Towards Management of OWL-S Effects by Means of a DL Action Formalism Combined with OWL Contexts

Domenico Redavid, Stefano Ferilli, and Floriana Esposito

Dipartimento di Informatica - Università di Bari Aldo Moro, Bari 70126, Italy
{redavid,ferilli,esposito}@di.uniba.it

Abstract. The implementation of effective Semantic Web Services (SWS) platforms allowing the composition and, in general, the orchestration of services presents several problems. Some of them are intrinsic within the formalisms adopted to describe SWS, especially when trying to combine the dynamic aspect of SWS effects and the static nature of their ontological representation in Description Logic (DL). This paper proposes a mapping of OWL-S with a DL action formalism in order to evaluate executability and projection by means of the notion of Contexts.

1 Introduction

OWL-S[1] is an OWL ontology that enables semantic description of Web services. One of its purposes is the automation of four use cases, namely Discovery, Selection, Composition and Invocation. In OWL-S, preconditions and effects are represented with logical formulas. These formulas cannot always be translated into OWL DL without losing some of their original semantics. The reasons are manifold. Firstly, the OWL-S specification enables to describe such logical formulas with different languages. Secondly, incompatibility between those languages and Description Logic (DL) makes not feasible the translation between candidate languages into DL themselves. Last, but not least, even if we restrict the representation to formalisms which are compatible with DL, the dynamicity of effects is not expressible natively in DL. In particular, the language at the state of the art that offers a greatest level of compatibility with DLs is the Semantic Web Rule Language (SWRL) [1] in its decidable fragment [2]. The adoption of this formalism for encoding preconditions and effects (partially) solves the first two compatibility issues mentioned above. However, the problem of managing effects without breaking DL semantics remains open. The issues are of ontological nature because DLs are monotonic languages. Although some DLs extensions could be used to deal with the dynamic aspects mentioned above, they are not included in the current OWL specifications. Moreover, DL monotonicity rules out any retraction primitives, i.e., there is no way of removing an axiom from a knowledge base, making it hard to deal with knowledge that changes over time. The consequences vary from wrong results for queries to inconsistencies in the knowledge base. This happens because the effect of a service is intended as an *alteration* of the state of the world. It is quite obvious that the new information should replace the old one, rather than just be added to the knowledge base.

[1] OWL for Services. http://www.w3.org/Submission/OWL-S/

G. Agre et al. (Eds.): AIMSA 2014, LNAI 8722, pp. 228–235, 2014.

2 OWL-S and SWRL Characteristics

OWL-S enables semantic descriptions of Web services using the *Service Model* ontology, which defines the OWL-S process model. Each process is based on the IOPR (Inputs, Outputs, Preconditions, and Results) model. *Inputs* represent the information required for the execution of the process. *Outputs* represent the information the process returns to the requester[2]. *Preconditions* are conditions imposed on *Inputs* that have to hold in order to invoke the process in a correct manner. Since an OWL-S process may have several results with corresponding outputs, the *Results* provide a mean to specify this situation. Each result can be associated to a result condition, called *inCondition*, which specifies when that particular result can occur. It is assumed that such conditions are mutually exclusive, so that only one result can be obtained for each possible situation. When an *inCondition* is satisfied, there are properties associated with this event that specify the corresponding output and, possibly, the *Effects* produced by the execution of the process. The OWL-S conditions (*Preconditions*, *inConditions* and *Effects*) are represented as logical formulas. Since OWL-DL offers limited support to formulate constructs like property compositions without becoming undecidable, a more powerful language is required for the representation of OWL-S conditions. One of the proposed languages is Semantic Web Rule Language (SWRL) [1]. Although SWRL is undecidable, a solution has been proposed in [2] where decidability is achieved by restricting the application of SWRL rules only to the individuals explicitly introduced in the *ABox*. This kind of SWRL rules, called DL-safe, makes this language the best candidate to describe OWL-S conditions [3]. Let us now briefly mention the characteristic of SWRL that are relevant to our scope. SWRL extends the set of OWL axioms to include *Horn-like* rules in the form of implications between an antecedent (body) and consequent (head), both consist of zero or more conjunctive atoms having one of the following forms:

- $C(x)$, with C an OWL class, $P(x, y)$, with P an OWL property,
- *sameAs*(x, y) or *differentFrom*(x, y), equivalent to the respective OWL properties,
- $builtIn(r, z_1, \ldots, z_n)$, functions over primitive datatypes.

where x, y are variables, OWL individuals or OWL data values, and r is a built-in relation between $z_1, \ldots, z_n$ (e.g., $builtIn(greaterThan, z_1, z_2)$). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent hold also. A rule with conjunctive consequent can be transformed into multiple rules by means of Lloyd-Topor transformations. Each rule has an atomic consequent.

