# DDL: EMBRACING ACTIONS INTO SEMANTIC WEB

He Huang
*Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, P.O. Box 2704-28, Beijing 100080 China*
*Graduate School of the Chinese Academy of Sciences, Beijing, China*
huangh@ics.ict.ac.cn

Zhongzhi Shi
*Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, P.O. Box 2704-28, Beijing 100080 China*
shizz@ics.ict.ac.cn

Jianwu Wang
*Institute of Computing Technology, Chinese Academy of Sciences, P.O.Box 2704-28, Beijing 100080 China*
*Graduate School of the Chinese Academy of Sciences, Beijing, China*
wjw@software.ict.ac.cn

Rui Huang
*Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, P.O. Box 2704-28, Beijing 100080 China*
*Graduate School of the Chinese Academy of Sciences, Beijing, China*
huangr@ics.ict.ac.cn

**Abstract** Service description usually presumes a representation of the world model. The Description Logic (DL) is an efficient way for representing the world model, esp. on Semantic Web, because of its framework, decidable reasoning, and popularity. DL can bring structure to services, but only DL itself is inadequate for modelling dynamic aspect of Web services. In this paper, Dynamic Description Logic (DDL) is proposed to combine DLs with action formalisms. The interaction between actions and the DL-based world model is embodied in two aspects. On one hand, DL knowledge base provides knowledge and information for the reasoning on actions; on the other hand, the information stored in DL knowledge base is changed by the execution of actions. In DDL, two basic reasoning tasks are defined to check precondition and effects of actions. Based on the relationship between DDL and a transition system, a reasoning support for DDL is also given by translating actions into logic programs. By the combination of DLs and actions, DDL brings a better view of how services impact the world, facilitates interoperation between services, and enables the reuse of already available algorithms and engines for service reasoning. Thus, it can provide a logical way for embracing actions into Semantic Web.

**Keywords:** Dynamic Description Logic, action formalism, Semantic Web, Web Services

## 1. Introduction

Now we are witnessing the proliferation of service offers accumulated on the Web. As the next generation of the Web, the Semantic Web [Berners-Lee et al., 2001] should support reasoning on not only semantic content but also services. Description Logics (DLs) [Baader et al., 2002] are a family of logic-based knowledge representation languages. Several DL-based ontology languages (e.g., OWL [Patel-Schneider et al., 2004]) are used to develop ontologies of Web services, e.g. OWL-S [Coalition, 2004]. However, DL by itself is inadequate for service reasoning. DLs were originally designed for representing only static knowledge. Without notion of modelling changes, DLs are unable to model the dynamic aspect of services (i.e., what changes of the world they cause under certain conditions), and neither to support reasoning on a series of such changes.

Based on our previous work on Dynamic Description Logics (DDL)[Shi et al., 2005], we extends DDL to model the dynamic aspect of services and support reasoning on services. DDL is designed to combine DLs with action formalisms by taking consideration of the following two aspects. On one hand, Web services can be abstracted as actions with preconditions and effects in the sense that services impact the world by changing the state of the world [McIlraith et al., 2001]. Thus, theories about actions in AI communities can be applied to model dynamic feature of Web service. On the other hand, service description usually presumes a representation of the world model, where the preconditions and effects of services root in [Ponnekanti and Fox, 2002]. DLs are suitable for the world model description, esp. on the Semantic Web, in that DLs can describe structural knowledge in a formal and network-based way, and provide decidable reasoning, too.

In DDL, the structure of the world is captured in TBox and the current information (or facts) about the world is stored in ABox. A formal meaning is given to the interaction between actions and the DL-based world model. Preconditions and effects of an action are defined by using vocabulary defined in TBox. The execution of an action causes the changes of facts stored in ABox and the semantics of the action is interpreted as interpretation transformation that corresponds to the changes. A group of action constructors are defined for forming complex actions out of primitive ones. Two basic reasoning problems (i.e. executability and projection) are discussed. A reasoning support for DDL is also given by translating actions into logic programs. By the combination, DDL provides a logical way for embracing actions into Semantic Web. It also brings several advantages to Web services on Semantic Web, such as a better view of how services impact the world, interoperation between services, and the reuse of algorithms and engines available for service reasoning.

