
*University of Coimbra
Faculty of Sciences and Technology
Department of Informatics Engineering*



Probabilistic Reasoning in the Semantic Web using Markov Logic

MSc Thesis Proposal

Pedro Carvalho de Oliveira

Candidate

Paulo Jorge de Sousa Gomes

Supervisor

January 2009

*Knowledge and Intelligent Systems Laboratory
Cognitive and Media Systems Group
Centre for Informatics and Systems of the University of
Coimbra*



Contents

1.	Introduction.....	1
2.	State of the Art	3
2.1.	Logic	3
2.1.1.	Logic as a Knowledge Representation Formalism	3
2.1.2.	Propositional Logic	4
2.1.3.	First-order Logic.....	5
2.1.4.	Description Logics.....	6
2.2.	Probabilistic Reasoning	9
2.2.1.	Probability Theory	9
2.2.2.	Probabilistic Graphical Models.....	11
2.2.3.	Inference in Probabilistic Graphical Models.....	14
2.3.	Statistic Relational Learning.....	17
2.4.	Semantic Web	20
2.4.1.	From the Syntactic Web to the Semantic Web	20
2.4.2.	Semantic Web Architecture	22
2.4.3.	URI and Unicode.....	23
2.4.4.	XML.....	23
2.4.5.	RDF	25
2.4.6.	Ontology Vocabulary	27
2.4.7.	Logic.....	29
2.4.8.	Digital Signatures, Proof and Trust.....	30
2.5.	Markov Logic	31
2.5.1.	Markov Logic Networks.....	31
2.5.2.	Inference	33
2.5.3.	Learning	33
2.6.	Related Work.....	35
2.6.1.	Deterministic Reasoning in the Semantic Web	35
2.6.2.	Uncertainty in the Semantic Web	37
2.6.3.	Vagueness in the Semantic Web	40
2.6.4.	Conclusions.....	41
3.	Technologies and Tools	43
3.1.	Programming Platform.....	43
3.2.	Supporting Tools	43
3.2.1.	Semantic Web Tools.....	43
3.2.2.	Markov Logic Tools.....	45

4. Proposed Approach.....	48
4.1. Solution strategy	48
4.1.1. Probabilistic Reasoning in Uncertainty-annotated Ontologies	49
4.1.2. Probabilistic Reasoning with Weight Learning	50
4.2. System	51
4.2.1. Features and Functionalities	51
4.2.2. Requirements	51
4.2.3. Architecture.....	52
4.3. Evaluation.....	54
5. Conclusions.....	55
6. References	57

Figures

Figure 1. Simple Bayesian network with 4 binary random variables and their conditional probability tables.	11
Figure 2. Simple Markov network with 4 binary random variables and their potential functions. Clique 1 is composed by nodes {Sunny, Walk}, Clique 2 by {Hot, Beach}, and Clique 3 by {Beach, Sunny}.	12
Figure 3. Simple Markov network with 4 binary random variables and one example feature for the clique {Sunny, Walk}.	13
Figure 4. Statistical relational learning taxonomy.	18
Figure 5. Visual representation of the previous HTML.	20
Figure 6. Semantic Web layered architecture (adapted from (Tim Berners-Lee 2000)).	22
Figure 7. XML document tree of the previous XML example.	24
Figure 8. Visual representation of a RDF statement.	26
Figure 9. Visual representation of three RDF statements.	26
Figure 10. OWL sub layer.	29
Figure 11. Markov network, with an example feature, built from the previous Markov logic network.	32
Figure 12. System architecture.	52
Figure 13. Flow chart representing the behavior of the System Core.	53
Figure 14. Project planning.	56

Tables

Table 1. Logical connectives.....	4
Table 2. Logical truth table.....	4
Table 3. First-order logic quantifiers.....	5
Table 4. Basic Description Logics concept constructors.....	7
Table 5. Description Logics labels.....	8
Table 6. Full joint probability distribution table of binary random variables Sunny and Hot.....	10
Table 7. Comparison between Bayesian networks and Markov networks.....	14
Table 8. Markov logic network example.....	31
Table 9. Alchemy features and algorithms.....	45
Table 10. PyMLNs features and algorithms.....	46
Table 11. Markov thebeast features and algorithms.....	46
Table 12. Preliminary study on ontology individuals.....	51

1. Introduction

The Semantic Web (T. Berners-Lee, J. Hendler, and Lassila 2001) envisions a world where agents share and transfer structured knowledge in an open and semi-automatic way. This knowledge, in most of the cases, is characterized by uncertainty (Lindley 2006), i.e., there is a lack of certainty in assigning a true/false value to a certain knowledge statement. However, the Semantic Web doesn't provide any means of dealing with knowledge uncertainty. Its languages, like RDF and OWL, are mainly based on crisp logic, being incapacitated of dealing with partial and incomplete knowledge. Reasoning in the Semantic Web resigns to a deterministic process of verifying if statements are true or false.

One field that has been trying to tackle the problem of knowledge uncertainty is the field of probabilistic reasoning (Pearl 1988). This field studies methodologies to represent and reason about uncertain knowledge through the use of probability theory. Some of the developed models, like Bayesian and Markov networks (Roller et al. 2007), are considered the state of the art on dealing with uncertainty. In fact, based on the success of probabilistic reasoning and other similar areas, some efforts have been made on representing and reasoning with uncertainty in the Semantic Web (Thomas Lukasiewicz and Umberto Straccia 2008). These works mainly focused on extending the logics behind Semantic Web languages with probabilistic/possibilistic/fuzzy concepts, or on combining these languages with probabilistic formalisms like Bayesian networks. However, most of these approaches have some problems of applicability in real domains, mainly because its complexity and domain restrictiveness.