3 Action Formalism for OWL-S

The OWL-S composition methodology based on SWRL DL-safe rules presented in [3] explain the procedure to encode the OWL-S services process model by means of the following (abstract) SWRL rule:

[2] Inputs, Outputs and Local variables (entities used within the process) are SWRL variables and their types are defined in the domain ontology.

$$Preconditions \wedge inCondition \rightarrow \{output\} \wedge Effect$$

If the service has more *Results*, multiple rules having different *inCondition*, *output* and/or *Effect* are used. In order to evaluate executability and projection we propose to map this abstract rule with the action formalism proposed in [4] based on $\mathcal{ALCQIO}$, an OWL-DL fragment. In detail, given:

- N_X and N_I disjoint and countably infinite sets of variables and individual names;
- an acyclic TBox $\mathcal{T}$;
- a set of primitive literals for $\mathcal{T}$ corresponding to the ABox assertions $A(a)$, $\neg A(a)$, $r(a, b)$, $\neg r(a, b)$, with A primitive concept in $\mathcal{T}$, r a role name, a,$b \in N_I$.

An atomic action is defined as $\alpha = (pre; post)$ where:

- pre is a finite set of ABox assertions, the preconditions;
- $post$ is a finite set of conditional postconditions of the form φ/ψ, where φ is an ABox assertion and ψ is a primitive literal for $\mathcal{T}$.

An operator for $\mathcal{T}$ is a parametrised atomic action for $\mathcal{T}$, i.e., an action in which some variables from N_X may occur in place of individual names. The postconditions of the form $\top(t)/\psi$ are called unconditional and denoted just by ψ. For the sake of simplicity, we consider an OWL-S atomic service with an arbitrary number of preconditions and results, with a single effect[3]. This implies that there are as many *inConditions* as *Effects*. Supposing that *inConditions* and *Effects* are formed by single SWRL atoms, the proposed mapping with the DL action formalism is:

pre :$\{pre_1(at_1), \ldots, pre_1(at_l), pre_2(at_1), \ldots, pre_2(at_m), pre_t(at_1), \ldots, pre_t(at_n)\}$
$post$:$\{inCond_a/Effect_a, \ldots, inCond_z/Effect_z\}$

with $pre_t(at_n)$ the n-th atom of the t-th OWL-S precondition, and $inCond_z/Effect_z$ the z-th conditional post-condition. The traceability amongst the atoms in the action pre and the service preconditions is guaranteed by the corresponding SWRL rule. The formalism in [4] allows to encode only simple inCondition and Effect of the form specified above ($A(a)$, $\neg A(a)$, $r(a, b)$, $\neg r(a, b)$).

4 A Mapping Example

To describe the proposed mapping, we use a simplified version of the OWL-S Atomic Service *ExpressCongoBuy*[4], based on a subset of inputs (renamed here to improve the readability of service conditions), and with simplified *inConditions* and *Effects* so that the limitation imposed by the definition of action in [4] are respected. The service features are described in Figure 1,a); the SWRL rules representing the service are depicted in Figure 1,b); finally, the resulting service described in terms of actions is described in Figure 1,c). When OWL-S atomic services present a single result, the inCondition can be omitted. In this case the service effect will be an unconditional postcondition.

[3] Outputs are not logical conditions and can be omitted because do not generate KB variations. Generally, built-in atoms are not a single OWL class or property; they are left as future work.

