This constitutes a key issue in terms of knowledge acquisition and representation, because dynamic data need to be constantly re-evaluated to allow reasoning. We focus on the way to represent the time-validity of statements in a knowledge base, and on the use of a mobile agent to update time-invalid statements while planning for "information freshness" as the main objective. We propose to use Semantic Web standards, namely the RDF model and the SPARQL query language, to represent time-validity of information and decide how long this will be considered valid. Using such a representation, a plan is created for the agent to update the knowledge, focusing mostly on guaranteeing the time-validity of the information collected. To show the feasibility of our approach and discuss its limitations, we test its implementation on scenarios in the working environment of our research lab, where an autonomous robot is used to sense temperature, humidity, wifi signal and number of people on demand, updating the knowledge base with time-valid information.**

I. Introduction

The focus of this paper is how to update knowledge bases containing statements that lose validity with time. Our problem is that, in many real-world contexts, knowledge bases contain both static information, i.e. statements that will be always valid, and dynamic information, i.e. statements that are only valid for a certain period of time. Dealing with dynamic data constitutes a key issue in terms of knowledge acquisition and representation, because data need to be constantly re-evaluated to allow inference and reasoning.

Let us take the very basic example of a knowledge base representing a working environment, e.g. the Knowledge Media Institute (KMi) research department. Here, information about locations is static, as for instance the coordinates of a room, but dynamic information such as temperature, humidity, wi-fi signal or number of people in a given room changes often, therefore statements about them in the knowledge base might not be valid anymore after some time. For instance, the number of people or the temperature in the seminar room (called "the Podium") varies depending on whether there is a seminar or a meeting going on, and such information should be re-evaluated once the seminar has finished.

Common solutions to this problem include temporal versions of a knowledge base with a time component, e.g. reproducing the KMi knowledge base with new information every day and annotating it with a time-stamp, as well as using sensors in each of the possible locations to be considered. These solutions are naturally more costly in terms of hardware deployment, power consumption and data collection. Also, they are less flexible as they might only allow to query the knowledge base at specific locations where sensors are available. Finally, information might be constantly updated but rarely queried, therefore much of it is likely not be useful. While such approaches might be suitable for a simplified scenario such as the KMi one, their application at large-scale, as for instance in the smartcities environment where our current research is focused, might be more complex.

An alternative solution is to move a sensor upon request, e.g. using a robot, to re-collect the information that is "expired" (i.e. that has lost time-validity), and consequently update the knowledge base, therefore guaranteeing that the knowledge base will return time-valid information when queried. For instance, if we query the knowledge base to know the temperature of the Podium and this is no longer valid, the robot would move and sense the temperature, update the knowledge base accordingly and return a result which is valid in time. In this scenario, our challenges become:

- how to establish if some information is outdated? In other words, how to represent time-validity in the knowledge base?
- how to instruct the robot to perform actions in the right order, so that none of the statements that answer the query will lose time-validity? In other words, how to make a plan that will favor the time-validity of the information that is collected?

To answer these questions, we present an approach to update the time-invalid information of a knowledge base using an autonomous mobile agent on demand, and which relies on Semantic Web technologies as a framework for knowledge representation. More specifically:

- we use the RDF[1] model to represent time-validity and annotate statements in the knowledge base with a time-stamp, which defines the moment in which they will become invalid (i.e. their expiry date);
- we rely on the SPARQL[2] query language to decide how long information will be considered valid, based on a set of time-validity rules;
- we use such representation to design a simple planner that focuses on guaranteeing the time-validity of the information collected by the robot;

[1]Knowledge Media Institute, The Open University, Milton Keynes, MK7 6AA, United Kingdom

[1]https://www.w3.org/RDF/
[2]https://www.w3.org/TR/rdf-sparql-query/

- we instruct the robot to get data at query time and update the knowledge base with the newly sensed information.

Our approach has several advantages. The use of Semantic Web technologies allows us to deal with more flexible schema-less databases and avoid to define specific data model as for temporal databases, therefore making our approach simpler and more reusable. Relying on the RDF/SPARQL paradigm also simplifies the implementation process of detecting the portion of data relevant to answer a given query. Finally, the use of an autonomous mobile agents allows to avoid having to deploy and manage many unnecessary data-flows. To show its feasibility, we apply it in a real-world scenario in which a robot moves and senses information in our department on demand.

II. Related Work

We divide the related work in two sections: first, we analyze works in robotics that have approached the problem of time-constrained planning; second, we discuss how time has been represented and which solutions exists to deal with dynamic data in the area of the Semantic Web.