Recently, a new area of research, called statistical relational learning (SRL) (Lise Getoor and Ben Taskar 2007), has arisen. SRL tries to expand probabilistic reasoning to complex relational domains, like the Semantic Web. This is achieved by combining representation formalisms, like logic and frame-based systems, with probabilistic models. Some of its more expressive and complete approaches, like Bayesian logic (Milch et al. 2007) and Markov logic (Pedro Domingos, Stanley Kok, et al. 2008), have proven to provide interesting capabilities on learning and reasoning about uncertainty in many real world domains.

The objective of this thesis is to study mechanisms to perform probabilistic reasoning in the Semantic Web. For this purpose, we will use Markov logic, a novel representation formalism that combines first-order logic with probabilistic graphical models.

In Markov logic, unlike first-order logic, worlds that violate formulas are not impossible, but only less probable. This is achieved by attaching weights to first-order formulas: higher the weight, bigger the difference between a world that satisfies the formula and one that does not, other things being equal. These weighted formulas represent a Markov logic network, which can be seen as a template to construct Markov networks from given sets of constants: each ground atom is a variable, logical connectives are the edges between variables, and each

grounded formula is a feature. The resulting Markov network gives a probability distribution over the possible worlds, being used to answer any probabilistic query about the domain.

Therefore, to apply Markov logic in the Semantic Web, two objects are needed: first-order formulas and their weights. *Formulas* can be acquired by interpreting the semantics of Semantic Web languages as sets of first-order formulas. Since most of these languages represent ontologies based on Description Logics (Baader et al. 2007), they follow a model-theoretic semantics, having a direct correspondence with formulas in first-order logic. *Weights* can be acquired by interpreting ontological uncertainty-annotations values, or by learning them through example data (e.g., ontology individuals and textual corpus about the domain).

To demonstrate the feasibility of our approach, we will develop a system that provides a Semantic Web interface to Markov logic reasoning and learning capabilities. This system can be accessed visually and programmatically, allowing its use in many interesting Semantic Web tasks, like ontology mapping (Euzenat and Shvaiko 2007) and learning (Maedche 2002).

The organization of this thesis proposal is as follows:

- *Chapter 2* reviews the main concepts that will be used in this thesis. A description of the related work is also provided;
- *Chapter 3* introduces some of the tools and technologies that will be used in this thesis;
- *Chapter 4* is dedicated to an in-depth description of our proposed approach;
- *Chapter 5* gives general conclusions about this thesis proposal;
- *Chapter 6* provides a complete list of the bibliography used in this work.

2. State of the Art

In this chapter, we present the main concepts used in this thesis. First, a review on logic and probabilistic reasoning is provided. Next, we introduce the field of statistical relational learning, followed by an explanation about the main concepts behind the Semantic Web. Finally, an introduction to Markov logic is provided, followed by a description of the most relevant related work to this thesis.

2.1. Logic

Over the centuries, logic has been studied has the underlying mechanism of valid demonstration and inference. The roots of many disciplines, like philosophy, mathematics, and computation, are based on the principles of logic. In this work, logic is used as a *knowledge representation formalism*. For this purpose, logic provides a meaningful and unambiguous way of representing knowledge. The next sections describe three types of logic that are of most importance to this work: propositional logic, first-order logic, and description logics. But before, a slight introduction to the general logic concepts is provided.

2.1.1. Logic as a Knowledge Representation Formalism

The main component of logic as a knowledge representation formalism is the *knowledge base* (KB) (Stuart Russell and Norvig 2002). Each logical KB is composed by a set of *formulas*, or *sentences*, expressed in a logical language. These formulas are expressed according to the syntax of the logical language, i.e., the specification if the formulas are well formed. Their semantics define the truth of each sentence with respect to each *possible world*. For example, the sentence $FatherOf(x,y)$ is true in a world where x is the father of y . These possible worlds are usually called *models*.

Inference, i.e., derive new facts and relations from existing knowledge, is the main objective of a logical KB. Inference algorithms use the *entailment* between sentences, i.e., if a sentence follows logically from another sentence, in this task. Entailment is usually represented by $\alpha \models \beta$, meaning that in every model where α is true, β is also true. For example, $FatherOf(x,y) \models SonOf(y,x)$ because in every model where x is father of y , y is also the son of x . Inference can be defined as the process of finding a specific entailment in a KB. This is usually represented by $KB \vdash_i \alpha$, meaning that the entailment α is derived from KB using inference algorithm i .

There are two important properties of inference algorithms: soundness and completeness. An algorithm is *sound* if it only derives entailed sentences (i.e., it only derives sentences that are valid with respect to their semantics). An algorithm is *complete* if it can derive any sentence that is entailed (i.e., it can derive all the true sentences).

Since logical KBs provide a representation of knowledge, the correspondence between the representation and the real knowledge must be defined. This procedure is called *grounding*, and is usually made by establishing connections between the symbols and their meanings.

Next, a simple logic to represent logical KBs, propositional logic, is defined.

2.1.2. Propositional Logic

A *propositional logic KB* (Stuart Russell and Norvig 2002) is a set of formulas in propositional logic. Formulas are composed by *symbols*, representing *propositions* that can be true or false. Symbols are usually represented in uppercase names (e.g., *A* and *BOB* are symbols). There are two symbols with fixed meaning: *TRUE* is the always-true proposition, and *FALSE* is the always-false proposition. These kind of simple formulas, with only one symbol, are called *atomic formulas* (or *atoms*). More complex formulas can be built from atoms by using *logical connectives*.

Connective	Meaning
\neg	Not
\wedge	And
\vee	Or
\Rightarrow	Implication
\Leftrightarrow	Biconditional

Table 1. Logical connectives.

Logical connectives have an order of precedence ($\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$, with decreasing precedence), but delimiters can be also used to ensure precedence (e.g., $A \vee (B \Rightarrow C)$ instead of $A \vee B \Rightarrow C$).

The models of propositional logic are defined by simply assigning true or false values to the proposition symbols (i.e., for N propositional symbols, there are 2^N models). The truth value of a formula in a model can be easily computed:

- Atomic formulas already have their trueness computed (they are either true or false);
- Complex formulas are computed recursively using *truth tables*.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

Table 2. Logical truth table.