[4] `http://www.ai.sri.com/daml/services/owl-s/1.2/CongoProcess.owl`

a) ExpressCongoBuy Service Characteristics

INPUTS:	Type
x : $ECBCreditCardType$	$CreditCardType$
y : $ECBBookISBN$	$profileHierarchy : Book$
z : $ECBSignInInfo$	$SignInData$
t : $ECBCreditCardNumber$	$xsd : decimal$
OUTPUT:	
o : $ECBOutput$	$ECBOutputType\,[ECBOT]$
LOCAL variables (SWRL variables):	
u : $ECBAcctID$	$AcctID\,[AID]$
v : $ECBCreditCard\,[ECBCC]$	$CreditCard$
PRECONDITIONS:	
$precond_1$: $validity(?v, Valid) \wedge cardNumber(?v, ?t)\,[V(?v, Valid) \wedge CN(?v, ?t)]$	
$precond_2$: $hasAcctID(?z, ?u)\,[hasAID(?z, ?u)]$	
RESULT 1:	
inCondition : $OutOfStockBook(?y)\,[OSBook(?y)]$	
Effect : $FailureNotification(?o)\,[FN(?o)]$	
RESULT 2:	
inCondition : $InStockBook(?y)\,[ISBook(?y)]$	
Effect : $OrderShippedAcknowledgement(?o)\,[OSA(?o)]$	

* The namespace is the base one for the service. $ExpressCongoBuy$ is shortened to ECB for readability. Further shortenings are in [...].

b) The service described with SWRL rules

1 : $[inputs] \wedge hasAID(?z, ?u) \wedge CN(?v, ?t) \wedge V(?v, Valid) \wedge safe(?y, ?o)$
$\wedge\, OSBook(?y) \;\rightarrow\; ECBOT(?o) \;\wedge\; FN(?o)$

2 : $[inputs] \wedge hasAID(?z, ?u) \wedge CN(?v, ?t) \wedge V(?v, Valid) \wedge safe(?y, ?o)$
$\wedge\, ISBook(?y) \;\rightarrow\; ECBOT(?o) \;\wedge\; OSA(?o)$

*$safe$ is an always true property needed to guard the SWRL safety condition.

c) The resulting service described in terms of actions

pre : { $CN(?v, ?t), ECBCC(?v, Valid), hasAID(?z, ?u)$ }
post : { $holds(?y, ?o), OSBook(?y)$ / $FN(?o), ISBook(?y)$ / $OSA(?o)$ }

Fig. 1. ExpressCongoBuy mapping with Action formalism

The proposed mapping allows to exploit the theoretical results about projection and executability working in DL. In detail, executability and projection together grant that a composite action, i.e., a service or sequence of services, can be executed to completion, thus obtaining the original goal. However, this is subject on each action in the sequence being *consistent* with the knowledge base as it evolves. When this is not the case, in order to obtain the goal, either a new plan must be computed, or a solution to inconsistent actions must be designed, so that the knowledge base can be modified to be consistent with the actions. Let us introduce an example of inconsistency due to the *Effects* of rules being applied, based on the service in Figure 1. The service is invoked twice, with $?o$ matched to an individual, i_o. The first invocation generates *OrderShippedAcknowledgement(i_o)* assertion, while the second generates *FailureNotification(i_o)*. Both assertions end up in the knowledge base. If these two classes are disjoint in the ontology defining them, the two type of assertions would force i_o to belong to a class and its complement, therefore making the knowledge base inconsistent. This is an example of the kind of inconsistencies arising from lack of retraction primitives that this paper aims at addressing.

5 Contexts for Incompatibility Management

In this work, the notion of *contexts* is used to justify a mechanism able to handle *actions* which are inconsistent with respect to a knowledge base. Recalling the definition 2 in Milicic et al. [4], given $\mathcal{T}$ an acyclic TBox, $\alpha = (pre, post)$ an atomic action for $\mathcal{T}$, and $\mathcal{I}$, $\mathcal{I}'$ models of $\mathcal{T}$ respecting the unique name assumption (UNA) and sharing the same domain and interpretation of all individual names. We say that α may transform

$\mathcal{I}$ to $\mathcal{I}'$ ($\mathcal{I} \Rightarrow_{\alpha}^{\mathcal{T}} \mathcal{I}'$) iff, for each primitive concept A and role name r, a change of the interpretation is defined as follows:

$$A^{\mathcal{I}'} := (A^{\mathcal{I}} \sqcup A^{\mathcal{I}'}_{A(a)}) \setminus A^{\mathcal{I}'}_{\neg A(a)}, r^{\mathcal{I}'} := (r^{\mathcal{I}} \sqcup r^{\mathcal{I}'}_{r(a,b)}) \setminus r^{\mathcal{I}'}_{\neg r(a,b)}$$

