Chapter 1

Introduction

1.1 Motivation and Goals

The World Wide Web (WWW) is a computer network where data is shared mainly for human consumption. Web contents are *visually* marked up by languages such as HTML, CSS, etc. The Web has been tailored for human consumption. The usefulness of the Web is limited by the fact that information cannot be easily understood and processed by machines.

Recent advances of XML [108] technology have separated the markup of contents of information from its layout. XML's characteristics, such as the separation of concerns, strict syntax well-formedness and the ability to allow user-defined tags permit for greater flexibility. However, with no mutually-agreed meaning for tag names, it is hard for information to be shared across organizational boundaries.

Proposed by Tim Berners-Lee *et al*, the Semantic Web [8] is a vision to extend the current World Wide Web so that Web resources are given well-defined, content-related and mutually-agreed meaning. The Semantic Web aims at realizing the full potential

of the Web by enabling software agents (intelligent software on the Web) to understand, process and aggregate information autonomously and collaboratively.

The realization of this vision depends on the ability to semantically markup Web resources, including both static data and dynamic Web services, by ontologies. Ontologies are formal specifications of conceptualizations [34]. Building on mature technologies such as XML, Unicode and URI (Uniform Resource Identifier) [7], the ontology languages are positioned in a layered "cake", as depicted in Fig. 1.1 by Tim Berners-Lee.

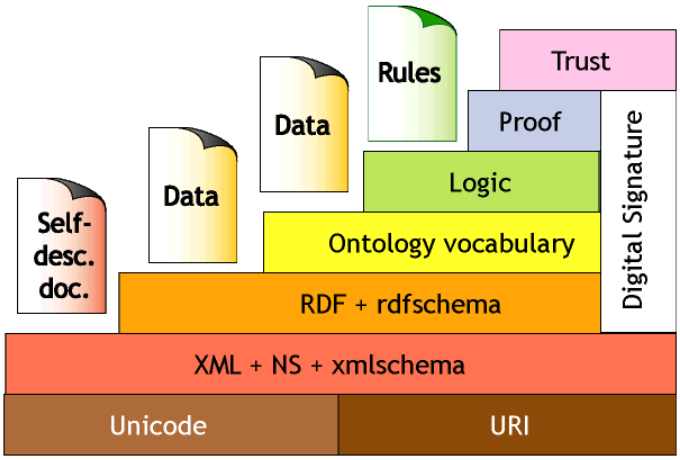

Figure 1.1: Generic architecture of the Semantic Web

Resource Description Framework (RDF) [68] and RDF Schema [17] are the foundation of the Semantic Web stack. They provide the core vocabularies and structure to describe Web resources. Based on RDF Schema and description logics (DLs) [74], the Web Ontology Language (OWL) [49] was developed and it provides more vocabulary for describing resources. Briefly, Web resources are categorized as *classes*, each of which holds a set of *instances*, pairs of which are related by *properties*.

Software agents' ability of autonomously understanding, processing and aggregating information builds on the decidability of the core ontology languages of the Semantic

Web. It is for this reason that DAML+OIL [101] and (a subset of) OWL were designed to be decidable [46, 40]. This is achieved by limiting their expressivity.

This design decision has made possible the construction of fully automated reasoning engines for ontologies written in these languages. However, certain desirable properties of resources cannot be represented by these languages due to the limited expressivity. This is mainly exhibited in the following two areas: expressivity limitation of the DL against first-order logic and the the dynamic nature of Web services.

Description logics are a very important knowledge representation formalism with a formal and rigid logical basis. They are a subset of first-order logic (FOL) [58] by carefully selecting only certain features to include. By limiting their expressivity, DLs are made decidable so that core reasoning services, namely concept subsumption, satisfiability and instantiation, can be solved in full automation. Being based on DL, ontology languages such as DAML+OIL and OWL are not expressive enough for certain complex ontology-related properties to be represented in these languages.

For example, consider the scenario of an ticket booking agent on the Semantic Web. It is very natural to express such a property that it should not book two tickets for any client with the durations of the two tickets overlap. Allowing booking only one ticket for a client is a possible, but overly restrictive solution. It is thus highly desirable that this information can be explicitly stated in the ontology and verified by reasoners.

