Context-Addressable Messaging Service with Ontology-Driven Addresses*

Jaroslaw Domaszewicz, Michal Koziuk, and Radoslaw Olgierd Schoeneich

Institute of Telecommunications, Warsaw University of Technology
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
{domaszew,mkoziuk,rschoeneich}@tele.pw.edu.pl

Abstract. The context-addressable messaging service allows applications to send messages to mobile users described by their context. The context of these users is described in terms of ontology assertions, and an ontology-driven expression is used as a context-based address. This expression can be interpreted as a definition of a new ontology class. The recipients of a context-addressed message are all the users (nodes) whose context makes them instances of the address class. This paper presents a model for an ontology-based context-addressable messaging system, and provides performance results of its implementation based on available 'off-the-shelf' software.

1 Introduction

In mobile context-aware systems, a piece of context data can often be associated with a particular node in the network. Such node-specific context data are used to describe the node's location, its environment, the role of its user, etc. Special use of this node-specific context data can be made to provide support for a Context-Addressable Messaging (CAM) service. This service allows sending messages whose recipients are specified using context data.

A simple use case for Context-Addressable Messaging is presented in Fig.1[1]. Imagine that on the scene of a major emergency, an injured person requires medical assistance. Such a person could inform nearby medical staff about his situation by sending a message with a request for help, with an appropriate context-based description of desired recipients (e.g. "All unoccupied medical staff which are within 1km from me"). The description is what we call a Context-based Address. A Context-based Address is used to describe the context of nodes to which a message is to be delivered. The Context-Addressable Messaging service would deliver the request for help to all users of the system meeting the specified criteria, in particular the two unoccupied nurses within 1km from the sender. Medical staff in the area who are currently occupied would not receive the message.

* This work was supported by the 6FP MIDAS IST project, contract no. 027055.

[1] Cliparts taken from Open Clip Art Library at http://openclipart.org/

R. Meersman and Z. Tari (Eds.): OTM 2008, Part II, LNCS 5332, pp. 1471–1481, 2008.

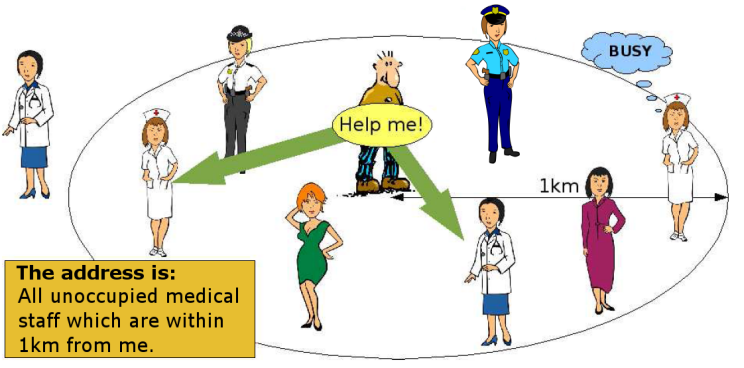

Fig. 1. Illustration of Context-Addressable Messaging

Medical staff further than 1 km from the injured person would not receive it either.

The CAM service, as envisioned in this paper, is: (a) unreliable (the service is best effort), (b) connectionless, (c) datagram-oriented (each message send operation causes a network packet to be sent), (d) group-oriented (a Context-based Address describes a group of message recipients), and (e) one-way (responses, if needed, can be handled by a regular, unicast communications service).

We have identified three major issues important for Context-Addressable Messaging. First, the system requires a context modeling mechanism. Second, a language for constructing Context-based Addresses is needed. Third, a dedicated routing protocol (not addressed in this paper) is required to efficiently route Context-Addressed Messages to their destinations.

The key contribution of this paper is the idea of using ontology-driven, runtime-formed class definitions as Context-based Addresses. A Context-based Address is constructed from terms taken from a context modeling ontology, using operators offered by the concept description language of a selected ontology language. Each address can be viewed as a definition of a new concept (class), which does not exist in the original ontology. The intended recipients of a Context-Addressed Message are all the nodes whose context makes them instances of the address class. Another contribution is the architecture for an ontology-driven Context-Addressable Messaging service, constructed as a middleware layer. The middleware is a domain-neutral framework customized for a specific domain by an exchangeable context modeling ontology. Finally, this paper presents an evaluation of a proof-of-concept implementation of the middleware, including performance results and conclusions for future research.