where:

$$\begin{array}{lcl} A^{\mathcal{I}'}_{A(a)} & = & \{a^{\mathcal{I}} \mid \varphi/A(a) \in post \wedge \mathcal{I} \models \varphi\} \\ A^{\mathcal{I}'}_{\neg A(a)} & = & \{a^{\mathcal{I}} \mid \varphi/\neg A(a) \in post \wedge \mathcal{I} \models \varphi\} \\ r^{\mathcal{I}'}_{r(a,b)} & = & \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/r(a,b) \in post \wedge \mathcal{I} \models \varphi\} \\ r^{\mathcal{I}'}_{\neg r(a,b)} & = & \{((a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/\neg r(a,b) \in post \wedge I \models \varphi\} \end{array}$$

The composite action $\alpha_1, \ldots \alpha_k$ may transform $\mathcal{I}$ to $\mathcal{I}'$ ($\mathcal{I} \Rightarrow_{\alpha_1,\ldots,\alpha_k}^{\mathcal{T}} \mathcal{I}'$) iff there are models $\mathcal{I}_0, \ldots, \mathcal{I}_k$ of $\mathcal{T}$ with $\mathcal{I} = \mathcal{I}_0, \mathcal{I}' = \mathcal{I}_k$, and $\mathcal{I}_{i-1} \Rightarrow_{\alpha}^{\mathcal{T}} \mathcal{I}_i$ for $1 \leq i \leq k$.
This definition does not cover the situation in which a postcondition generates an inconsistency due to the lack of retraction primitives, as in the example presented in Sect. 4. OWL contexts are introduced as a possible solution to this kind of problems. In particular, by identifying contexts with the portions of ABox it is possible to manage this kind of inconsistencies without modifying the definition of interpretation change. This is done by defining *context relations* that enable a dynamic partitioning of the knowledge base, simulating the retraction of conflicting assertions. The proposed solution is inspired to the work presented in [5], even though the work proposed in [6], if suitably adapted for the contexts, could be a valuable alternative. For our aims, the *content* of a context is a set of complete OWL axioms.

Definition 1 (Content of an OWL context). *Given a context C_{TX}, a signature $S = C \cup R \cup I$ where C are* concept names*, R are* role names*, and I are* individuals*, the content of C_{TX} consists of the union of:*

- *a TBox $\mathcal{T}_C$ whose concepts and nominals are included in $C \cup I$;*
- *a RBox $\mathcal{R}_C$ whose roles are included in R;*
- *an ABox $\mathcal{A}_C$ whose individuals are included in I.*

$\mathcal{T}_C$, $\mathcal{R}_C$ and $\mathcal{A}_C$ are expressed in OWL.

The *parameters* representation adopted is the one outlined in [5]:

Definition 2 (Parameter or Contextual Relation). *Given a context C_{TX}, a parameter P and a value X for P, the relation $P(C_{TX}, X)$ associates C_{TX} with X and is represented with the triple (C'_{TX}, P', X') where C'_{TX} and P' are URIs assigned to C_{TX} and P respectively, and X' is a RDF node identifying a resource, another context or a datatype value.*

Two *parameters* or *contextual relations* (CR) are defined as:

Definition 3 (EXTENDS Contextual Relationship). *Given contexts C_{TX1} and C_{TX2}, the expression C_{TX2} $EXTENDS$ C_{TX1} is interpreted as:*
the content *of C_{TX2} includes the* content *of C_{TX1}, and therefore transitively the* content *of any context D_i that C_{TX1} is declared to $EXTEND$.*

Definition 4 (INCOMPATIBLE Contextual Relationship). *Given contexts C_{TX1} and C_{TX2}, the expression $C_{TX2}\ INCOMPATIBLE\ C_{TX1}$ is interpreted as:*
called $\mathcal{K}$ the knowledge base containing the content *of C_{TX1} and C_{TX2}, $\mathcal{K}$ is inconsistent. The $INCOMPATIBLE$ relation is symmetric: $C_{TX2}\ INCOMPATIBLE\ C_{TX1}$ is equivalent to $C_{TX1}\ INCOMPATIBLE\ C_{TX2}$.*