A. Time-constrained planning for autonomous mobile agents

The use of mobile agents for mobile sensing as an alternative to static sensors is not new in the field of robotics. Many works focused on adapting the traditional planning strategies to dynamic environments, where plans can be aborted based on the circumstances [1]. Autonomous mobile agents have been proposed in Wireless Sensor Networks to collect and update information from static sensors [5], [9], [27]. These works approach the robot routing problem using information from the sensors as the time-constraint (i.e. an ordered sequence to be visited) that allows to minimize transmission and traveling energy. These solutions are however specific and difficult to reuse. A domain-independent planning algorithm based on a heuristic to estimate the completion time of the plan was presented in [12]. However, this as well as others, focus on optimizing the way the space is explored and its coverage so that plans can be repeated periodically, and without interruption. In temporal planning and scheduling, time-constraints have been the focus of timeline-based approaches [8], [15], [18]. These works rely on a temporal representation based on timelines defined by time-points that represent the possible evolutions and changes of individual state variables over time. Temporal constraints in the planning operators are expressed using interval algebra. This planners, however, are not scalable and difficult to write. To cope with that, frameworks such as FAPE [6] and CHIMP [21] propose to hybrid planners based on timeline-based planners combined with hierarchical task decomposition, to be integrated into robotic platforms. The optimization of plans for robots with limited and uncertain data collection time is presented in [26]. While the idea is very similar to the one presented here, only the planning aspect is approached. Rather than concentrating on planning optimization, our work is focused on the representation of the time-validity to execute a plan.

B. Time representation in Semantic Web knowledge bases

Time representation was extensively investigated in the Semantic Web area with the aim of modeling time-changing information and being able to reason about it.

Theoretical frameworks to formally extend the RDF data model to represent time-validity were presented by [10], and then further extended to include time-intervals [22], OWL 2 expressions [14] or indeterminate temporal annotations [17]. These models, however, use a time interval-based representation requiring additional operators, introducing further complexity in the reasoning process. We instead represent temporal information as time-validity, enabling the processing of the knowledge base focusing only on time-valid information, and the creation of data collection plans to update the knowledge base when necessary, at query time.

Point in time semantics has been used to analyze how RDF data change over time, and how to obtain historical records of them. Temporal databases to represent RDF data with a time-stamped validity have been proposed by [7] and [24], while [11] proposes to use named graphs to annotate the different versions of RDF data stores using the PROV vocabulary[3]. The main disadvantage of these solutions is that they can only deal with static queries executed against the dataset once, providing retroactive results. Our aim instead is to identify the portion of the knowledge base that requires to be updated for a given query.

Continuous queries, which guarantee a constantly time-valid information, have been studied in the area of stream processing and reasoning. The work of [20] proposed a vocabulary to describe the spatio-temporal or context semantics for sensors, and for the networks that provide them (e.g. measurement precision or battery level). Several works presented approaches to represent streams as time-stamped RDF triples (see [4], [13], [16], [19]), and the C-SPARQL execution engine was presented in [3] as a support to it. RDF stream reasoning solutions, however, present high technical barriers, such as scalability and low execution throughput due to the ad-hoc operators used to perform data aggregation [16]. Also, they can only be applied to the kind of scenarios considered here if large numbers of continuously active sensors are deployed.

Little work has been done in the Semantic Web area towards the incremental maintenance of large knowledge bases in dynamic environments with frequent changes. This problem was previously approached by [25], that presented a solution to identify, remove and update time-invalid information from domain ontologies. We report the work of [2], which used time-varying graphs allowing information updates in the context of the Streaming Linked Data framework, and of [23], that presents a system to efficiently maintain and update large knowledge bases using parallelization. These works are however focusing on a different aspect of temporal

[3] https://www.w3.org/TR/prov-o/

representations than what is considered here, and are not directly applicable in our scenario.

The related work shows how different ways of representing temporal information exists and which problems derive from each of them. In this work, we propose to maintain a knowledge base up-to-date at query time by using a mobile agent, therefore avoiding unnecessary data collection. We focus on the representation of time-validity and on the way to exploit it as a planning constraint. Before presenting our approach in details, the next section gives an overview of the proposed process.

III. Overview of the process

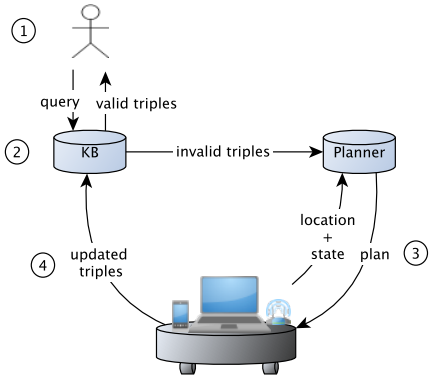

Fig. 1. Update of time-invalid information using a mobile agent on request.

We designed and implemented a process in which a user submits a query to a knowledge base and receives results that are valid in time, provided that there exists a plan that enables the agent to collect the required information in the limited time before it becomes invalid. In the following, we refer to the guide use case of the Knowledge Media Institute, already mentioned in the introduction. We also invite the reader to refer to the experiment section for a map of KMi. The process, shown in Figure 1, is articulated as described below.