This paper is organized as follows. Section 2 presents existing solutions similar to the CAM service. Section 3 explains the proposed structure of Context-based Addresses. Section 4 presents the architecture of a middleware which provides the CAM service. Section 5 contains results of the performed experiments. Conclusions and directions for future work are presented in section 6.

2 Related Work

Addressing network nodes by their attributes has already been proposed in the literature. In particular, discarding node identifiers has been of particular interest to the wireless sensor networks research community, where focus is placed on creating a data-centric network. Solutions such as Directed Diffusion [1] or the EYES project [2] propose to use attributes of nodes (coupled with values of those attributes) in order to describe destination nodes.

A similar context-based messaging architecture for MANET networks, called FlavourCast [3], has been proposed. There, destination nodes can be described by an arbitrary attribute such as color.

Another area where nodes are not interested in directly addressing each other is Content-Based Addressing [4] [5]. In this solution communicating entities specify the content of the data they are producing and send out appropriate notifications. Nodes which would like to receive data, subscribe to it by creating appropriate filters. The system selects relevant messages, by matching the attributes and values from notifications with constraints contained in filters.

The main difference between all the mentioned work and our research is that (a) we propose to incorporate a domain model in the form of an ontology and (b) we use ontology-driven, runtime defined classes as addresses. All the above solutions rely on functions performed on attributes, their values, and arbitrary other parameters. There are however, a number of systems which aim at enhancing the publish/subscribe scheme with ontologies. S-ToPSS (Semantic Toronto Publish/Subscribe System) [6] proposes to introduce semantics into publish/subscribe matching algorithms, by using a taxonomy of concepts from a domain specific ontology (as one of three possible extensions).

A similar concept (which uses ontologies to match subscriptions and published data) is presented in [7] and [8], where the Siena subscription language is extended by ontologies and ontological operators. The proposed extension relies on the usage of ontological equivalence and subsumption to perform matching between filters and subscriptions.

Solutions described above are implemented according to the publish-subscribe scheme. The proposed Context-Addressable Messaging service differs from them in that the destination nodes do not need to subscribe to any events. Instead each node receives messages that match its current context (which can change dynamically).

3 Ontology-Driven Context-Based Addresses

A basic assumption behind Context-Addressable Messaging is that each node in the network has access to a common context (domain) model. The context model is an ontology which describes the current domain of the CAM service. Thus, there exists a collection of predefined classes, relations and individuals specific to the domain. Every piece of information entered into the system has to be structured in terms of the ontology. A simple example is presented in Fig.2.

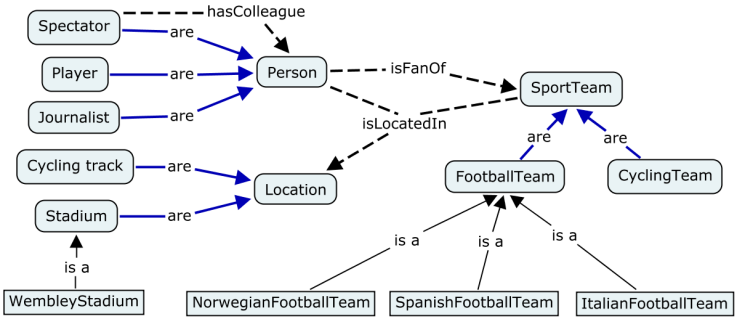

Fig. 2. A simple ontology

Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap ... \sqcap C_n$	Human ⊓ Male
unionOf	$C_1 \sqcup ... \sqcup C_n$	Doctor ⊔ Lawyer
complementOf	$\neg C$	¬ Male
oneOf	$\{x_1 \ ... \ x_n\}$	{john,mary}
toClass	$\forall P.C$	∀hasChild.Doctor
hasClass	$\exists r.C$	∃hasChild.Lawyer
hasValue	$\exists r.\{x\}$	∃CitizenOf.{USA}
minCardinalityQ	$(\geq nr.C)$	(≥2 hasChild.Lawyer)
maxCardinalityQ	$(\leq nr.C)$	(≤1 hasChild.Male)
inverseOf	r^-	hasChild^-