Inference in propositional logic is both sound and complete, and several algorithms, like *model-checking* and *resolution* (Stuart Russell and Norvig 2002), were studied.

Propositional logic is a simple and powerful language to represent knowledge. However, it lacks of expressive power to describe concisely very large domains. For example, it is difficult to state the fact that all the persons have a name, since we have to write a formula with that information for each person, increasing the number of models exponentially by the number of persons. For this reason, more expressive languages have been developed. First-order logic is one of them.

2.1.3. First-order Logic

First-order logic (FOL) (Barwise and Etchemendy 2002) (Stuart Russell and Norvig 2002) builds a more expressive language in the foundations of propositional logic. This new language is designed to create accurate real world models, characterized by a large number of objects, with relations between them.

A *first-order logic KB* is a set of formulas in first-order logic. FOL uses four types of symbols:

- *Constants*, representing objects (i.e., individuals) in the domain (e.g., *Anna*, *Bob*);
- *Variables*, ranging over constants (e.g., x , y);
- *Functions*, mapping objects to objects (e.g., $motherOf(Bob)=Anna$);
- *Predicates*, representing binary relations between objects (e.g., $Friends(Anna, Bob)$).

Variables can be *typed*, meaning that they can only model a restricted range of objects (e.g., variable x only ranges about *People*).

A *term* is an expression representing an object. It can be a constant, a variable or a function of finite arity (e.g., x , *Pedro* and $Friends(x,y)$ are terms). A term is grounded when it contains no variables (e.g., $Friends(Anna, Bob)$ is a grounding of $Friends(x,y)$).

The simplest FOL formula is an *atom*, composed by a predicate with a finite number of terms as parameters (e.g., $Friends(x,Pedro)$ and $Friends(Pedro,motherOf(Bob))$ are atoms). Just like propositional logic, more complex formulas can be formed by joining atoms with logical connectives ($\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$) and delimiters. FOL also supports *quantifiers*.

Quantifier	Meaning
\forall	Universal Quantification
\exists	Existential Quantification

Table 3. First-order logic quantifiers.

A formula is grounded when all of its terms are also grounded. Some examples of formulas are provided:

-
1. $\forall x \forall y \forall z \text{ Friends}(x, y) \wedge \text{Friends}(y, z) \Rightarrow \text{Friends}(x, z)$
 2. $\forall x \text{ Friends}(x, \text{MotherOf}(x))$
 3. $\text{Neighbors}(\text{Portugal}, \text{Spain})$
-

Formula 1 states that friends of friends are also friends. In the second formula, everyone is friend of his mother, and formula 3 states that two countries, *Portugal* and *Spain*, are neighbors.

Models in FOL are tuples (D, I) , where D is a set of domain elements, and I is an interpretation. The *domain elements* are the objects contained in that model (e.g., the domain of $\forall x, y \text{ Friends}(x, y)$ is all the objects that can take the values x and y). The *interpretation* is an assignment that maps objects, functions and relations into symbols (e.g., one possible interpretation of $\text{motherOf}(\text{Pedro})$ is that motherOf refers to the motherhood relation, and Pedro refers to the author of this work). Since symbols can have various interpretations, there are distinct models relating to those interpretations. The truth value of a formula can be determined given a model and an interpretation of the values of the variables (i.e., an assignment of objects from the domain to the variables).

Every first-order KB can be translated to *clausal form*, also known as *conjunctive normal form* (Barwise and Etchemendy 2002). A KB in clausal form is a conjunction of clauses, each clause being a disjunction of literals. For example, Formula 1 can be translated as $\neg \text{Friends}(x, y) \vee \neg \text{Friends}(y, z) \vee \text{Friends}(x, z)$. This translation is useful in automated inference.

Propositionalization and *unification* techniques (Stuart Russell and Norvig 2002) can be used in FOL inference, providing both sound and complete inference. However, entailment is only *semidecidable* (i.e., the algorithm return *yes* when the right answer is yes, but it fails to say *no* when the right answer is no). Since this characteristic arise problems in many domains, other logical languages, like Description Logics, have been developed.

2.1.4. Description Logics

Description Logics (DLs) (Baader et al. 2007) are a family of logical languages based on *Semantic Networks* and *Frame Systems*. They are specially designed to model terminological domains. Historically, they were designed as a subset of first-order logic, providing decidable reasoning, while maintaining some of the most important expressivity characteristics of it. DLs use two types of symbols:

- *Atomic concepts*, representing sets of individuals;
- *Atomic roles*, expressing relationships between individuals.

Description Logics KBs are composed by *descriptions*. Elementary descriptions are composed by atomic concepts and atomic roles. More complex descriptions can be composed by using *concept constructors*. The following basic concept constructors are usually used:

Constructor	Meaning
\top	Universal Concept
\perp	Bottom concept
\neg	Atomic Negation
\sqcap	Logical Intersection
\sqcup	Logical Union
\equiv	Logical Equivalence

Table 4. Basic Description Logics concept constructors.

All of these constructors have a similar meaning to those presented in the previous logics. However, more complex constructors can also be used:

- $\forall R.C$, called *value restrictions*, describing individuals which are always connected by role R to the concept C ;
- $\exists R.C$, called *existential quantification*, describing all the individuals connected by role R with concept C ;
- $C \sqsubseteq D$, called *subsumption*, describing that the concept D is more general than concept C .
- $\leq nR$, called *number restrictions*, restricting the cardinality of the role R to the number n (\leq can be substituted by \geq or $=$).

DLs divide KBs in two distinct parts: the *intensional* knowledge in the form of a terminology, called *Terminological Box* (TBox), and the *extensional* knowledge, called *Assertional Box* (ABox). The TBox provides the vocabulary, in terms of concepts and rules, of the KB. This is usually done by defining concepts using the logical equivalence constructor (e.g., $Woman \equiv Person \sqcap Female$). The ABox uses the TBox vocabulary to make assertions about individuals. There are two kinds of assertions, corresponding for the two symbols of the language: *concept assertions* (e.g., $Person(PEDRO)$) and *role assertions* (e.g., $Friends(BOB, ANNA)$).