1) QUERY. The user expresses the query to the knowledge base. For instance, a user in KMi asks what is the temperature of the Podium before bringing his invitees there.
2) INVALID INFORMATION COLLECTION. In this step, the time-validity of the statements required to compute the query answer is evaluated, and invalid statements (marked "invalid triples" in the figure) are collected. In the previous example, if the information about the temperature in the Podium has lost its time-validity (it has been not sensed for some time), the corresponding statement in the knowledge base is marked as time-invalid and sent to the planner.
3) PLANNING. The planner receives the invalid information and asks the mobile agent for its location. Based on this, it builds the initial state (planning problem) and calculates the plan to send to the agent, i.e. the right sequence of actions to perform so that none of the statements in the answer set remains time-invalid (the planning goal). For instance, given the time-invalid information of the temperature in the Podium, and a mobile agent which is located in the coffee area, the planner will compute a plan of the form `[move_to(Podium), sense(temperature, Podium)]`.
4) KNOWLEDGE UPDATE. The robot receives the plan and performs it. When collecting a new piece of information, it is sent to the knowledge base, which is updated, including calculating the duration of its time-validity. Once all the actions have been executed, the user is shown the answer in which all the involved statements are time-valid, e.g. the freshly sensed temperature in the Podium.

It is worth mentioning that if no answer is received, this means that there is no plan that can be executed so that all the statements in the answer are time-valid. Let us consider the case in which the user asks for the temperature of both the Podium and a room further away (called Room 22). If, by the time the agent has sensed the temperature of the second one, the first information collected is no longer valid, then there is no plan that guarantees time-validity, and such a query needs to be simplified, e.g. executed once per room. In the next section, we present our framework to represent time-validity knowledge, how we used it to evaluate plans for the robot, and how the required updates are performed.

IV. Time Validity and Dynamic Knowledge Base Update

In this section, we detail the various aspects of our approach. First, we describe how we represent and assess the validity of statements in time; second, we detail how such information is managed and validated in the knowledge base; third, we describe how the time-validity of statements used to answer a query is calculated; finally, we show the basic planning mechanism which generates robot plans that are optimal with respect to the time-validity of the information collected through executing them.

A. Representing time-validity in the Knowledge Base

In our scenario, the knowledge base contains both static information, and information that is time-dependent, i.e. which is only valid for a certain period of time. We use Semantic Web technologies, namely RDF graph representation and the SPARQL query language, as a basic framework for knowledge representation.

RDF is a directed, labeled graph representation language connecting resources (entities) and literals of various datatypes. The elementary representation component in RDF is a *triple* representing a link.

Definition 1 (RDF triple): A RDF triple is a unit of representation of the form $\langle s, p, o \rangle$, where s is a resource, p is a resource, and o is a literal or a resource. Resources are identified through URIs[4]. s is generally called the subject of

[4] `https://tools.ietf.org/html/rfc3986`

the triple, p the predicate and o the object. A triple therefore represents a labeled link between the subject and the object where the label is the predicate.

For example, the triples in Figure 2 represent information about the Podium, specifying its type and linking it to its location. To simplify, we will use for the remainder of this article an abbreviated representation for URIs using prefixes, where for example `<http://data.open.ac.uk/kmi/area/Podium>` is replaced by the shorter version `location:Podium`. The main issue we are tackling in this paper is that, while the information represented in Figure 2 is relatively static, other types of information, such as the temperature in the Podium, are very much dependent on time. As discussed in the related work section, there are many ways to represent temporal information, including in RDF.

Here, we focus on the validity of the information, i.e. on representing the fact that some triples might only be valid for a certain period of time (e.g. the temperature in the room is only considered valid for 5 minutes). In our scenario, such validity is represented as a binary-concept; however, other scenarios can be studied to make the information validity a more fuzzy notion, which decays gracefully with time. This is left for future work. We therefore need to be able to associate to each triple in the knowledge base an additional annotation related to the time at which it will become invalid (i.e. its "expiry date").

To achieve this, we rely on the *named graph* feature of RDF, which can be used to divide the general graph into sub-graphs. Effectively, this corresponds to extending RDF triples into RDF quads, where the fourth component is the graph to which the triple belong.

Definition 2 (RDF Quad): A RDF quad is a unit of representation of the form $\langle s, p, o, g \rangle$, where s, p and o are as in Definition 1 and g is a resource (identified by a URI) corresponding to a graph to which the triple $\langle s, p, o \rangle$ belongs.

Our approach therefore relies on creating graphs which URIs are constructed to represent the time at which the triples they include will expire. For example, the following quad:

```
location:Podium robo:hasTemperature
                        "23.4" timegraph:1468917352.
```

represents the fact that the temperature in the Podium room is 23.4 °C and that this information will expire at the time represented by the epoch timestamp 1468917352.