These definitions allow to describe a situation where two contexts C_{TX2} and C_{TX3} extend a third context C'_{TX1} in two different, incompatible ways, i.e., C_{TX3} contains $\neg A(x)$ and C_{TX2} contains $A(x)$ (where A is defined in $\mathcal{T}_{C_{TX3}}$ and x is a named individual belonging to $\mathcal{A}_{C_{TX1}}$ and $\mathcal{A}_{C_{TX2}}$, but not necessarily to $\mathcal{A}_{C_{TX3}}$). C_{TX2} and C_{TX3} are incompatible, since any knowledge base containing both $\neg A(x)$ and $A(x)$ will be inconsistent. With reference to the definition for change of interpretation given at the beginning of this section, C_{TX2} and C_{TX3} correspond to $A^{\mathcal{I}'}_{A(a)}$ and $A^{\mathcal{I}'}_{\neg A(a)}$, and therefore implement the operator for the change of interpretation. This implies that the change needed to fulfil the goal of the action sequence (i.e. the right context and therefore the right change of interpretation) is chosen as the actual state of the knowledge base. It is straightforward that a means to know when to apply contextualization is needed. A possible strategy can be summarized as follows. We keep track of the current model and use a heuristic function $t(e)$ (where e is a type or role assertion) to estimate whether the change produced by e affects a constrained part of the model, therefore possibly producing inconsistencies, or an unconstrained part, in which case the change can be applied safely.

6 Executability and Projection Evaluation by Means of Contexts

This section presents the algorithm for verifying the validity of a sequence of services. Suppose that we have a set of sequences of SWRL rules that achieve an input goal. Then the procedure can be summarized as follows:

- For each sequence S_i, every SWRL rule is mapped with an action, as described in Sect. 3, resulting in the sequence of actions AS_i;
- $Executability$ and $projection$ are then used to determine whether the sequence of actions $AS_i = \{\alpha_1, \ldots, \alpha_n\}$ is executable and produces the required effect.

Assuming that the sequence AS_i is executable in $\mathcal{A}$ w.r.t. $\mathcal{T}$, the $projection$ verifies that the assertion φ is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff for all the models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ and for all $\mathcal{I}'$ with $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_n} \mathcal{I}'$, $\mathcal{I}' \models \varphi$.
Consider a sequence S containing an action α_i, e.g., the action described in Figure 1. For this action, the postcondition φ_i/ψ_i is $OSBook(?y)\ /\ FN(?o)$.
When the projectability verification reaches α_i, it is evaluated according to the heuristic $t(e)$ delineated in Sect. 5. If t does not detect the need for a contextualization of the knowledge base, i.e., no potential inconsistency is detected, the projection can continue to $i+1$. When, instead, a possible inconsistency is detected by t, e.g., because $OSA(?o)$ has been asserted previously by an action α_j with $j < i$, the proposed contextualization will be applied, creating alternate ABoxes for the conflicting assertions, while the TBox will be left untouched. With reference to the example given in sect. 4, the Knowledge base at $i-1$: $\mathcal{K}_{i-1} = \mathcal{T} \sqcup C_{i-1}$ produce the following contexts and context relations:

- Content of context $C_{TXi-1} = \mathcal{A} \sqcup \{OSA(?o)\}$
- Content of context $C'_{TXi-1} = C_{TXi-1} \setminus \{OSA(?o)\}$
- Content of context $C'_{TXi} = \{OSA(?o)\}$
- Content of context $C''_{TXi} = \{FN(?o)\}$
- C'_{TXi} EXTENDS C'_{TXi-1}
- C''_{TXi} EXTENDS C'_{TXi-1}
- C'_{TXi} INCOMPATIBLE C''_{TXi}

If the projection requires one or more contextualizations, it is necessary to verify the executability in the new contexts. Since C''_{TXi} is the new KB for the sequence $\alpha_i, \ldots, \alpha_n$, we need to verify whether the preconditions belonging to $\alpha_{i+1}, \ldots, \alpha_n$ actions hold in C''_{TXi}. This is equivalent to split in two parts the actions sequence: the executability of the new sequence $S'' = \{\alpha_{i+1}, \ldots, \alpha_n\}$ must be verified against $\mathcal{A}'$ w.r.t. $\mathcal{T}$, where $\mathcal{A}' = C''_{TXi}$. If this new sequence is not executable, the original sequence S will not be considered valid, since it cannot be automatically executed. In fact, its execution would either result in an inconsistent knowledge base (if executed without contextualization) or some preconditions would not be verified when the services mapped by S' are executed, resulting in unreliable behavior of the system.