DLs provide a *model-theoretic semantics*. This means that descriptions can be, in most of the cases, identified with formulas in first-order logic. The main idea behind this identification is that concepts correspond to unary predicates, roles to binary predicates, and individuals correspond to constants (e.g., $Woman \equiv Person \sqcap Female$ is interpreted as $\forall x Woman(x) \Leftrightarrow Person(x) \wedge Female(x)$).

As previously referred, Description Logics is a family of languages. These languages are characterized by the use of different operators. Usually, all the languages are based in the *Attributive Language* (\mathcal{AL}). This language supports atomic negation, concept intersection, universal restrictions, and limited existential

quantification. To distinguish different languages, labels were created to determine the expressivity of these languages.

Label	Meaning
\mathcal{EL}	Intersection and Full Existential Restrictions
\mathcal{F}	Functional Properties
\mathcal{E}	Full Existential Quantification
\mathcal{U}	Concept Union
\mathcal{C}	Complex Negation
\mathcal{S}	ALC with transitive Roles
\mathcal{H}	Role Hierarchy
\mathcal{R}	Ir/reflexivity and role disjointness
\mathcal{O}	Nominals
\mathcal{I}	Inverse Properties
\mathcal{N}	Cardinality Restrictions
\mathcal{Q}	Qualified Cardinality Restrictions
(\mathcal{D})	Use of Datatype properties

Table 5. Description Logics labels.

For example, the language \mathcal{SHOIN} is \mathcal{ALC} with Transitive Roles (\mathcal{S}), plus Role Hierarchy (\mathcal{H}), Nominals (\mathcal{O}), Inverse Properties (\mathcal{I}), and Cardinality Restrictions (\mathcal{N}).

The expressivity of the language determines the complexity of its inference. While some DLs are a decidable subset of first-order logic, others can go beyond first-order and have problems of decidability, soundness and completeness. Inference in DLs is usually made by using specific *tableau-based* algorithms (Ralf Moller and Volker Haarslev 2008).

2.2. Probabilistic Reasoning

Many real world domains are characterized by uncertainty (Lindley 2006). There is only a limited knowledge about the domain, being very difficult to exactly describe the domain or predicting its future behavior. When modeling a domain, uncertainty can be the result of distinct reasons (Stuart Russell and Norvig 2002):

- *Laziness*, when it is too difficult to correctly model the domain;
- *Theoretical ignorance*, when there is not enough information to correctly model the domain;
- *Practical ignorance*, when all the knowledge is available, but the user does not know how to model it correctly.

One way of dealing with uncertainty is through probability theory, where a degree of belief (i.e., a probability) is assigned to the knowledge in the domain. Probabilistic reasoning is an area that tries to find efficient mechanisms to reason under uncertain knowledge expressed through probability theory. In this area, probabilistic graphical models provide a compact and expressive tool to deal with uncertainty and complexity, by joining concepts from probability theory and graph theory in the same representation. There are two main kinds of probabilistic graphical models: directed (Bayesian networks) and undirected (Markov networks). In this section, the concepts behind these two models are described. First, a brief introduction on probability theory is given.

2.2.1. Probability Theory

The main concept behind *probability theory* (Stuart Russell and Norvig 2002) is the *probability*: a value between 0 and 1 representing the degree of belief in a statement (assigning a probability near 1 to a statement represents a strong belief in that statement, while near 0 represents discredit in the statement).

The basic unit of probability theory is the *random variable*, which represents a variable with initial value unknown. This variable has a domain, representing the values that it can take. In this work, we focus on variable with discrete domains (e.g., the random variable *Weather* can take the values $\{sunny, rainy, cloudy, snow\}$). Random variables can be joined in *propositions*, using the same logical connectives as logical languages (e.g., $Weather = rainy \wedge Temperature = low$). Probabilities are given to propositions, declaring the degree of belief in those propositions. There are two main kinds of probabilities:

- *Prior probability* $P(a)$, which is the degree of belief in preposition a in the absence of any other information (e.g., $P(Weather=rainy) = 0.2$);
- *Conditional (or posterior) probability* $P(a|b)$, which is the degree of belief in proposition a given preposition b (e.g., $P(Weather=rainy|Temperature=low) = 0.4$).

Conditional probabilities can be defined in terms of prior probabilities as

$$P(a|b) = \frac{P(a \wedge b)}{P(b)}, \quad 2.1$$

$$P(a \wedge b) = P(a, b) = P(a|b)P(b). \quad 2.2$$

Each discrete random variable has a *prior probability distribution* associated. This distribution gives the values for the probabilities of each individual state of the variable (e.g., $P(\text{Weather}) = [0.6, 0.2, 0.15, 0.5]$ means that $P(\text{Weather}=\text{sunny})=0.6$, $P(\text{Weather}=\text{rainy})=0.2$, and so on). *Joint probability distributions* between random variables are defined by a combination of states of the variables (e.g., $P(\text{Weather}, \text{Temperature})$ has a $N \times M$ probability table, being N the size of the domain of *Weather*, and M the size of the domain of *Temperature*). A joint probability distribution with all the variables in the world is called a *full joint probability distribution*. This type of probability distribution is a complete specification of the uncertainty of the world, and can be therefore used to answer any probabilistic query about the world. For example, the full joint probability distribution table of two binary random variables (*Sunny* and *Hot*) can be defined as:

	Hot	-Hot
Sunny	0.55	0.1
-Sunny	0.05	0.3

Table 6. Full joint probability distribution table of binary random variables Sunny and Hot.

