Chapter VI
Reasoning on the Semantic Web
Rogier Brussee
Telematica Instituut, The Netherlands
Stanislav Pokraev
Telematica Instituut, The Netherlands

ABSTRACT

We describe reasoning as the process needed for using logic. Efficiently performing this process is a prerequisite for using logic to present information in a declarative way and to construct models of reality. In particular we describe description logic and the owl ontology language and explain that in this case reasoning amounts to graph completion operations that can be performed by a computer program. We give an extended example, modeling a building with wireless routers and explain how such a model can help in determining the location of resources. We emphasize how different assumptions on the way routers and buildings work are formalized and made explicit in our logical modeling, and explain the sharp distinction between knowing some facts and knowing all facts (open vs. closed world assumption). This should be helpful when using ontologies in applications needing incomplete real world knowledge.

WHAT DO WE MEAN BY REASONING AND WHAT IS IT GOOD FOR?

Any reader of this text is equipped with the most capable reasoner found on this planet: the human brain. Thus, it is not surprising that we have come to take reasoning for granted. That sound reasoning follows a restricted set of formal rules is a relatively recent invention. Long after people learned to grow food, rule empires and measure land, the Greek philosophers formalized the rules of logic and set standards for mathematical proof. They also realized the importance of sound reasoning for rhetoric and decision making. Formal axiomatic logic is much younger, dating to the late nineteenth and twentieth century, with applications to computer science going back to the 1930s, predating computers themselves. Historically speaking, we need to define what we actually mean by reasoning.
In this chapter we will take the viewpoint that reasoning is a tool for formulating concise models of “real world” phenomena such as people, mobile phones, transactions, services or databases. Roughly speaking, we want to model the world in terms of dots and arrow (subject predicate object) diagrams, that model how “things” relate to each other, and define classes to group “things.” In fact, interpreting dots and arrows in a sufficiently general way using different dots for “things” and for “classes” the latter can be considered a special case of the former. We want to impose logic rules that force the existence (or non existence) of implied arrows. In practice, this means that we can specify model concepts in terms of fewer primitive properties, that our model can be more accurate and that we need not specify all the properties that logically follow from possessing primitive properties. It also means that we can query a model in terms of the implied property arrows in addition to the stated ones. On the downside, it means that we need a computer program to compute these implied arrows, at least at query time, and that we want to control the time it takes the computer to do this.

This viewpoint naturally fits with an orientation towards ontologies as a consensual vocabulary for “real world” phenomena. In its most restricted form, a vocabulary is merely a collection of words, and a description of the real world amounts to naming: associating the word with something. Such a vocabulary has just points and no arrows. In a more liberal interpretation of vocabulary we not only have symbols to name real world phenomena, but we also have names for the relations that exist between them such as groupings, and attribute values. There are many computer applications where we define attributes and values. Such a model has both points and arrows, but each arrow must be explicitly mentioned. Finally the real world relations may naturally satisfy constraints. These relations are often used in programs and, if you are lucky, they are documented. For example, we have abstract constraints such as “the member of a group is a member of any larger group” or concrete ones such as “a database record has exactly one primary key.” Such a model is often specified with only a generating set of arrows, for example, by only explicitly stating subclass relations. Making use of such implied constraints, and making them an explicit part of information model has the advantage that models can often be simpler and is a primary motivation for using semantic techniques. Reasoning is computing the implied relations and constructing the missing arrows.

LOGICAL IMPLICATIONS IN DESCRIPTION LOGIC

The reasoning we do here deals with organizing Classes, Individuals and Properties. The associated logic is called description logic (DL) (Calvanece, McGuinness, Nardi & Patel-Schneider, 2003) which has become popular as the basis for the Web Ontology Language (OWL) (Dean, Schreiber, Bechhofer, van Harmelen, Hendler, Horrocks, et al., 2004) which for our purposes can be considered as an extension of RDF.1 Difficult papers have been written about the subject but since we want to draw practical conclusions we give as mundane a description as possible. A similarly practical approach is present in the Protégé OWL-manual (Horridge, Knublauch, Rector, Stevens, & Wroe, 2004). A list of further reading material was collected by Franconi (2002).2

Individuals and Classes

Individuals represent entities or phenomena in a domain of interest, for example, John, TopTech-Company or London (Figure 1). Logically an individual is an assertion of existence for example, by naming JohnSmith we assert the existence of John. Two individuals can be equal to each other but known under a different name for example, John and JSmith. Existence can already be inter-
Related Content

Addressing Device-Based Adaptation of Services: A Model Driven Web Service Oriented Development Approach
www.igi-global.com/chapter/addressing-device-based-adaptation-services/69479?camid=4v1a

Service Discovery and Composition Based on Contracts and Choreographic Descriptions
Mario Bravetti and Gianluigi Zavattaro (2013). Adaptive Web Services for Modular and Reusable Software Development: Tactics and Solutions  (pp. 60-88).
www.igi-global.com/chapter/service-discovery-composition-based-contracts/69470?camid=4v1a

From SOA to Pervasive Service Ecosystems: An Approach Based on Semantic Web Technologies
www.igi-global.com/chapter/soa-pervasive-service-ecosystems/69475?camid=4v1a

A Generation Method of Network Security Hardening Strategy Based on Attack Graphs
www.igi-global.com/article/a-generation-method-of-network-security-hardening-strategy-based-on-attack-graphs/125458?camid=4v1a