In the light of this, the OWL Rules Language, (ORL) [47] (and its later version, the Semantic Web Rules Language (SWRL) [48]), a rules extension to OWL, was proposed to add Horn-style rules to OWL. Although SWRL extends the expressiveness of OWL, it is still limited in expressing certain properties. The correctness of these properties may, as we will see later in Chapter 2, have a significant impact on the validity of the ontology. Hence, the expression and verification of these properties are very

important.

Formal methods [16, 12, 11] have made significant development [41, 100, 62] and received much attention in both academia and industries. Z [89, 107] is a formal specification language designed to model system data and states. It is based on ZF set theory and first-order predicate logic. Therefore, Z is more expressive than ontology languages and it allows the specification of complex constraints which is not available in ontology languages. There are tools developed to support it. Z/EVES [84] is one such interactive proof tool for checking and reasoning about Z specifications.

Alloy [54], originally developed as a lightweight modeling language, is essentially aimed at automated analysis. Its design is influenced by Z but is less expressive. Alloy Analyzer [55] is a fully-automated tool for analyzing Alloy specifications with special model checking features, which are helpful to trace the exact source of errors.

Our earlier works [24, 27] showed that data-oriented formal methods and tools, e.g., Z/EVES and Alloy Analyzer, are capable of reasoning about ontologies. We also noticed the complementary reasoning capabilities among Z/EVES, Alloy Analyzer and Semantic Web reasoners such as FaCT++ [98] and RACER [36]. This motivated us to propose a combined approach [23] to using these tools in conjunction so that the synergistic reasoning power of these tools can be harnessed. By applying these tools systematically to an ontology, not only can we uncover more errors than using any one of them alone, inconsistencies can also be corrected more easily and precisely.

The effectiveness of the above combined approach relies on the soundness of the transformation from DAML+OIL/OWL ontologies to Z specifications. As these languages have different semantical bases, a higher-level device that is able to abstract and represent the underlying logics of DAML+OIL/OWL and Z is necessary to prove the soundness of the transformation. The notion of institutions [31] was introduced to formalize the concepts of "logical systems". Institutions provide a means of reasoning

about software specifications regardless of the logical system. We find the concept of institutions suitable for proving the soundness of our approach. It was observed that the underlying logical systems of DAML+OIL (OWL) and Z can be represented as institutions and further, by applying Goguen and Roşu's institution comorphisms [33], the soundness of the transformation from OWL (hence DAML+OIL) to Z can be proved.

Not all Semantic Web practitioners are experts in formal methods and they may find it difficult to interact with tools such as Z/EVES or Alloy Analyzer. An integrated tools environment is also conceived and designed to ease the application of the combined approach. The functionalities of this environment include systematic ontology creation, automatic ontology transformation, ontology querying, invocation of various reasoning tools, etc.

The above text highlights the issues related to the static aspect of the Web. However, the Web is more useful only if online services can be dynamically discovered and invoked to effect changes in the real world. The Semantic Web can also play a role by semantically marking up Web services to facilitate automatic service advertisement, discovery, invocation and composition. The OWL Services ontology (OWL-S) [95] is an OWL ontology that defines a core set of vocabularies to describe the Web services' capabilities, requirements, control constructs, etc. The dynamic nature of services makes the static reasoning techniques such as theorem proving insufficient. Live Sequence Charts (LSCs) [18] are a broad extension of the classic Message Sequence Charts (MSCs [53]). They rigorously capture communicating scenarios between system components. Play-Engine [38] is the tool support to visualize and simulate LSCs. In this thesis, we use LSC to represent OWL-S service process model ontologies and use Play-Engine to visualize and simulate them. This enables us to simulate and inspect the execution of services without actually implementing them.

1.2 Thesis Outline

This section gives an overview of the structure of the thesis.

1.2.1 Chapter 2 – Overview

Chapter 2 introduces background information on technologies, languages, tools and notations used in the presented work.

The Semantic Web languages take the central stage in this thesis. Hence, we first introduce the Semantic Web and the various ontology languages, such as RDF, RDF Schema, DAML+OIL, OWL, OWL$^-$ [56], SWRL, SWRL FOL [9] and WRL [1]. We present the syntax and semantics of the main language constructs, followed by a brief discussion on their tool support, including reasoners and visual editors.

Formal languages Z and Alloy are used extensively in the combined approach briefly introduced in the previous section. These languages together with their proof tools such as Z/EVES and Alloy Analyzer are also discussed and compared.

As a preparation for the discussion of the formal soundness proof of the transformation from ontology language OWL to Z using institutions, we present background information on category theory, institutions and institution morphisms.