Using the named graphs as meta-level annotations makes the information query-able in the same format as it would without this feature, as a homogeneous triple collection. This makes the choice of quads preferable to other possible RDF representations of time-validity (e.g. the reification the triple), which change the data model from a triple based one to a more complex set of N-ary relations.

B. Updating the Knowledge Base with Time-Validity Information

Having established our approach for representing the time-validity of triples, we need a mechanism to derive this information for any triple added to the knowledge base. To achieve this, we define, as a configuration of the knowledge base, a set of contextual time-validity rules. Each rule represents a way to decide how long a given triple will be considered valid.

Definition 3 (Time-validity rule): A time validity rule $r = (p, d)$ associates a triple pattern p to a duration d. A triple pattern is a triple in which some elements might have been replaced by variables, acting as wildcards. The duration d represents the time during which triples that match the pattern p will be considered valid.

For any triple $\langle s, p, o \rangle$ to be added to the knowledge base, we therefore select the rules $r = (p, d)$ for which the triple matches the pattern p, and create a quad of the form

$$\langle s, p, o, \texttt{current\_time} + d \rangle.$$

For example, given a triple

```
location:Podium robo:hasTemperature 20.6 .
```

the rule $r = (p, 300)$ with p being the pattern

```
?x robo:hasTemperature ?y .
```

applies and can be used to decide that the considered triple will expire in 5 minutes (and therefore add it to the relevant named graph).

Note that more than one rule might match a given triple. In this case, we prioritize the most specific rule, and if some rules are equally specific, we use the shortest declared duration. For example, a rule with a pattern

```
location:Podium robo:hasTemperature ?y .
```

would be considered more specific than the one previously defined.

C. Assessing the time-validity of query results

In terms of the process described in Figure 1, the first step to be considered once having establish the way to handle time-validity information in the knowledge base is to process the user query so that we can extract which part of the knowledge base used to answer the query is invalid, and establish a plan to update this part.

SPARQL is the standard language for querying RDF knowledge bases. To briefly illustrate the principles on which SPARQL relies, the example below shows a *SELECT* query asking for the temperature in every meeting room:

```
SELECT ?t ?g WHERE {
   ?x rdf:type robo:MeetingRoom .
   ?x robo:hasLocation ?y .
   ?y robo:hasTemperature ?t
}
```

We can see from this example that SPARQL is essentially based on defining a set of triple patterns, which identify a

```
<http://data.open.ac.uk/kmi/area/Podium>
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://data.open.ac.uk/kmi/robo/MeetingRoom> ;
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://data.open.ac.uk/kmi/robo/Location> ;
        <http://data.open.ac.uk/kmi/robo/xCoord> "7.05" ;
        <http://data.open.ac.uk/kmi/robo/yCoord> "-29.3" .
```

Fig. 2. Example of RDF representation of a room.

sub-graph within the knowledge base, and returning from this sub-graph the nodes (resources, literals) and the relations (predicates) that correspond to bound variables.

Our goal is therefore to extract from the knowledge base the expiry dates of all the triples in the sub-graph that match the given triple patterns. Without going into the details, this is achieved by first executing the query onto the current (potentially time-invalid) knowledge base, and monitoring the query execution process to retrieve the graphs from which triples are obtained. From this, we can obtain the latest expiry date of each of these triples, and identify which have expired and which have not.

The objective of this step is first to assess whether the knowledge base is currently able to answer the query with time-valid information, but also (in case it is not) to create a plan for an autonomous agent to be used as a dynamic, mobile sensor that can update the knowledge base with the necessary, time-valid information. The input to this planning phase, as described in the next section, is therefore constructed as a set of RDF quads, formed out of the triples that require an update, with the fourth element representing the current state of time-validity of this triple. To simplify this step, we transform the expiry dates extracted from the query and the knowledge base into a "time to invalidity" value, representing the time during which a given triple will remain valid (which is negative in case the triple is currently time-invalid).

D. Creating a plan to collect time-valid information

Based on the foundations described in the previous section, the problem represented by the second step of the overall process described in Figure 1 can be formulated as: *Given a set of time-invalid quads and an autonomous agent (a robot) capable of navigating and sensing the kind of information required, how to utilize the robot to update the time-invalid information?* This can be straightforwardly translated into a planning problem, where the plan to be generated is designed to favor the validity in time of the information collected by the robot. In this section, we describe the basic elements of this planning task.

Since the focus of this article is on the time-validity representation and on demonstrating the feasibility of the overall approach, we employ a naïve implementation of a planner based on fixed and constant motion times between locations, and make no claim regarding the efficiency of this planner. We use a best-first search strategy where a plan is a path in a tree, whose nodes represent the states of the considered quads and the current location of the robot, and whose edges are the operators (i.e. the actions) that might be applied by the robot to transition from one state to the other, as per the definitions below.