Fig. 3. An example of a concept description language (taken from [9])

$$Spectator \sqcap (\exists isLocatedIn.\{WembleyStadium\}) \sqcap (\exists isFanOf.\{SpanishFootballTeam\})$$

Fig. 4. An example of a Context-based Address (in a Description Logic notation)

In our approach to CAM, Context-based Addresses are domain ontology-driven classes defined with a concept description language (a part of a selected ontology language). In other words, Context-based Addresses are definitions of new classes, formed from existing concepts (classes, relations, individuals) by means of concept constructors (operators). The structure of these definitions follows the formalism of the chosen ontology language. An example of concept constructors from a concept description language (taken from [9]) is presented in Fig.3.

Consider the example ontology of a sports domain, presented in Fig.2. A sample Context-based Address for this ontology is presented in Fig.4. This address denotes all spectators located at the Wembley Stadium who are fans of the

Address:	Payload:
`Spectator and (isLocatedIn value WembleyStadium) and (isFanOf value SpanishFootballTeam)`	(ANY DATA)

Fig. 5. An example of a Context-Addressed Message

Spanish football team. It is constructed by combining one class, two relations, and two individuals, all belonging to the sports domain ontology. Two concept-description language constructors (`intersectionOf` and `hasValue`) are used in the definition. Note that such an address class does not exist in the ontology; it is produced at runtime by an application or a user.

A binary representation of a Context-based Address is placed in the header of a Context-Addressed Message (see Fig.5). The message's payload can be any data. Such message is sent into the network and is delivered to all nodes whose context matches the description contained in the attached address (i.e., whose context makes them instances of the address class). Note that the Manchester OWL Syntax [10] notation used in Fig.5 is equivalent to the Description Logic notation from Fig.4.

4 CAM Middleware Architecture

The CAM middleware architecture, for a single mobile node, is given in Fig.6. As can be seen, the domain model (i.e., an ontology) is not hardwired into the middleware but is imported into it. To use the middleware in a different environment (i.e., for a different domain), it is enough to exchange the domain model. Hence, the CAM middleware can be described as a domain-neutral framework customized by a domain model. All nodes in the network have to use the same domain model.

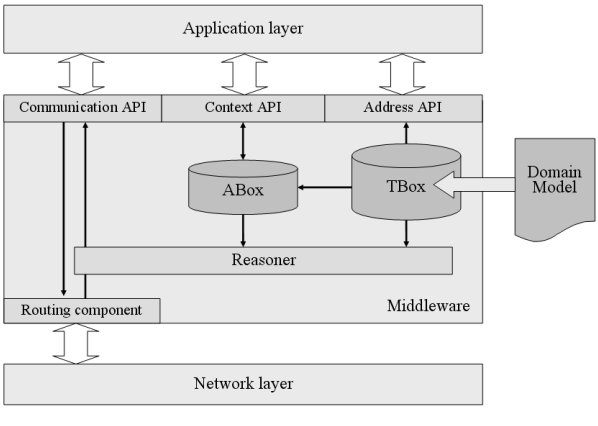

Fig. 6. CAM middleware architecture (one mobile node shown)

The CAM middleware consists of four main building blocks: the internal domain model representation (TBox), the node's context data (ABox), the reasoner, and the routing component. The TBox holds a runtime representation of all the information provided in the domain model ontology (such as, for example, the class hierarchy). The TBox, once produced from the imported domain model, remains unchanged at runtime, except for temporary insertion of Context-based Addresses (explained below). The context data (ABox) are statements entered into the middleware by applications and expressed in terms of the domain model (such as a statement that some instance belongs to a certain class). The ABox changes at runtime in response to changing context of the node. Also, while the TBox is the same at all nodes in the network (all the nodes share the domain model), the ABox is node-specific. A node's ABox contains context data collected and injected into the middleware by this node's applications (it is the applications' responsibility to acquire context from the environment and inject it into the middleware). In general, the context data stored in the Abox vary from node to node. The middleware reasoner performs address resolving, i.e., using the ABox and the TBox, it provides an answer to the question: "does this node's context match a certain Context-based Address". Finally the routing component routes Context-Addressed Messages. The goal of the routing component is to avoid flooding the network with Context-Addressed Messages. As a result, a Context-Addressed Message is delivered to only a subset of nodes, so that address resolving need not be performed by all nodes in the network[2].