Due to space limitation, we do not introduce the DLs in this paper. Interested readers can refer to [Baader et al., 2002] for review. The rest of this paper is organized as follows. The DDL formalism is described in section 2. A reasoning support for DDL is given in section 3. Related works are discussed in section 4. A conclusion and future research are given in section 5.

## 2. Dynamic Description Logic Formalism

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL knowledge base composed of an acyclic TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. In this paper, we restrict ourselves on *acyclic* TBox. Based on it, we introduce the DDL formalism by defining the syntax and semantics of actions. Two reasoning tasks are described.

### Syntax and Semantics of Actions

Actions without variables are called *ground actions*. Actions can be parameterized by introducing variables in place of object names. Ground actions can be viewed as actions where parameters have already been instantiated by object names, while parametric actions should be viewed as a compact representation of all its ground instances. For simplicity, we concentrate on *ground actions*. Actions can be nondeterministic, but in DDL we only consider deterministic actions.

Definition 1 (Atomic Action[Shi et al., 2005]) *Given $\mathcal{K}$, an atomic action description based on $\mathcal{K}$ is in the form of $\mathbb{A} = (P_{\mathbb{A}}, E_{\mathbb{A}})$, where*

- *$\mathbb{A}$ is the action name;*
- *the pre-condition, $P_{\mathbb{A}}$, is a finite set of ABox assertions;*
- *the effects of the action, $E_{\mathbb{A}}$, is a finite set of pairs of the form $b_i/h_i$, where $b_i$ is a finite set of ABox assertions, while $h_i$ is a literal of an assertion, e.g., $C(a)$, $R(a, b)$, $\neg C(a)$, or $\neg R(a, b)$.*

An atom action is specified by first stating the preconditions under which the action is applicable. Secondly, one must specify how the action impacts the world with effects $b_i/h_i$. $b_i/h_i$ describes under the condition $b_i$ (called body) doing the action results in the addition of $h_i$ (called head) if $h_i$ is in the positive form, or the remove of $\neg h_i$ if $h_i$ is in the negative form. According to the law of inertia, only those facts that are forced to change by the effects should be changed by the performance of the action [Reiter, 2001]. Given two different atomic actions, we say that they are similar if they differ only by the heads in their effects. This paper focuses only on those actions that are not similar.

Given a declarative specification of the atomic actions, a complex action can be specified in terms of atomic ones.

DEFINITION 2 (ACTIONS[SHI ET AL., 2005]) *Let $N_{\mathbb{A}}$ be a set of atomic action names, an action is in the form of:*

$$\mathbb{A}, \mathbb{B} \rightarrow \mathbb{A}_S \,|\, \mathbb{A}; \mathbb{B} \,|\, \mathbb{A} \cup \mathbb{B} \,|\, \mathbb{A}^* \,|\, \psi\,? \tag{1}$$

An action can be defined as either an atomic action $\mathbb{A}_S$, or a complex action composed of these defined atomic actions by using constructors. Sequential composition of actions, $\mathbb{A}; \mathbb{B}$, means "Do $\mathbb{A}$ and then do $\mathbb{B}$". Non-deterministic choice of actions, $\mathbb{A} \cup \mathbb{B}$, means "Do either $\mathbb{A}$ or $\mathbb{B}$". Iteration of action, $\mathbb{A}^*$, means "iteratively do $\mathbb{A}$ finitely many (including zero) times". Test of formula, $\psi$?, means the test of current truth value of $\psi$, i.e., $\psi? \equiv (\{\psi\}, \{\})$, where $\psi$ is a DL formula.

The widely used conditional choice and loops can be defined in terms of these constructs:

$$\begin{aligned} \texttt{if } \psi \texttt{ then } \mathbb{A} \texttt{ else } \mathbb{B} &\equiv (\psi\,?\,; \mathbb{A}) \cup (\neg\psi\,?\,; \mathbb{B}) \\ \texttt{while } \psi \texttt{ do } \mathbb{A} \texttt{ endWhile } &\equiv (\psi\,?\,; \mathbb{A})^*\,; \neg\psi\,? \end{aligned}$$

In this paper, we focus on acyclic actions, which means an action can not be composed of itself or can not appear on the right side of its own definition. Given an acyclic action, an action can be always expanded into a sequence that is composed with only atomic actions.