Given that distribution, some probabilistic queries can be made:

$$P(\text{Sunny} \wedge \text{Hot}) = 0.55$$

$$P(\text{Hot}) = 0.55 + 0.05 = 0.6$$

$$P(\text{Sunny}|\text{Hot}) = \frac{0.55}{0.6} \approx 0.92$$

Conditional probabilities, like Formula 2.1, can also be defined by the *Bayes' theorem*

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}. \quad 2.3$$

Since the size of the full joint distribution tables grows with the number of random variables (and the size of their domains), this type of probabilistic reasoning can be intractable in many cases (e.g., N binary random variables correspond to a table with 2^N elements). For this purposes, more advanced representations were designed to overcome this problem. The field of probabilistic graphical models studies how some interesting properties of graph theory can be used in probabilistic reasoning.

2.2.2. Probabilistic Graphical Models

Probabilistic graphical models (Roller et al. 2007) (Jordan 2004) combine probability theory and graph theory in the same representation, allowing to compactly represent complex domains. The main goal is to efficiently represent a joint probability distribution over a set of random variables. For this purpose, they explore the structure of the distribution, by using the independence properties between random variables, to generate a compact and modular representation of the distribution. The independence property that they explore is the *conditional independence*, i.e., if two random variables are independent in their conditional probability distribution given a third random variable. A random variable X is conditionally independent of Y given Z if

$$P(X, Y | Z) = P(X | Z)P(Y | Z). \quad 2.4$$

There are two main classes of probabilistic graphical models that explore these properties: directed (Bayesian networks) and undirected (Markov networks).

Bayesian Networks

Bayesian networks (Stuart Russell and Norvig 2002) represent a joint distribution of random variables as a directed acyclic graph. Its nodes are the random variables, while the edges correspond to direct influence from one node to another. Each random variable has an associated *conditional probability distribution*, normally represented as a table (*conditional probability table*, CPT). These CPTs capture the conditional probability of the random variable given its parents in the graph.

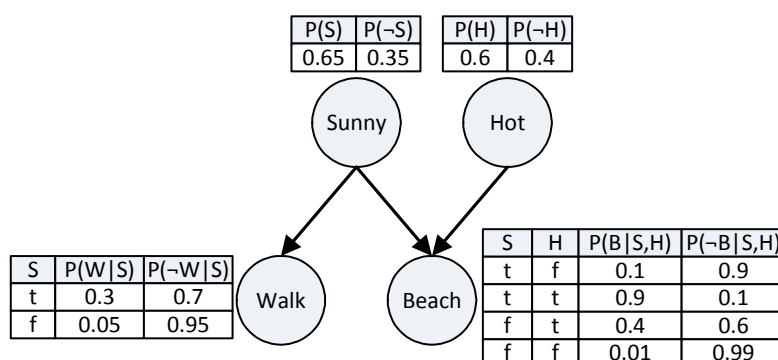


Figure 1. Simple Bayesian network with 4 binary random variables and their conditional probability tables.

This graphical representation provides a complete description of the joint probability distribution of its random variables. This way, the joint probability distribution of a set of random variables $X = \{X_1, \dots, X_n\}$ can be written as

$$P(X = x) = \prod_{i=1}^n P(x_i | \text{parents}(X_i)), \quad 2.5$$

where $parents(X_i)$ returns the specific values of the parents of the random variable X_i . For example, given the Bayesian network of Figure 1, the probability of being sunny and hot, and we go to the beach but do not take a walk is $P(S = true, H = true, W = false, B = true) = P(S)P(H)P(\neg W|S)P(B|S, H) = 0.65 * 0.6 * 0.7 * 0.9 \approx 0.25$.

Bayesian networks graph representation also encodes important results to study the interdependence between random variables. The most important is that a node is conditionally independent of its non-descendants given its parents (e.g., *Walk* is conditionally independent of $\{Beach, Hot\}$ given *Sunny*). Other independence assertions can be found by using a complex procedure called *d-separation* (Bishop 2007). This procedure uses the flow of the nodes edges to decide if a set of nodes X is independent of another set Y given a third set Z .

Markov Networks

Markov networks (also called *Markov random fields*) (Roller et al. 2007) (Pedro Domingos, Stanley Kok, et al. 2008) represent a joint probability distribution of random variables as an undirected graph, where the nodes represent the variables, and the edges correspond to some notion of direct probabilistic interaction between neighboring variables. This interaction is parameterized by *potential functions*. There is a potential function for each *clique* (i.e., a completely connected sub-graph) in the graph, being this potential function a non-negative real-valued function of the state of the corresponding clique.

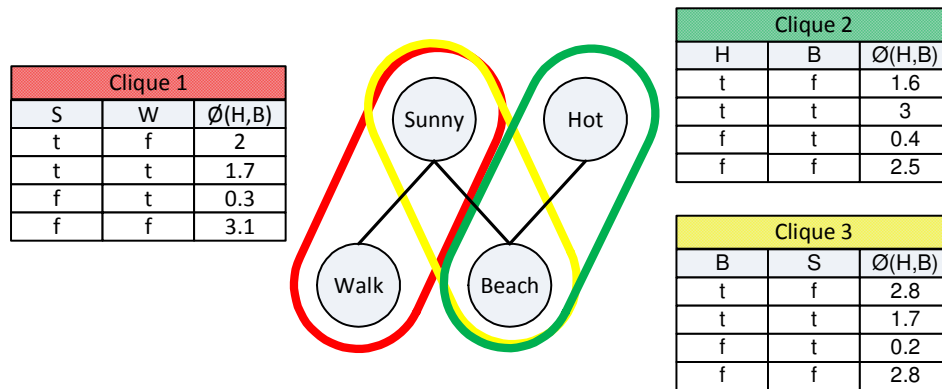


Figure 2. Simple Markov network with 4 binary random variables and their potential functions. Clique 1 is composed by nodes {Sunny, Walk}, Clique 2 by {Hot, Beach}, and Clique 3 by {Beach, Sunny}.