Definition 4 (Plan state): A state $s = (Q, loc)$ is represented by a set of quads Q, including their expected time to invalidity tv, and the expected location loc of the robot. To simplify, the location is represented in terms of a room or area (e.g. the Podium in our previous example), for which information about coordinates is available.

Definition 5 (Robot operator): An operator represents an action that the robot can perform, as well as a transition from a given state to a modified state. We consider two types of operators: Sensing operators that carry out a measurement and modify the time-validity of the corresponding quads, and movement operators that update the location of the robot[5]. Each operator O is associated with the estimated time t_O it requires to be executed, which is often dependent on the parameters of the operator (e.g. the duration of a movement operator depends on the location of the robot in the input state and on the required location in the output state).

Therefore, starting from an initial state corresponding to the quad extracted from the user query as described in the previous section, and the current location of the robot, the objective is to find the best path to a state where all the information in the quads is time-valid. A key element here is in the way we evaluate a plan in terms of the time-validity of the information resulting in executing the plan. Indeed, while it would appear natural to compute a plan that minimizes the time required to execute the plan, this "shortest" plan might in some cases lead to information that is less "fresh" than it could be, depending on the time-validity duration of the various measures to be collected. We therefore define the cost of a transition between states (i.e. the application of an operator) in the following way:

Definition 6 (Cost of a transition): The cost of a transition, corresponding to applying an operator at a given stage of the plan, is calculated as a function of the minimum time to invalidity of the quads of the state generated as output of the applied operator. In other words, given $pre = (Q, loc)$ the current state, O the operator to apply and $post = (Q', loc')$ the state resulting from applying o on the state pre, the cost of the transition from pre to $post$ is given by:

$$cost(pres, post) = - \min_{\forall <s,p,o,tv> \in Q'} (tv)$$

Naturally, this requires that the creation of the output state

[5]This is assuming that the implementation of the approach relies on robots with basic spatial navigation and motion planning capabilities.

post within the planning process is designed so that it updates the quads in Q' from Q depending on both the time t_O required to execute the operator (Definition 5) and on the information produced by O in the case of a sensing operator.

Following from this, a key property of applying best-first search for planning in the setting described above is that the search is optimal with respect to the considered cost-function. This means that, if a valid plan exists that leads to a state where all the quads in the final state are time-valid, we can guarantee (based on the cost-function in Definition 6) that the plan will produce information which stays valid for the longest possible time. On the other hand, it also means that if the best-first search process fails to produce a plan where all quads end-up being time-valid, we can guarantee that no such plan exists (i.e. that it always takes longer to collect the required information than for this information to expire). The next section illustrates a set of use cases within the working environment of a research lab.

V. Experiments

The aim of this section is to evaluate the feasibility of our approach. We used a real-world example, where a robot has to move in the KMi department on demand, and updates the outdated information of the KMi knowledge base. Next, we present the four scenarios that were designed, which reflect real application requests. We report the queries that were asked to the system and discuss its behavior in details. Note that a video of each scenario can be seen online, following the link provided. Our code is also publicly available online[6].

A. Experimental Setting

For the experimentation, in a first instance we acquired a map of the north wing of KMi through SLAM, as provided in the `gmapping` package of the Robot Operating System (ROS). As platform, we used a iRobot Create 2 equipped with a Kinect to emulate the laser range finder. The so-acquired map used in the experiment is shown in Figure 3. It consists of 8 areas, divided in 4 meeting rooms (namely the Podium, the MarkBucks, Room 20 and Room 22), and 4 open-space activity areas numbered from 2 to 5.

We then built the KMi knowledge base as follows. We represented environments as URI resources, e.g. the `location:Podium` that we used in our running example. We then used coordinates, types of area and sensor information such as temperature, humidity, wifi signal quality in dB and number of people, to define the set of static and dynamic predicates to build an RDF representation for each area. We invite the reader to refer to Figure 2 for an example of such a representation. At the first initialization of the knowledge base, all the dynamic information is set as invalid.

In the following step, we defined the set of time-validity rules r that will be matched against a triple once a new information is sensed, in order to establish the triple expiry date. For the sake of readability, we report in Table I only the time-validity that can be inferred from the rules, for each location (rows) and property (columns).

[6]`http://data.mksmart.org/apps/dka/code`

TABLE I

Time-validity (in secs) of the properties for each location.