Now, we define the semantic of the action by showing how it makes an interpretation transformed into another.

DEFINITION 3 (ORDERED INTERPRETATIONS[BAADER ET AL., 2005]) *Given $\mathcal{K}$ and an interpretation $\mathcal{I}$ that $\mathcal{I} \models \mathcal{K}$. The performance of $\mathbb{A}$ in $\mathcal{K}$ results in a new interpretation $\mathcal{I}'$. We say that $\mathcal{I}'$ is the $\mathbb{A}$-induced ordered interpretation to $\mathcal{I}$, denoted $\mathcal{I} \preceq_{\mathbb{A}} \mathcal{I}'$, if $\mathcal{I}'$ is obtained through following steps:*

- *if $\mathbb{A}$ is an atomic action and $\mathbb{A} = (P_{\mathbb{A}}, E_{\mathbb{A}})$,*
  - *for each primitive concept $A$ in $\mathcal{T}$,*

$$\begin{aligned} A^{\mathcal{I}'} = A^{\mathcal{I}} &\cup \{a^{\mathcal{I}} \,|\, \forall(\varphi/A(a)) \in E_{\mathbb{A}}, \mathcal{I} \models \varphi\} \\ &\setminus \{a^{\mathcal{I}} \,|\, \forall(\varphi/\neg A(a)) \in E_{\mathbb{A}}, \mathcal{I} \models \varphi\} \end{aligned} \tag{2}$$

– *for each role $R$ in $\mathcal{T}$,*

$$R^{\mathcal{I}'} = R^{\mathcal{I}} \cup \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \,|\, \forall(\varphi/R(a,b)) \in E_{\mathbb{A}}, \mathcal{I} \models \varphi\} \\ \setminus \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \,|\, \forall(\varphi/\neg R(a,b)) \in E_{\mathbb{A}}, \mathcal{I} \models \varphi\} \quad (3)$$

– *for each defined concept $C$, it is expanded into $C'$ which contains only primitive concepts. The interpretation of $C$ is uniquely determined by the interpretation of the primitive concepts in $C'$.*

- *if $\mathbb{A} = \mathbb{B}\,;\mathbb{C}$, and $\exists\mathcal{I}'', \mathcal{I} \preceq_{\mathbb{B}} \mathcal{I}'' \wedge \mathcal{I}'' \preceq_{\mathbb{C}} \mathcal{I}'$;*
- *if $\mathbb{A} = \mathbb{B} \cup \mathbb{C}$, and $\mathcal{I} \preceq_{\mathbb{B}} \mathcal{I}' \vee \mathcal{I} \preceq_{\mathbb{C}} \mathcal{I}'$;*
- *if $\mathbb{A} = \mathbb{B}^*$, and $\forall\, n(\geq 0), \forall\, i(0 \leq i \leq n), \exists\mathcal{I}'', \mathcal{I} \preceq_{\mathbb{B}^i} \mathcal{I}'' \wedge \mathcal{I}'' \preceq_{\mathbb{B}^{n-i}} \mathcal{I}'$;*
- *if $\mathbb{A} = \psi\,?$, and $\mathcal{I} \preceq_{\mathbb{A}} \mathcal{I}$.*

Clearly, $\mathcal{I}$ and $\mathcal{I}'$ share the same domain and individual name interpretation, and $\mathcal{I}'$ is determined by $\mathbb{A}$, $\mathcal{T}$ and $\mathcal{I}$.

## Reasoning about Actions

Before trying to perform an action, it is needed to check whether it is executable in the current state[Baader et al., 2005], i.e., whether all necessary preconditions are satisfied.