The joint probability distribution of set of random variables $X = \{X_1, \dots, X_n\}$ in a Markov network can be represented by

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}), \quad 2.6$$

where ϕ_k and $x_{\{k\}}$ are the potential function and the state of the k th clique, respectively. This way, the probability of a state can be obtained by multiplying the values of the potential function of all the cliques in that state. Since the sum of all

probabilities of all the states it is not 1, it is necessary to divide the formula with a normalizing constant, Z , called *partition function*. The partition function is defined by the sum of the product of potentials for all possible states

$$Z = \sum_{x \in X} \prod_k \phi_k(x_{\{k\}}). \quad 2.7$$

Just like in Bayesian networks, the probability of being sunny and hot, and we go to the beach but do not take a walk can be easily calculated by using the Markov network of Figure 2: $P(S = true, H = true, W = false, B = true) = \frac{1}{Z} (2 * 1.6 * 2.8) = \frac{8.96}{Z}$.

However, there is a problem with this representation. Since we need a value of the potential function for each state $x_{\{k\}}$ of a clique, the size of the representation is exponential in the size of the cliques. For example, if we have a clique with N binary nodes, there are 2^N possible states. So, it is often convenient to use a different way of specifying potentials that can ease this problem. This is done by using *log-linear models*, with each clique potential replaced by an exponentiated weighted sum of features of the state

$$P(X = x) = \frac{1}{Z} \exp \left(\sum_j w_j f_j(x) \right), \quad 2.8$$

$$Z = \sum_{x \in X} \exp \left(\sum_j w_j f_j(x) \right). \quad 2.9$$

Now, instead of potential functions, there are features $f_j(x)$, each one with an associated weight w_j . A *feature* is a real-valued function of a state. Each feature can represent more than one state of a clique, being this way a more compact representation than potential functions. If a feature is represented for each state of each clique, the log-linear model can be easily translated to the potential function model, and vice-versa. So, both models can be used interchangeably, depending on the purposes. A simple feature of the previous example can be defined as:

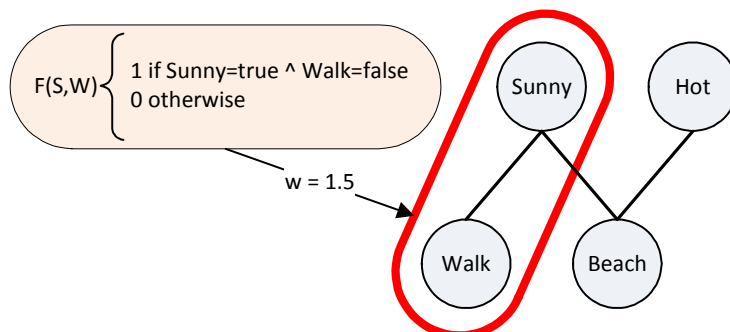


Figure 3. Simple Markov network with 4 binary random variables and one example feature for the clique {Sunny, Walk}.

Like Bayesian networks, Markov networks also encode useful information about the independence between random variables. However, independence in Markov networks is way easier to assert: a node is conditionally independent of the rest of the nodes in the graph given only its immediate neighbors (usually called the *Markov blanket* of a node) (e.g., *Beach* is conditionally independent of *Walk* given *Sunny* and *Hot*).

Comparison between Probabilistic Graphical Models

As seen in the previous sections, both Bayesian networks and Markov networks provide a compact representation of the joint probability distribution over a set of random variables. However, they are both different in many important aspects:

Property	Bayesian Network	Markov Network
Graph	Directed	Undirected
Parameterization	Conditional probabilities	Potential function/Features
Cycles	Not Allowed	Allowed
Partition Function	Not necessary ($Z=1$)	Necessary
Independence Checking	Parents/D-Separation	Neighbors

Table 7. Comparison between Bayesian networks and Markov networks.

Both approaches have advantages and disadvantages. While Bayesian networks are easier to model and understand, they do not allow cycles and the independence between variables is harder to determine. Markov networks allow cycles and simple independence checking, but its parameterization is hard to understand, and the calculations of the partition function can be very intensive in large domains. So, when choosing a representation, the domain must be carefully studied to determine the best relation between advantages/disadvantages offered by these representations.

However, for the purpose of the following section, both representations are treated as the same model representation. It is studied that Bayesian networks are a special case of Markov networks, and they can be converted in a Markov network by the construction of a *moral graph*, where the directed edges are transformed in undirected ones, and conditional probability distributions are contained in cliques (for a complete explanation of this process see (Bishop 2007)).

Until now, only the general properties of probabilistic graphical models were provided. In the next section, efficient inference techniques for these representations are demonstrated.

2.2.3. Inference in Probabilistic Graphical Models

Since probabilistic graphical models can represent the full joint probability distribution over the set of random variables $X = \{X_1, \dots, X_n\}$, they can be used to

answer any probabilistic query about the world. There are two main types of queries (Roller et al. 2007):

- The *conditional probability query* $P(Y | E=e)$, composed by two parts: the *evidence*, a set of random variables E and their observed values e ; and the *query*, a set Y of random variables. Our task is to compute the probability distribution over the possible values y of Y , conditioned on the fact that $E=e$. For example, $P(\text{Beach} | \text{Sunny}=\text{true}, \text{Hot}=\text{true})$ will give two probability distributions: one when $\text{Beach}=\text{true}$ and another when $\text{Beach}=\text{false}$;
- The most probable assignment to some subset of variables given the evidence $E=e$. This type of query has two variants:
 - *Most probable explanation* (MPE), when we want to find the *most likely assignment* to all of the *non-evidence variables*, i.e., if $W=X-E$, and we want to find the most likely assignment to W given the fact $E=e$. This can be defined as $\text{argmax}_w P(w, e)$. For example, a MPE query given $\{\text{Sunny}=\text{true}, \text{Hot}=\text{true}\}$ will give the most likely assignment to the variables *Beach* and *Walk*;
 - *Maximum a posteriori* (MAP), when we want to find the most likely assignment to a set of variable Y given the evidence $E=e$, i.e., $\text{argmax}_y P(y|e)$. For example, a MAP query of *Beach* given $\{\text{Sunny}=\text{true}, \text{Hot}=\text{true}\}$ will give the most likely assignment to the variable *Beach*.