Among the context data items stored in a node's ABox, we distinguish a special ABox individual which we refer to as `thisNode`. Facts about the context of the node are expressed as: (a) object properties linking `thisNode` with other individuals, (b) datatype properties assigning certain values to `thisNode` or (c) statements that `thisNode` belongs to certain classes. Of course, once a relation exists between `thisNode` and other individuals, the facts related to those individuals also become a part of the node's context. An important feature of this way of context representation is that each node holds a self-centric view of the context, centered around the `thisNode` individual.

The key part of the CAM middleware is the address resolving, i.e., checking if the node's context (i.e., all data in the node's ABox directly or indirectly related to the `thisNode` individual) matches the Context-based Address of a newly received Context-Addressed Message. The address resolving is done at each prospective message recipient. This is carried out as a three step process (illustrated in Fig.7). First, the definition of a class forming the Context-based Address is inserted to the TBox. Next, the reasoner is requested to determine (taking into account all relationships captured by the domain ontology) if the `thisNode` individual belongs to this new class. As the domain model has been changed, the reasoner has to perform a classification process, on a structure which includes the new class. Finally, after a response to the query has been produced, the address class is removed from the TBox. This is required to restore

[2] The routing component as envisioned by the authors, is described in [11].

Address: Spectator **and** (isLocatedIn **value** WembleyStadium) **and** (isFanOf **value** SpanishFootballTeam)
Step 1: - Add the class `CL = Spectator and (isLocatedIn value WembleyStadium) and (isFanOf value SpanishFootballTeam)` to the TBox Step 2: - Check if `thisNode` individual belongs to the class `CL` Step 3: - Remove the class `CL` from the TBox

Fig. 7. The steps of address resolving

a) Defining the context of a node.

```
//entering context data
ClientContextAPI thisNode = new ClientContextAPI();
thisNode.addBelongsToClass("Spectator");
thisNode.addUniqueProperty("isLocatedIn", "WembleyStadium");
thisNode.addProperty("isFanOf", "SpanishFootballTeam");
//retrieving context data
String myTeam = thisNode.getProperty("isFanOf");
```

b) Creating Context-based Addresses.

```
ClientAddressAPI a = new ClientAddressAPI();
int Spectator = a.getClass("Spectator");
int SpectatorAtWembley = a.addRestriction(Spectator, "isLocatedIn", "WembleyStadium");
int SpanishFanAtWembley = a.addRestriction(SpectatorAtWembley, "isFanOf",
"SpanishFootballTeam");
```

c) Sending and receiving Context-Addressed Messages.

```
//sending a message
ClientCommAPI c = new ClientCommAPI();
c.send(a.getProperAddress(SpanishFanAtWembley),
"The game of the Spanish team will start in 10 minutes !!!");
//receiving a message
c.buffer.wait();
processMessage(c.buffer.data);
```

Fig. 8. Using the CAM middleware

the TBox of the ontology to its original state. Depending on the query response, the message is either passed to the application layer or discarded.

Fig.8 illustrates how the described middleware can be used by applications. Fig.8a shows an example of how the context of a node can be entered into the middleware using the Context API (retrieval and removal of context data is also possible). Fig.8b presents an example of how different Context-based Addresses can be constructed using the Address API. Fig.8c illustrates the Communication API, showing how a message can be sent to a constructed address, and how a message can be received by nodes whose context matches the message's address.

5 Experimental Results

The CAM middleware architecture has been implemented by using mainstream technologies and off-the-shelf software components presented in Fig.9. As the research on a dedicated routing protocol was still work in progress, we used flooding based on broadcasting for delivery of messages (i.e., all nodes in the network were prospective recipients).