DEFINITION 4 (EXECUTABILITY) *Given $\mathcal{K}$ and an interpretation $\mathcal{I}$ that $\mathcal{I} \models \mathcal{K}$, an action $\mathbb{A}$ is said to be executable in $\mathcal{I}$, denoted $poss(\mathbb{A}, \mathcal{I})$,*

- *if $\mathbb{A}$ is an atomic action, $\mathbb{A} = (P_{\mathbb{A}}, E_{\mathbb{A}})$, and $\mathcal{I} \models P_{\mathbb{A}}$;*
- *if $\mathbb{A} = \mathbb{B}\,;\mathbb{C}$, and $poss(\mathbb{B}, \mathcal{I}) \wedge \mathcal{I} \preceq_{\mathbb{B}} \mathcal{I}' \wedge poss(\mathbb{C}, \mathcal{I}')$;*
- *if $\mathbb{A} = \mathbb{B} \cup \mathbb{C}$, and $poss(\mathbb{B}, \mathcal{I}) \vee poss(\mathbb{C}, \mathcal{I})$;*
- *if $\mathbb{A} = \mathbb{B}^*$, and $\forall\, n(\geq 0), \forall\, i(0 \leq i \leq n), poss(\mathbb{B}^i, \mathcal{I}) \wedge \mathcal{I} \preceq_{\mathbb{B}^i} \mathcal{I}' \wedge poss(\mathbb{B}^{n-i}, \mathcal{I}')$;*
- *if $\mathbb{A} = \psi\,?$, and $\mathcal{I} \models \psi$.*

Since an action impacts the world, it is needed to check whether performing it can result in the desired effects, i.e., whether a formula that we want to make true really holds after the performance[Baader et al., 2005].

DEFINITION 5 (PROJECTION) *Given $\mathcal{K}$ and an interpretation $\mathcal{I}$ that $\mathcal{I} \models \mathcal{K}$, a DL formula $\varphi$ is a consequence of performing $\mathbb{A}$ in $\mathcal{I}$, denoted $[\mathbb{A}]\varphi$, iff $\mathcal{I} \preceq_{\mathbb{A}} \mathcal{I}' \wedge \mathcal{I}' \models \varphi$.*

Actually, executability and projection can be reduced into each other [Baader et al., 2005]. Therefore, we only consider the reasoning task of executability in section 3.

## 3. Reasoning Support for DDL

In DDL, $\mathcal{T}$ brings structure to the world model of an action domain. Each interpretation that models $\mathcal{T}$ denotes a state of the world. A set of actions defined on the world model depicts the dynamic features of the action domain. The execution of an action causes the interpretation transformation. In this context, DDL can be viewed as a transition system consisting of:

1 a set $W = \{\mathcal{I}_1, ..., \mathcal{I}_n\}$, where $\forall\, i\,(1 \leq i \leq n), \mathcal{I}_i \models \mathcal{T}$, and $\mathcal{I}_i$ is called a *state*;

2 a function $V : F \times W \rightarrow$\{true,false\}, where $F$ is the set of DL formulae and $V(\psi, \mathcal{I}_i)$ means whether $\mathcal{I}_i \models \psi$ is true or false;

3 a set $T \subseteq W \times S \times W$, where $S$ is the set of actions, and $\langle \mathcal{I}_i, \mathbb{A}, \mathcal{I}_j \rangle \in T$ denotes that $\mathcal{I}_i \preceq_{\mathbb{A}} \mathcal{I}_j$.

Logic Program (see, e.g., [Baral and Gelfond, 1994] for review) is an efficient way for representing a transition system [Lifschitz and Turner, 1999]. In order to provide reasoning support for DDL, we translate actions into a logic program. Based on this translation, the action reasoning in DDL can be transformed into computing answer sets on a logic program.

Several symbols are needed for the translation. There is a distinguished binary function symbol $do : S \times W \rightarrow W$; $do(\mathbb{A}, \mathcal{I})$ denotes the successor state to $\mathcal{I}$ resulting from performing the action $\mathbb{A}$. The qualification problem for actions is denoted by a predicate symbol $poss : S \times W$; $poss(\mathbb{A}, \mathcal{I})$ means that $\mathbb{A}$ is executable in $\mathcal{I}$. And another function $holds(\psi, \mathcal{I})$ asserts that a certain fact "holds" in a certain state, i.e., $\mathcal{I} \models \psi$.