Room	temp	hum	wifi	people	X	Y	type[7]
Podium	180	300	540	1.800	∞	∞	∞
MarkBucks	180	300	540	1.800	∞	∞	∞
Room 20	300	300	540	3.600	∞	∞	∞
Room 22	60	300	540	1.800	∞	∞	∞
Activity 2	300	300	540	3.600	∞	∞	∞
Activity 3	300	300	540	3.600	∞	∞	∞
Activity 4	300	300	540	3.600	∞	∞	∞
Activity 5	300	300	540	3.600	∞	∞	∞

From our setting, we can see how the coordinates of a location or its type (meeting room/activity area) never lose time-validity, while sensed information does. We considered activity areas to have a longer time-validity, because they are areas with assigned desks, while meeting rooms, that are booked on regular basis for different purposes, have a shorter time-validity.

Finally, we carried out the evaluation with a simulation. We simulated the environment and the robot through the `stage` simulator of the ROS, and used `move_base` as motion planner. We also activated odometry and laser reading errors in the simulator in order to reproduce real application conditions. We also simulated sensor readings using a pseudo-random generator for temperature, humidity and people count, while wifi signal was read directly from the interface of the laptop running the simulation.

B. Scenarios

The four scenarios we designed are described next, including the original query and a discussion on the behavior of our system.

1) Which activity area has the best wifi signal?[8]

The first scenario is a simple query in which a user wants to know where is the best activity area in terms of wifi signal. This means that we need to measure the wifi signal of 4 activity areas before answering, and consequently that the system needs to find a plan that guarantees that none of the 4 wifi signals will be expired by the time the robot has achieved the last operation. The query is expressed as:

```
SELECT ?room ?wifiSignal
WHERE {
    graph ?expiryDateInMs {
        ?room robo:hasWiFiSignal ?wifiSignal.}.
    graph ?static {
        ?room a location:Activity. }
    }
ORDER BY DESC(?t) LIMIT 1
```

In the video, the knowledge base is queried a first time to check how many results are time-invalid. In this case, all the information about the wifi signal in the activities is expired and has to be updated. A plan of 8 operations is found. The time-validity for the wifi signal is likely to be sufficiently long (540 seconds, as per Table I) to guarantee to performed

[7]Columns X, Y and type correspond to the properties `robo:Xcoord`, `robo:Xcoord` and `rdf:type` of Figure 2.

[8]`http://data.mksmart.org/apps/dka/v1/video1`

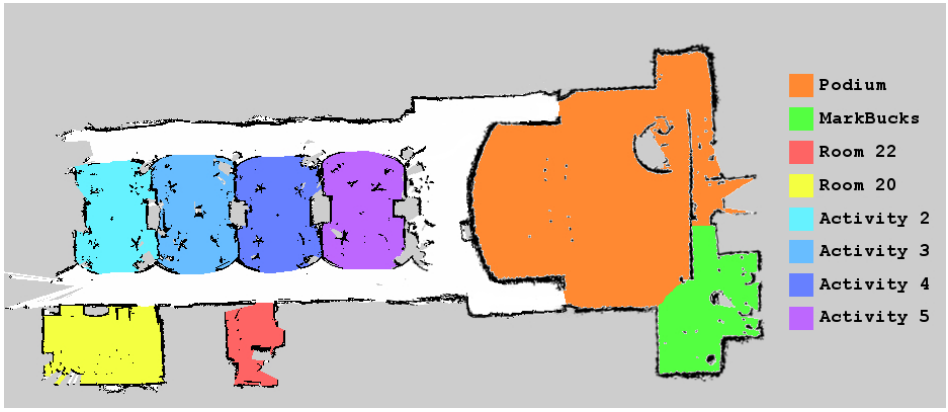

Fig. 3. Map of the Knowledge Media Institute (north wing).

the entire plan without losing time-validity of the results. The plan is therefore sent to the robot, which moves from Activity 2 to Activity 5 starting from its original position, i.e. Room 22. As said, each time that the robot reaches an Activity and senses its wifi, the new information is written in the knowledge base with an epoch timestamp of when the triple will expire, as:

```
location:Activity2 robo:hasWiFiSignal -81
                                timegraph:1468917352
```

Once the robot has executed the plan, the original query is executed again, and the valid answer (in our case, Activity 2) is shown to the user.

2) Which is the temperature of Room 20 and 22?[9]

In this scenario, a user wants to know the temperature of Room 20 and Room 22 before choosing where to work. This is expressed as follows:

```
SELECT ?room ?temp
WHERE {
    graph ?expiryDateInMs {
      VALUES(?room)
        {(location:Room20 ) (location:Room22)}.
      ?room robo:hasTemperature ?temp.
      }
    }
```

The robot is situated in the MarkBucks and needs to sense the temperatures of both rooms. The plan to be executed includes 4 actions. As one can see, although Room 22 is the first one on the robot's path, the robot moves first to Room 20 and then to Room 22. This is motivated by the fact that Room 22 has a shorter time-validity, which would expire if sensed before the one of Room 20. This scenario shows in practice how the plan is designed to favor the validity in time of the information collected and not the overall time required to execute the plan. Once both temperatures have been sensed (respectively, `location:Room20 robo:hasTemperature 18.26` and `location:Room22 robo:hasTemperature 21.09`) and updated in the knowledge base, they are finally shown to the user.