By generating the full joint distribution of the random variables, all of these queries can be answered in a simple way: in a conditional probability query, we just sum the corresponding entries in the joint distribution; in MPE queries we just search for the most likely entries of the non-evidence variables in the joint distribution; and in the MAP query, we must sum the entries like the conditional probability query and search for the most likely entry. However, generating the full joint distribution is impractical in many cases. So, specialized algorithms were developed to overcome this limitation. There are two main classes of those algorithms: *exact algorithms* and *approximate algorithms*.

Exact Algorithms

If $Z=X-Y-E$ is a set of random variables, usually called *hidden variables*, a conditional probability query of Y given $E=e$ can be calculated as

$$\sum_Z P(Y, Z | E = e). \tag{2.10}$$

For example, given the Markov network of Figure 2, $P(\text{Beach} | \text{Sunny} = \text{true}, \text{Hot} = \text{true}) = P(\text{Beach}, \text{Walk} | \text{Sunny} = \text{true}, \text{Hot} = \text{true})$. However, in the worst case, the complexity of this algorithm, in the case of n binary variables, is $O(n2^n)$. The biggest reason of this complexity is that there are many repeated calculations. For this purpose, algorithms based on dynamic

programming, like the *variable elimination algorithm* (Stuart Russell and Norvig 2002), can be used. This algorithm evaluates the expression from right-to-left and save the intermediate results, avoiding repeated calculations. In most of the cases, the complexity is reduced to $O(2^n)$. More advanced algorithms, like *join trees* (Jordan 2004), can be used to reduce the complexity to $O(n)$ in the best case. However, even if the complexity of the algorithm is reduced, this problem continues to be *NP-Hard*, requiring exponential time and space to construct the graphical model representation. Approximate algorithms were developed to overcome this limitation.

Approximate Algorithms

The most used approximate inference algorithms are based on *randomized sampling*, being the calculations made in a set of random samples taken from the distribution represented by the graphical model. *Markov Chain Monte Carlo* (MCMC) (Stuart Russell and Norvig 2002) is one of these algorithms.

The algorithm starts with a random state, representing the first sample, and iteratively generates the next samples by sampling the value of one of the non-evidence variables Z_i , conditioned on the current values of the variables in the Markov blanket of Z_i . For example, given the Markov network of Figure 2, if we want to calculate $P(\text{Beach}|\text{Sunny} = \text{true}, \text{Hot} = \text{true})$, the first state can be $\{\text{Beach}=\text{true}, \text{Walk}=\text{false}, \text{Sunny}=\text{true}, \text{Hot}=\text{true}\}$. In the next state, we can sample *Walk* given its Markov Blanket, i.e., $P(\text{Walk}|\text{Sunny} = \text{true})$. If the random sample is $\text{Walk}=\text{true}$, the next state is $\{\text{Beach}=\text{true}, \text{Walk}=\text{true}, \text{Sunny}=\text{true}, \text{Hot}=\text{true}\}$. Basically, the algorithm wanders randomly through the search space, flipping the values of the non-evidence variables, maintaining the evidence variables fixed. The visited states are used to estimate the value of the query. For example, if we have 8 states where $\text{Beach}=\text{true}$ and 2 states where $\text{Beach}=\text{false}$,
 $P(\text{Beach} = \text{true}|\text{Sunny} = \text{true}, \text{Hot} = \text{true}) = 0.8$ and
 $P(\text{Beach} = \text{false}|\text{Sunny} = \text{true}, \text{Hot} = \text{true}) = 0.2$.

2.3. Statistic Relational Learning

Real world data is characterized by high degrees of relational complexity and uncertainty. For example, to assert if a webpage belongs to a certain topic, not only the words in the page have to be taken in account, but also its relations (*hyperlinks*) with another pages. Since webpages can influence each other results, there is a *collective classification* of webpages, where the results for all the webpages are simultaneously decided by using the correlation between pages. Webpages are also characterized by noisy and incomplete information. For example, certain words can be ambiguous (e.g., *Wordnet* (Fellbaum 1998) identifies 18 different senses¹ to the word *bank*), and they can only disambiguated with a certain degree of confidence. So, when dealing with this kind of domains, systems must be prepared to deal with uncertainty.

However, most of the current techniques are not prepared to deal with this type of information. *Statistical relational learning* (SRL) (Lise Getoor and Ben Taskar 2007) is a new area of research that attempts to represent, reason, and learn in domains with complex relational and rich probabilistic structure. SRL tries to combine ideas from the area of *statistical learning*, which deals with uncertainty, and the area of *relational learning*, which deals with complexity, in a unifying representation.

Most of the SRL approaches can be segmented by its *representation formalism*, dealing with the complexity problem, and by the *probabilistic semantics*, dealing with the uncertainty. Representation is usually based on either logic (e.g., first-order logic, logic programs) or frame-based (e.g., entity-relationship models, object-oriented). Probabilistic semantics are mostly based on probabilistic graphical models (e.g., Bayesian networks, Markov networks) or stochastic grammars. Based on this segmentation, the most relevant SRL approaches can be represented in a taxonomy (Figure 4).