Lastly, we introduce the OWL Services (OWL-S) ontology and the visual design language Live Sequence Charts (LSC). The visualization and simulation tool PlayEngine is also discussed to facilitate the presentation of the work later in Chapter 7 on simulating and checking Semantic Web services.

1.2.2 Chapter 3 – Checking DAML+OIL Ontologies using Z/EVES

Software engineering is a broad and well-developed research area over the past decades. We believe that mature software engineering languages and tools can contribute to the development of the Semantic Web vision. In this chapter, we demonstrate the ability of formal language Z in expressing Web ontologies and checking ontology-related properties. Specifically, we define the semantics of ontology language DAML+OIL in Z. By automatically transforming DAML+OIL and RDF ontologies into Z specifications, Core ontology reasoning services, namely concept subsumption, satisfiability and instantiation, checking can be performed in Z/EVES, a powerful theorem prover for Z.

It can be observed in this chapter that the proof process using Z/EVES is very interactive and requires substantial user expertise. This inspired us to propose a combined approach of checking Web ontologies to harness the synergy of Semantic Web and software engineering tools. This work is presented in the following chapter.

1.2.3 Chapter 4 – A Combined Approach to Checking Web Ontologies

As briefly discussed in Section 1.1, the trade-off between decidability and expressivity of ontology languages makes it awkward and difficult to represent certain complex properties in these languages. The newly proposed rules extension SWRL and SWRL FOL provide a partial remedy to this problem but they are still not as expressive as first-order predicate logic. Further, since they are undecidable languages, a reasoning engine to support full automation of all reasoning tasks would be an impossible task.

This shortcoming of DAML+OIL and SWRL led us to and propose to use Z to express complex properties inexpressible in DAML+OIL, OWL or SWRL. This makes it possible for Z proof tool such as Z/EVES to perform formal reasoning on these properties to ensure the correctness of ontologies.

Proof using Z/EVES is highly interactive and requires substantial expertise. The ontology languages were designed so that core reasoning tasks can be performed using Semantic Web reasoning tools fairly automatically. Hence, it is natural to combine Z/EVES and Semantic Web reasoning tools to harness their synergistic proof power. Moreover, the inclusion of Alloy Analyzer adds another useful dimension to the synergy since Alloy Analyzer is able to locate the source of errors in a specification.

In the rest of Chapter4, we present a combined approach to checking DAML+OIL and RDF ontologies by using proof tools RACER, Z/EVES and Alloy Analyzer together. We begin by defining Z and Alloy semantics for DAML+OIL. The Z and Alloy semantics enables Z/EVES and Alloy Analyzer to understand DAML+OIL and RDF ontologies. With this semantics as a basis, we then develop a transformation program to automatically transform an ontology to Z and Alloy specifications, respectively.

The complementary proof power can be exploited through applying these reasoning tools in turn and expressing complex properties in Z and use Z/EVES to prove these properties. Firstly, ontological consistency can be checked by SW reasoning engines such as RACER and FaCT++ with full automation. Secondly, any such inconsistency found can be precisely located by Alloy Analyzer. Thirdly, more complex properties inexpressible in DAML+OIL and OWL can be expressed in Z and checked by Z/EVES. The strength of the combined approach is demonstrated through a real-world military planning case study. It is observed that Alloy Analyzer located the source of ontological inconsistencies found by RACER; and a number of errors undiscovered by RACER were found by Z/EVES.

1.2.4 Chapter 5 – Z Semantics for OWL: Soundness Proof Using Institution Morphisms

Chapter 4 presents on the practical aspects of the combined approach, namely, the transformation from DAML+OIL to Z and Alloy and the actual reasoning approach using the combination of tools. A fundamental issue, the soundness of the Z and Alloy semantics of DAML+OIL, is not addressed there.

Replacing DAML+OIL, the Web Ontology Language (OWL) became the W3C recommendation in February 2004[1]. As OWL is the successor of DAML+OIL, they are very similar in many aspects. Since OWL is also a W3C recommendation as the ontology language designed to replace DAML+OIL, it is natural to shift focus to the support of OWL.

Based on our work in [24], we have developed a Z semantics for OWL. In chapter 5, we attempt to formally prove the soundness of the Z semantics for OWL by using institutions [31] and institution morphisms [33].