Now we are ready for defining the translation $\pi$ from DDL to logic programming. We tailor the methods mentioned in [Gelfond and Lifschitz, 1993, Lifschitz and Turner, 1999] to do the translation.

## Translating Atomic Actions

Let $D$ be a domain description of DDL without similar actions. The program $\pi D$ will consist of four rules which are motivated by the "commonsense law of inertia". These rules are used to specify that DL formulae normally remain the same after performing an action.

$$holds(\psi, do(\mathbb{A}, \mathcal{I})) \leftarrow holds(\psi, \mathcal{I}), not\ Noninertial(\psi, \mathbb{A}, \mathcal{I}) \tag{4}$$

$$\neg holds(\psi, do(\mathbb{A}, \mathcal{I})) \leftarrow \neg holds(\psi, \mathcal{I}), not\ Noninertial(\psi, \mathbb{A}, \mathcal{I}) \tag{5}$$

$$holds(\psi, \mathcal{I}) \leftarrow holds(\psi, do(\mathbb{A}, \mathcal{I})), not\ Noninertial(\psi, \mathbb{A}, \mathcal{I}) \tag{6}$$

$$\neg holds(\psi, \mathcal{I}) \leftarrow \neg holds(\psi, do(\mathbb{A}, \mathcal{I})), not\ Noninertial(\psi, \mathbb{A}, \mathcal{I}) \tag{7}$$

Rules (4) and (5) state that, when a formula is known to be true (or false) in the past, generally it remains true (or false) after performing an action. Rules (6) and (7) state that, when a formula is known to be true (or false) after performing an action, generally it was true (or false) before the performance. The auxiliary predicate $Noninertial$ is used to show the abnormal cases.

The translation of an primitive action $\mathbb{A} = (P_{\mathbb{A}}, E_{\mathbb{A}})$ is, for all formulae $\psi_1, ..., \psi_n \in P_{\mathbb{A}}$,

$$poss(\mathbb{A}, \mathcal{I}) \leftarrow holds(\psi_1, \mathcal{I}), \ldots, holds(\psi_n, \mathcal{I}) \tag{8}$$

and for each condition $b_i/h_i (1 \leq i \leq m) \in E_{\mathbb{A}}$,

$$holds(h_i, do(\mathbb{A}, \mathcal{I})) \leftarrow poss(\mathbb{A}, \mathcal{I}), holds(b_i, \mathcal{I}) \tag{9}$$

$$holds(b_i, \mathcal{I}) \leftarrow poss(\mathbb{A}, \mathcal{I}), \overline{holds(h_i, \mathcal{I})}, holds(h_i, do(\mathbb{A}, \mathcal{I})) \tag{10}$$

$$\overline{holds(b_i, \mathcal{I})} \leftarrow poss(\mathbb{A}, \mathcal{I}), \overline{holds(h_i, do(\mathbb{A}, \mathcal{I}))} \tag{11}$$

$$Noninertial(|h_i|, \mathbb{A}, \mathcal{I}) \leftarrow poss(\mathbb{A}, \mathcal{I}), \overline{holds(h_i, \mathcal{I})}, holds(b_i, \mathcal{I}) \tag{12}$$

where $\overline{holds(x, y)}$ is the literal complementary to $holds(x, y)$ and $|x|$ means the absolute value of $x$. These rules have following meanings:

- Rule (8) allows us to prove the action is executable, if the preconditions $h_i$ $(1 \leq i \leq m)$ are satisfied.
- For each each condition $b_i/h_i (1 \leq i \leq m) \in E_{\mathbb{A}}$, when $\mathbb{A}$ is executable,
  - Rule (9) allow us to prove that $h_i$ will hold after performing $\mathbb{A}$ under the condition of $b_i$;
  - Rule (10) justifies if the truth value of $h_i$ has changed to true after performing $\mathbb{A}$, then we can conclude that $b_i$ was satisfied when $\mathbb{A}$ was performed.
  - Rule (11) allows us to conclude that $b_i$ was false from the fact that performing an action did not lead to $h_i$ to be true after performing the action.