[9]http://data.mksmart.org/apps/dka/v1/video2

3) Which meeting room is more comfortable?[10]

In this scenario, the user wants to know which meeting room is more comfortable to book its meeting. We designed a simple comfort factor based on temperature, humidity and number of people in a room, as $comfort = \frac{temp*h}{ppl}$, and execute the following query:

```
SELECT ?room ((?temp+?h)/?ppl) AS ?comfort)
WHERE {
  graph ?g { ?room robo:hasPeopleCount ?ppl }.
  graph ?g1 {?room robo:hasHumidity ?h }.
  graph ?g2 {?room robo:hasTemperature ?temp }.
  graph ?g3 {?room a location:MeetingRoom }
  }
ORDER BY DESC(?t) LIMIT 1
```

The robot is situated in the MarkBucks and is supposed to move in 4 different spaces and update their temperature, humidity and wifi signal before sending a response to the user. However, the video shows that an empty plan is sent to the robot and the answer to the query is not updated. This means that no valid plan existed, where all the quads involved in the answer could be time-valid and, in order to guarantee it, the initial query should be simplified.

4) Which meeting room is more comfortable?[11]

Following from the previous scenario, the user simplifies the query and asks for the comfort of Room 22 and the Podium. A query similar to the previous one is executed:

```
SELECT ?room ((?temp+?h)/?ppl) AS ?comfort)
WHERE {
  graph ?g {
    VALUES(?room)
      {(location:Room22) (location:Podium)}.
    ?room robo:hasPeopleCount ?ppl }.
  graph ?g1 {
    VALUES(?room)
      {(location:Room22) (location:Podium)}.
    ?room robo:hasHumidity ?h }.
  graph ?g2 {
    VALUES(?room)
      {(location:Room22) (location:Podium)}.
    ?room robo:hasTemperature ?temp }.
  }
ORDER BY DESC(?t) LIMIT 1
```

[10]http://data.mksmart.org/apps/dka/v1/video3
[11]http://data.mksmart.org/apps/dka/v1/video4

and this time a plan of 8 actions is found. The robot moves from the MarkBucks towards the Podium, updates the information about this room and then moves to Room 22 to update its information. The comfort is then calculated and the Podium is shown in the answer to the user as the most comfortable (with a comfort factor of -0.2[12]).

VI. Conclusions

This paper presented an approach to update information in a knowledge base that has lost its validity in time using an autonomous mobile agent moving on demand. We showed how we relied on Semantic Web technologies for the representation of our knowledge base, namely how we used the RDF model to represent the time-validity by annotating statements with time-stamps representing their expiry date, and how we used the SPARQL query language to assess the validity of new information to be written in the knowledge base, based on a set of time-validity rules. Once defined that, we used such knowledge representation to design a basic planner where the generated plans favor the validity in time of the information to be collected by the mobile agent.

Our work demonstrates the feasibility of the approach and the encouraging results opened new perspectives to us. As said, we focused on the knowledge representation aspect and left the optimization of the planner for future work, e.g. by using of more realistic models to calculate motion times rather than constant travel times, or using time-constraints such as the timeliness of the response expected by the user. This would allow to express more complex queries without running into scalability issues. A more flexible notion of information validity, which decays with time, should also be investigated. In such a way, we could report the best information possible for a query, rather than discarding plans and reporting no answer as soon as validity cannot be guaranteed anymore. We also look into utilizing the same knowledge representation framework in a scenario with multiple coordinating robots, therefore making it more likely that a plan can be found that can be executed by the time the information has lost validity. Finally, we intend to work on more complex time-validity rules, and to see how this affects our process.

References

[1] Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). Models and issues in data stream systems. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (pp. 1-16). ACM.

[2] Balduini, M., Della Valle, E., Dell'Aglio, D., Tsytsarau, M., Palpanas, T., & Confalonieri, C. (2013). Social listening of city scale events using the streaming linked data framework. In International Semantic Web Conference (pp. 1-16). Springer Berlin Heidelberg.

[3] Barbieri, D. F., Braga, D., Ceri, S., Della Valle, E., & Grossniklaus, M. (2009). C-SPARQL: SPARQL for continuous querying. In Proceedings of the 18th international conference on World wide web (pp. 1061-1062). ACM.

[4] Calbimonte, J. P., Jeung, H. Y., Corcho, O., & Aberer, K. (2012). Enabling query technologies for the semantic sensor web. International Journal on Semantic Web and Information Systems, 8(EPFL-ARTICLE-183971), 43-63.

[12]The comfort factor may result in a negative score since the strength of the wi-fi signal is likely to take negative values.