7 Related Work

In [7], the notion of contexts is used for the operation of SWS composition. The adopted meaning of context, i.e. any information that can be used to characterize the situation of an entity, comes from context-aware computing [8]. The dynamic description logic adopted in [9] uses static and dynamic context information in order to compose Web services adapting user, provider and broker contexts. A dynamic DL knowledge base has, in addition to TBox and ABox, an ActionBox that contains assertions about actions. The reasoning tasks about actions proposed, i.e. executability and projection, are based on [4]. However, all the approaches above do not provide a concrete mapping with OWL-S services, and the management of non-monotonic problematic situations that can be caused by service *Effects* is missing as well. A representative work on OWL contexts is [10]. It proposes a modification of OWL semantics in order to contextualize ontologies, and formalizes a particular type of rules, called bridge-rules, allowing the creation of explicit mappings for the contexts management. Furthermore, it handles, among the others, the management of localized inconsistency and its propagation in this type of contexts. The OWL contexts we proposed in this paper do not require modifications to OWL semantics, nor the use of rules for their management.

8 Conclusions

In this paper we analyzed the current limitations that prevent the implementation of an OWL-S based service management platform. We observed that even removing or mitigating the incompatibility between the languages for specifying domain ontologies and those for annotating services that operate in such domains, the problem of static

ontology against the dynamic services remains. We proposed the adoption of a theoretical framework that accomodates dynamicity inside DL [4]. However, such framework requires the implementation of non standard operations on the interpretations of Knowledge base. In particular, these operations allow for overcoming the absence of retraction primitives and avoiding inconsistencies due to DLs monotonicity property. We describe a possible solution based on the notion of contexts for an OWL knowledge base. As future work, we plan the integration of SWRL built-in atoms and design of a concrete implementation of heuristic function.

Acknowledgments. This work fulfills the objectives of the PON 02_00563_3489339 project "Puglia@Service - Internet-based Service Engineering enabling Smart Territory structural development" funded by the Italian Ministry of University and Research (MIUR).

References

1. Horrocks, I., Patel-Schneider, P.F., Bechhofer, S., Tsarkov, D.: OWL rules: A proposal and prototype implementation. J. of Web Semantics 3, 23–40 (2005)
2. Motik, B., Sattler, U., Studer, R.: Query answering for owl-dl with rules. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 3, 41–60 (2005)
3. Redavid, D., Ferilli, S., Esposito, F.: Towards dynamic orchestration of semantic web services. T. Computational Collective Intelligence 10, 16–30 (2013)
4. Milicic, M.: Planning in action formalisms based on dls: First results. In: Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Turhan, A.Y., Tessaris, S. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 250. CEUR-WS.org (2007)
5. Stoermer, H., Bouquet, P., Palmisano, I., Redavid, D.: A context-based architecture for RDF knowledge bases: Approach, implementation and preliminary results. In: Marchiori, M., Pan, J.Z., de Sainte Marie, C. (eds.) RR 2007. LNCS, vol. 4524, pp. 209–218. Springer, Heidelberg (2007)
6. Rosati, R.: On the complexity of dealing with inconsistency in description logic ontologies. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011, Barcelona, Catalonia, Spain, July 16-22, pp. 1057–1062. IJCAI/AAAI (2011)
7. Niu, W., Shi, Z., Chang, L.: A context model for service composition based on dynamic description logic. In: Shi, Z., Mercier-Laurent, E., Leake, D. (eds.) Intelligent Information Processing IV. IFIP, vol. 288, pp. 7–16. Springer, Boston (2008)
8. Schilit, B.N., Adams, N., Want, R.: Context-aware computing applications. In: Proceedings of the Workshop on Mobile Computing Systems and Applications, pp. 85–90. IEEE Computer Society (1994)
9. Chang, L., Shi, Z., Qiu, L., Lin, F.: Dynamic description logic: Embracing actions into description logic. In: Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Turhan, A.Y., Tessaris, S. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 250. CEUR-WS.org (2007)
10. Serafini, L., Borgida, A., Tamilin, A.: Aspects of distributed and modular ontology reasoning. In: Kaelbling, L.P., Saffiotti, A. (eds.) IJCAI, pp. 570–575. Professional Book Center (2005)