Issue	Solution
Ontology language	OWL-DL
Tbox and ABox	JENA [12]
Reasoner	Pellet [13]
Routing	None (flooding)

Fig. 9. CAM middleware implementation summary

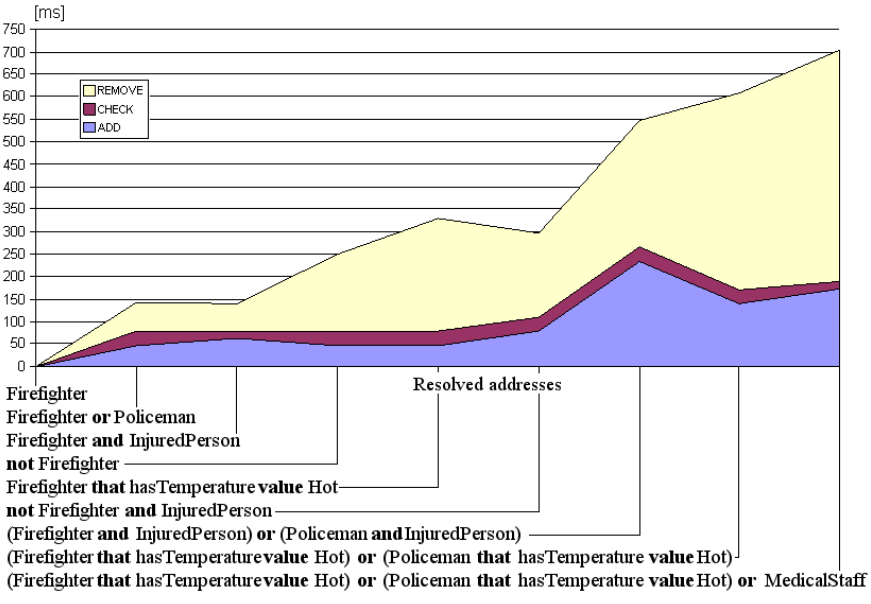

Fig. 10. Address resolution time at the receiver for a small emergency ontology

Performance testing of the CAM middleware was performed on a PC with a 2,66GHz Celeron processor and 1GB RAM. During the tests we used two ontologies. The first one was a very small ontology for emergency situations, developed internally for test purposes. The second ontology was the publicly available, much larger Pizza [14] ontology.

To evaluate the performance of the CAM middleware, we measured the times of different stages of address resolving (due to reasoning, it is by far the most time consuming operation in the middleware). The test procedure was the following: (a) the context of each node was set-up and did not change later on, (b) a Context-based Address was created on one node, (c) the message with this address was sent to the other nodes, and (d) one of the destination nodes measured the time required to resolve the address. As address resolving is a three phase process, we measured the time required to perform each phase.

Fig.10 presents the results obtained for the emergency ontology. The first resolved address is a class present in the TBox (`Firefighter`). In this case, both the TBox and the ABox do not change and resolving an address is instantaneous.

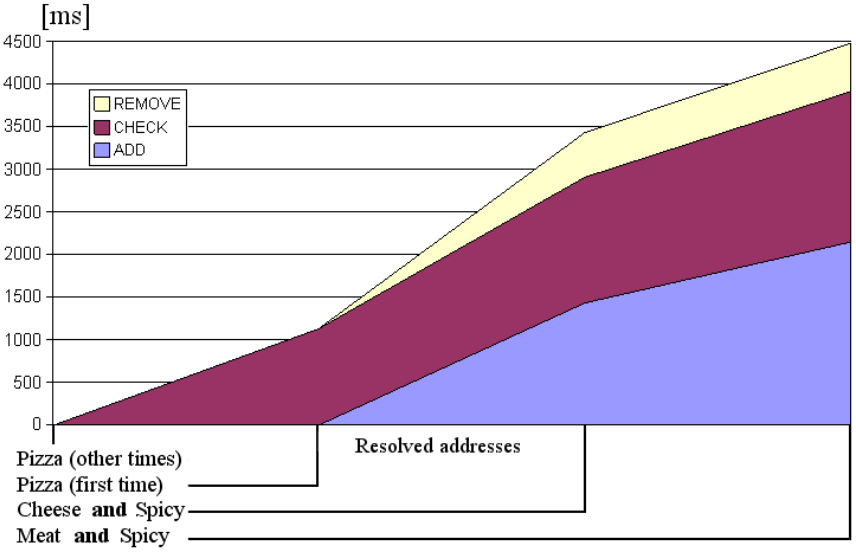

Fig. 11. Address resolution time at the receiver for the pizza ontology

More complicated addresses consist of an intersection or a union of two classes (e.g., `Firefighter or Policeman`), and we observe that the time required for resolving an address increases. The steps which were not required previously (adding a new class, and removing it later) are now performed and constitute the major part of the address resolving process. In general, the more complicated an address, the longer the address resolving process. Resolving the most complicated address (out of the tested ones) requires around 700ms. Looking at results for the significantly bigger Pizza ontology (Fig.11), we observe that the address resolving time has dramatically increased.