- Rule (12) states $|h_i|$ is affected by performing $\mathbb{A}$. $Noninertial(|h_i|, \mathbb{A}, \mathcal{I})$ means that $|h_i|$ does not conform to the law of inertial. It disables the inertial rules (4 – 7) in the cases when $|h_i|$ can be affected by $\mathbb{A}$.

## Translating Complex Actions

The translation of a complex $\mathbb{A}$ is defined as:

- if $\mathbb{A} = \mathbb{B}; \mathbb{C}$, then

$$poss(\mathbb{A}, \mathcal{I}) \leftarrow poss(\mathbb{B}, \mathcal{I}), poss(\mathbb{C}, do(\mathbb{B}, \mathcal{I})) \tag{13}$$

- if $\mathbb{A} = \mathbb{B} \cup \mathbb{C}$, then

$$poss(\mathbb{A}, \mathcal{I}) \leftarrow poss(\mathbb{B}, \mathcal{I}) \tag{14}$$
$$poss(\mathbb{A}, \mathcal{I}) \leftarrow poss(\mathbb{C}, \mathcal{I}) \tag{15}$$

- if $\mathbb{A} = \mathbb{B}^*$, then $\forall n(n \geq 0)$

$$poss(\mathbb{A}, \mathcal{I}) \leftarrow poss(\mathbb{B}^1, \mathcal{I}), poss(\mathbb{B}^2, \mathcal{I}), ..., poss(\mathbb{B}^n, \mathcal{I}) \tag{16}$$

- if $\mathbb{A} = \psi?$, then

$$poss(\mathbb{A}, \mathcal{I}) \leftarrow poss(\psi, \mathcal{I}) \tag{17}$$

Through the translation $\pi$ on actions, the reasoning on DLL is transformed into computing answer set of a logic program. Of course, to compute the answer set needs to use DL reasoning for the DL fragment of DDL. The combination of DL reasoning and answer set semantics of logic programs for DDL reasoning, as well as the complexity of this hybrid reasoning, will be discussed in other papers.

## 4. Related Work

Motivated by the expressive limitations of DLs in non-structural knowledge representation, several methods (e.g., [Levy and Rousset, 1996, Horrocks et al., 2003]) have been proposed to combine DLs with rules. In these combinations, the rules can be viewed as constraints that the domain should satisfy. The rule-extended DLs still describe domain knowledge in a static sense. Thus, the representation of actions' functionalities is beyond the scope of them.

DLs were originally designed for representing only static knowledge. To take into account changes in time or under certain actions, it is natural to extend it by dynamics-related formalisms (e.g. modal logic, action formalisms, and temporal logic). In [Wolter and Zakharyaschev, 2000], a dynamic dimension was introduced via extending the DL with propositional dynamic logic (PDL) for representing and processing knowledge in dynamic application domain. In this method, actions were treated as modal operators, but no action description mechanism was given. In [Huang et al., 2005], an interval-based knowledge model was presented to bring structure to time-varying information by providing temporal constructs for concepts, relationships and integrity constraints, but no explicit action description was given. In [Baader et al., 2005], an action formalism based on DLs was presented to model dynamic features of Web services. The complexity of reasoning task was also analyzed according to different choice of DLs. But no other action operator except for sequential composition was defined, which weakens the capability of the formalism.

## 5. Conclusion and Future Work

In this paper, DDL is designed to combine DLs with action formalisms. The interactions between actions and the DL-based world model are embodied in two aspects. On one hand, DL reasoning provides basis for interpreting action semantics and checking executability and projection. On the other hand, actions are defined in terms of preconditions and effects by using vocabulary defined in DL, and executions of actions impact the DL knowledge bases by changing facts stored in ABox. We have shown that the interaction can be viewed as a transition system. Based on the observation, reasoning tasks in DDL can be translated into answer set computing in logic programs.