Introduced by Goguen and Burstall [31], institutions are used to formalize the notion of "logical systems". They provide a means of reasoning about software specifications regardless of the underlying logical systems.

The basic components of a logical system, an institution, are *models* and *sentences*, related by the *satisfaction relation*. The compatibility between models and sentences is provided by *signatures*, which formalize the notion of vocabulary from which the sentences are constructed. By modeling the signatures of a logical system as a category, we get the possibility to translate sentences and models across signature morphisms. The consistency between the satisfaction relation and the translation is given by the

[1] This is about the time when the work on combined approach [23] was in progress.

satisfaction condition, which intuitively means that *the truth is invariant under the change of notation*.

Institutions are suitable for relating Z and OWL DL (and DAML+OIL) as the logical systems (semantics) of these languages can be represented as institutions. In Chapter 5, we also present the institutions of Z and OWL and by applying Goguen and Roşu's institution comorphisms [33], the soundness of the Z semantics for OWL (and DAML+OIL) can be proved.

1.2.5 Chapter 6 – SESeW - An Integrated Tools Environment for the Semantic Web

Formal methods usually make extensive use of mathematical concepts and symbols, which often prove to be difficult for users without the relevant mathematical background. In order to hide as much underlying formal methods notations as possible and make the combined approach more friendly to users who are not familiar with the various reasoning tools, an easy-to-use visual tool that supports automated creation, transformation and querying of ontologies is much desired and valuable.

In Chapter 6, we present such an integrated tools environment, the SESeW (**S**oftware **E**ngineering for **Se**mantic **W**eb), that serves as a graphical front-end to the various reasoning tools used in the combined approach under one umbrella. Using SESeW, tasks such as ontology transformation, validation, querying, etc. can be visually performed. To make SESeW more more versatile, we also implemented a systematic approach to ontology creation, the Methontology [29]. With these functionalities, SESeW is a prototype of an ontology creation, transformation, validation and querying tool based on sound software engineering methods.

1.2.6 Chapter 7 – Simulating Semantic Web Services with LSC and Play-Engine

The full potential of the Semantic Web can only be realized when dynamic resources such as the Web Services are incorporated. The Semantic Web services ontology OWL-S is an OWL ontology that defines an essential set of vocabularies for describing the capabilities, requirements, effects, output, etc., of a Web service. It is meant to be used together with the Web Services standards such as WSDL [14] and SOAP [110] to enable software agents to automatically advertise, discover and negotiate Web services.

The correctness of Semantic Web services is essential to the functioning of software agents crawling the Semantic Web. We believe that erroneous service descriptions will give rise to invocation of wrong services, with wrong parameters or resulting in undesired outcome.

In Chapter 7, we propose to apply software engineering methods and tools to visualize, simulate and verify OWL-S process models. Live Sequence Charts (LSCs) [18] are a broad extension of the classic Message Sequence Charts (MSCs [53]). They capture communicating scenarios between system components rigorously. LSCs are used to model services, capturing the inner workings of services, and its tool support Play-Engine [38] is used to perform automated visualization, simulation and checking.

1.2.7 Chapter 8 – Conclusion

Chapter 8 concludes the thesis, summarizes the main contributions and discusses future work directions.

1.3 Publications

Most of the work presented in this thesis has been published/accepted in international conferences proceedings.

The work on the Z semantics (Chapter 3) of DAML+OIL and checking DAML+OIL ontologies using Z/EVES has been published in *The Twenty-sixth International Conference on Software Engineering* (ICSE'04, May 2004, Edinburgh, acceptance rate 13%) [24].

The combined approach for checking Web ontologies (Chapter 4) has been published in *The Thirteenth International World Wide Web Conference* (WWW'04, May 2004, New York, acceptance rate 14.6%) [23].

Work on soundness proof of transformation from OWL to Z using institutions [63] in Chapter 5 has been published in *The Seventeenth International Conference on Software Engineering and Knowledge Engineering* (SEKE'05, July 2005, Taipei) [64].

The work on the integrated tools environment was presented at *The Twelfth Asia-Pacific Software Engineering Conference* (APSEC'05, December 2005, Taipei) [22].

The work on simulating and visualizing Semantic Web services using LSC and Play-Engine was published in *Seventh International Conference on Formal Engineering Methods* (ICFEM'05, November 2005, Manchester) [90].

I have also contributed to other published works [25, 26, 21, 91, 104, 103, 105, 61, 65], which are mostly as pre-thesis/follow-up works.