Based on the presented results we draw a number of conclusions. First, not surprisingly, the more complicated the address, the longer it takes to resolve it. Second, a significant amount of time required for address resolving is consumed when adding the new address class to the TBox, and later removing it. Finally, the time required to resolve an address grows with the size of the ontology; this has a severe impact on performance of the CAM middleware for large ontologies.

6 Conclusions and Future Work

The performance results of the CAM middleware show that, even for a PC, it is hard to obtain near real-time operation with mainstream, off-the-shelf software technologies. This is even more true for mobile devices. Clearly, the Pellet reasoner has not been optimized for real-time reasoning on a variable TBox

and ABox. We assume that this conclusion holds for other existing OWL-DL reasoners.

We intend to address the above performance problems by (a) choosing an ontology language less complex than OWL-DL (DL-Lite [15], along with its conjunctive query mechanism, seems to be a good candidate) and (b) implementing a dedicated reasoner that can handle the chosen ontology language on a mobile device. Another important part of the work on the CAM architecture is the Context-Based Routing [11] protocol; such a protocol is essential to avoid flooding the network with Context-Addressed Messages and having to do address resolving at every node.

References

1. Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., Silva, F.: Directed diffusion for wireless sensor networking. IEEE/ACM Trans. Netw. 11(1), 2–16 (2003)
2. Handzinski, V., Koepke, A., Frank, Ch., Karl, H., Wolisz, A.: Semantic addressing for wireless sensor networks. Technical report, Telecommunication Networks Group, Technische Universität Berlin (May 2004)
3. Cutting, D., Corbett, D.J., Quigley, A.: Context-based Messaging for Ad Hoc Networks (May 8-13, 2005)
4. Carzaniga, A., Rosenblum, D., Wolf, A.: Content-based addressing and routing: A general model and its application (2000)
5. Carzaniga, A., Wolf, A.L.: Forwarding in a content-based network. In: SIGCOMM 2003: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 163–174. ACM, New York (2003)
6. Petrovic, M., Burcea, I., Jacobsen, H.A.: S-ToPSS: semantic Toronto publish/subscribe system. In: VLDB 2003: Proceedings of the 29th international conference on Very large data bases, VLDB Endowment, pp. 1101–1104 (2003)
7. Keeney, J., Lynch, D., Lewis, D., O'Sullivan, D.: On the Role of Ontological Semantics in Routing Contextual Knowledge in Highly Distributed Autonomic System. Technical report, Department of Computer Science, Trinity College Dublin (2006)
8. Keeney, J., Jones, D., Roblek, D., Lewis, D., O'Sullivan, D.: Knowledge-based semantic clustering. In: SAC 2008: Proceedings of the 2008 ACM symposium on Applied computing, pp. 460–467. ACM, New York (2008)
9. Baader, F., Horrocks, I., Sattler, U.: Description logics as ontology languages for the semantic web. In: Festschrift in honor of Jörg Siekmann. LNCS (LNAI), pp. 228–248. Springer, Heidelberg (2003)
10. Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., Wang, H.H.: The Manchester OWL Syntax. In: OWL: Experiences and Directions 2006, Athens, Georgia, USA, November 10-11 (2006)
11. Schoeneich, R.O., Domaszewicz, J., Koziuk, M.: Concept-Based Routing in Ad-Hoc Networks. In: The 10th International Conference on Distributed Computing and Networking - ICDCN 2009, January 3-6 (submitted, 2009)
12. Carroll, J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: Implementing the semantic web recommendations (2003)

13. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semant. 5(2), 51–53 (2007)
14. Drummond, N., Horridge, M., Stevens, R., Wroe, C., Sampaio, S.: Pizza ontology v1.3 (October 18, 2005), http://www.co-ode.org/ontologies/pizza/2005/10/18/
15. Calvanese, D., Giuseppe, D.G., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: Proceedings of the Twentieth National Conference on Artificial Intelligence, pp. 602–607 (2005)