DDL provides a logical way for embracing actions into Semantic Web. And it can bring several advantages to Web Service applications, esp. on the Semantic Web. First, the framework of DLs makes a clear distinction between intensional knowledge in TBox and extensional knowledge in ABox. Thus, DDL enables a better view of how services impact the world by using DL to represent the world model. Second, DDL facilitates interoperation between services. Based on DLs, modelling services can reuse terms of those already existing domain ontologies, and thus provide services with shared knowledge. Third, the reasoning support for DDL enables the reuse of those efficient and scalable algorithms and engines available for DL reasoning and logic programming, and combine them for Web service composition, which is also our future work.

## Acknowledgements

This work has been funded by the National Basic Research Priorities Programme (No.2003CB317004) and the National Science Foundation of China (No.90604017).

## References

[Baader et al., 2005] Baader, F., Lutz, C., Milicic, M., Sattler, U., and Wolter, F. (2005). A description logic based approach to reasoning about web services. In *Proceedings of the WWW 2005 Workshop on Web Service Semantics (WSS2005)*, Chiba City, Japan.

[Baader et al., 2002] Baader, Franz, Calvanese, Diego, McGuinness, Deborah, Nardi, Daniele, and Patel-Schneider, Perter F. (2002). *The Description Logic Handbook: Theory, Implementation and Applciations*. Cambridge University Press.

[Baral and Gelfond, 1994] Baral, Chitta and Gelfond, Michael (1994). Logic programming and knowledge representation. *Journal of Logic Programming*, (19/20):73–148.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):34–43.

[Coalition, 2004] Coalition, The OWL-S (2004). Owl-s 1.1 release. http://www.daml.org/services/owl-s/1.1/.

[Gelfond and Lifschitz, 1993] Gelfond, Michael and Lifschitz, Vladimir (1993). Representing action and change by logic programs. *Journal of Logic Programming*, 17(2,3,4):301–321.

[Horrocks et al., 2003] Horrocks, Ian, Patel-Schneider, Peter F., Boley, Harold, Tabet, Said, Grosof, Benjamin, and Dean, Mike (2003). Swrl: A semantic web rule language combining owl and ruleml. (Version 0.5).

[Huang et al., 2005] Huang, He, Shi, Zhongzhi, He, Xiaoxiao, and Qiu, Lirong (2005). An interval-based knowledge model and query language for temporal information. In *Proceedings of Eighth Pacific Rim International Workshop on Multi-Agents (PRIMA'05)*, pages 401–415.

[Levy and Rousset, 1996] Levy, Alon Y. and Rousset, Marie-Christine (1996). CARIN: A representation language combining horn rules and description logics. In *European Conference on Artificial Intelligence*, pages 323–327.

[Lifschitz and Turner, 1999] Lifschitz, Vladimir and Turner, Hudson (1999). Representing transition systems by logic programs. In *Logic Programming and Non-monotonic Reasoning (LPNMR'99)*, number 1730 in Lecture Notes in AI (LNAI), pages 92–106.

[McIlraith et al., 2001] McIlraith, S., Son, T.C., and Zeng, H. (2001). Semantic web services. *IEEE Intelligent Systems. Special Issue on the Semantic Web*, 16(2):46–53.

[Patel-Schneider et al., 2004] Patel-Schneider, Peter F., Hayes, Patrick, and Horrocks, Ian (2004). Owl web ontology language semantics and abstract syntax. *W3C Recommendation*.

[Ponnekanti and Fox, 2002] Ponnekanti, S.R. and Fox, A. (2002). Sword: A developer toolkit for web service composition. In *Proceedings of the 11th World Wide Web Conference*.

[Reiter, 2001] Reiter, R. (2001). *Knowledge in Action*. MIT Press.

[Shi et al., 2005] Shi, Zhongzhi, Dong, Mingkai, Jiang, Yuncheng, and Zhang, Haijun (2005). A logic foundation for the semantic web. *Science in China, Series F Information Sciences*, 48(2):161–178.

[Wolter and Zakharyaschev, 2000] Wolter, Frank and Zakharyaschev, Michael (2000). Dynamic description logics. In Segerberg, K., de Rijke, M., Wansing, H., and Zakharyaschev, M., editors, *Advances in Modal Logic*, volume 2. CSLI Publications.