[5] Ciullo, D., Celik, G. D.,& Modiano, E. (2010). Minimizing transmission energy in sensor networks via trajectory control. In Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on (pp. 132-141). IEEE.

[6] Dvorak, F., Bit-Monnot, A., Ingrand, F., & Ghallab, M. (2014). A flexible ANML actor and planner in robotics. In Planning and Robotics (PlanRob) Workshop (ICAPS).

[7] Fernández, J. D., Schneider, P., & Umbrich, J. (2015). The DBpedia wayback machine. In Proceedings of the 11th International Conference on Semantic Systems (pp. 192-195). ACM.

[8] Frank, J., & Jónsson, A. (2003). Constraint-based attribute and interval planning. Constraints, 8(4), 339-364.

[9] Goerner, J., Chakraborty, N., & Sycara, K. (2013). Energy efficient data collection with mobile robots in heterogeneous sensor networks. In Robotics and Automation (ICRA), 2013 IEEE International Conference on (pp. 2527-2533). IEEE.

[10] Gutierrez, C., Hurtado, C., & Vaisman, A. (2005). Temporal rdf. In European Semantic Web Conference (pp. 93-107). Springer Berlin Heidelberg.

[11] Halpin, H., & Cheney, J. (2014). Dynamic provenance for SPARQL updates using named graphs. In Proceedings of the 23rd International Conference on World Wide Web (pp. 287-288). ACM.

[12] Haslum, P., & Geffner, H. (2014). Heuristic planning with time and resources. In Sixth European Conference on Planning.

[13] Komazec, S., Cerri, D., & Fensel, D. (2012). Sparkwave: continuous schema-enhanced pattern matching over RDF data streams. In Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems (pp. 58-68). ACM.

[14] Motik, B. (2012). Representing and querying validity time in RDF and OWL: A logic-based approach. Web Semantics: Science, Services and Agents on the World Wide Web, 12, 3-21.

[15] Muscettola, N., Dorais, G. A., Fry, C., Levinson, R., & Plaunt, C. (2002). Idea: Planning at the core of autonomous reactive agents.

[16] Le-Phuoc, D., Dao-Tran, M., Pham, M. D., Boncz, P., Eiter, T., & Fink, M. (2012). Linked stream data processing engines: Facts and figures. In International Semantic Web Conference (pp. 300-312). Springer Berlin Heidelberg.

[17] Pugliese, A., Udrea, O., & Subrahmanian, V. S. (2008). Scaling RDF with time. In Proceedings of the 17th international conference on World Wide Web (pp. 605-614). ACM.

[18] Rajan, K., Py, F., McGann, C., Ryan, J., O'Reilly, T., Maughan, T., & Roman, B. (2009). Onboard Adaptive Control of AUVs using Automated Planning. In International Symposium on Unmanned Untethered Submersible Technology (UUST), Durham, NH.

[19] Rinne, M., Abdullah, H., Törmä,, S., & Nuutila, E. (2012). Processing Heterogeneous RDF Events with Standing SPARQL Update Rules. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 797-806). Springer Berlin Heidelberg.

[20] Sheth, A., Henson, C., & Sahoo, S. S. (2008). Semantic sensor web. IEEE Internet computing, 12(4), 78-83.

[21] Stock, S., Mansouri, M., Pecora, F., & Hertzberg, J. (2015). Hierarchical hybrid planning in a mobile service robot. In Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz) (pp. 309-315). Springer International Publishing.

[22] Tappolet, J.,& Bernstein, A. (2009). Applied temporal RDF: Efficient temporal querying of RDF data with SPARQL. In European Semantic Web Conference (pp. 308-322). Springer Berlin Heidelberg.

[23] Urbani, J., Margara, A., Jacobs, C., Van Harmelen, F., & Bal, H. (2013). Dynamite: Parallel materialization of dynamic rdf data. In International Semantic Web Conference (pp. 657-672). Springer Berlin Heidelberg.

[24] Van de Sompel, H., Sanderson, R., Nelson, M. L., Balakireva, L. L., Shankar, H., & Ainsworth, S. (2010). An HTTP-based versioning mechanism for linked data. arXiv preprint arXiv:1003.3661.

[25] Volz, R., Staab, S., & Motik, B. (2005). Incrementally maintaining materializations of ontologies stored in logic databases. In Journal on Data Semantics II (pp. 1-34). Springer Berlin Heidelberg.

[26] Wang, R., Veloso, M., & Seshan, S (2016). Active Sensing Data Collection with Autonomous Mobile Robots. In Proceedings of IEEE International Conference on Robotics and Automation.

[27] Yuan, B., Orlowska, M., & Sadiq, S. (2007). On the optimal robot routing problem in wireless sensor networks. IEEE Transactions on Knowledge and Data Engineering, 19(9), 1252-1261.