¹ <http://wordnetweb.princeton.edu/perl/webwn?s=bank>

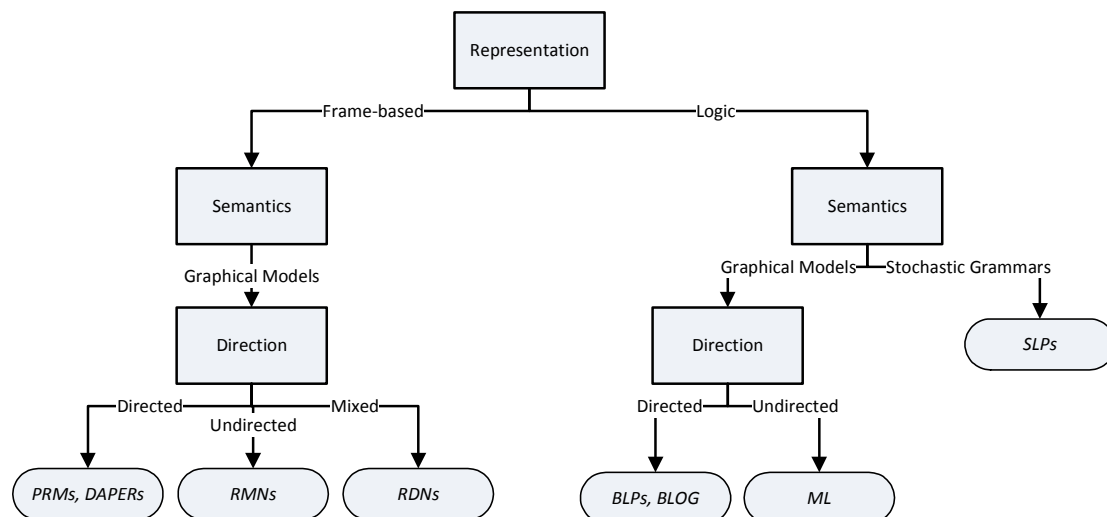


Figure 4. Statistical relational learning taxonomy.

Most of the frame-based systems use probabilistic graphical models as the underlying probabilistic semantic. *Probabilistic Relation Models* (PRMs) (U. Getoor et al. 2007) use pure frame-based systems to extend Bayesian networks with the concepts of objects, their properties, and relations between them. This way, Bayesian networks can naturally deal with relational complexity. Given a database of objects and their relations, a PRM define a probability distribution over the attributes of the objects. PRMs can be manually built or learned from existing databases. *Direct Acyclic Probabilistic Entity-Relationship Models* (DAPERS) (Heckerman, Meek, and Koller 2007) also use Bayesian networks as the probabilistic semantics. However, DAPERS focus on a more expressive frame-based representation than PRMs: *entity-relationship* (ER) models. They extend ER models to handle probabilistic relationships, creating a statistical modeling tool to model probabilistic ERs.

Relational Markov Networks (RMNs) (B. Taskar et al. 2007) combine relational models and Markov networks, providing a probabilistic modeling tool to model relational models. RMNs provide a compact way of representing Markov networks over a relational domain by specifying the cliques and potentials between attributes of related entities. Since an RMN provides a model of the structure of the Markov network, it can be used to provide a coherent distribution over any instance (or collection of instances) that fits in this model. The uncertainty of the model (i.e., its parameterization) can be learned from example data by using *discriminative learning* or *conjugate gradient* methods combined with *belief propagation*.

Relational Dependency Networks (RDNs) (Neville and Jensen 2007) is an extension of dependency networks for relational data. *Dependency networks* are mixed probabilistic graphical models that combine in the same representation both directed and undirected relations between variables. The main difference to Bayesian and Markov networks, apart its mixed representation, is that dependency networks are a purely approximate models (they do not guarantee a coherent

probability distribution). By using *sample-based* techniques, inference and learning can be easily implemented in RDNs.

Logic is another representation used in SRL approaches. *Stochastic Logic Programs* (SLPs) (Muggleton and Pahlavi 2007) combine logic programs with stochastic grammars. *Logic programs*, in their most basic form, are a subset of first-order logic. They provide a simple and powerful language to represent possible theories of a world in a rule-like syntax. Each rule is composed by a head, also called conclusion, followed by a rule body, also called premise. Logic programs are the base of many logic programming environments, like *Prolog* (Sterling and Eshud Shapiro 1994). *Stochastic grammars* are grammars where each production has an associated probability. This way, when deriving, some grammatical derivations are more relevant than another's. Since logic programs are more expressive than stochastic grammars, the main idea behind SLPs is to lift stochastic grammars to the expressive level of logic programs, by combining both approaches. Both parameters and structure can be easily learned from facts or clauses by using *expectation maximization* techniques and *beam-search*, respectively. *Bayesian Logic Programs* (BLPs) (Rersting and De Raedt 2007) also use logic programs as the underlying representation formalism. However, their semantics are provided by Bayesian networks. Just like in PRMs, Bayesian networks are extended with the concepts of objects, their properties, and relations between them. The difference is that BLPs use logic programs as the formalism to provide those features.

Bayesian Logic (BLOG) (Milch et al. 2007) combines first-order logic with Bayesian networks in the same representation. Compared to BLPs, BLOG is way more expressive, making possible to model more complex domains. BLOG is especially designed to deal with *unknown objects*, i.e., when there are no information about some objects and the system must infer that they actually exist. *Markov Logic* (ML) (Pedro Domingos, Stanley Kok, et al. 2008) combines first-order logic with Markov networks. Each first-order formula has an associated weight, representing a *Markov logic network* (MLN). This MLN can be seen as a template to construct Markov networks from given sets of constants: each ground atom is a variable, logical connectives are the edges between variables, and each grounded formula is a feature. The resulting Markov networks give a probability distribution over the possible worlds. Parameters can be learned generatively or discriminatively, and structure can be learned or refined by using bottom-up structure learning algorithms.

2.4. Semantic Web

The Semantic Web envisions a world where agents share and transfer structured knowledge in an open and semi-automatic way. In this section, the main concepts and technologies behind that vision are described. First, the main reasons behind the need for a Semantic Web are identified, followed by an explanation of its layered architecture.

2.4.1. From the Syntactic Web to the Semantic Web

In the current *World Wide Web* (WWW) most of the information is represented in *Hyper Text Markup Language* (HTML)². This language was made to represent visually the information to the user, and no efforts have been made to make it understandable by machines. For example, consider this excerpt of a simple HTML web page and its Web-browser visualization:

-
1. `<h1>Pedro Oliveira</h1>`
 2. `<p>MSc Student in Informatics Engineering</p>`
 3. `<p>University of Coimbra, Portugal</p>`
-

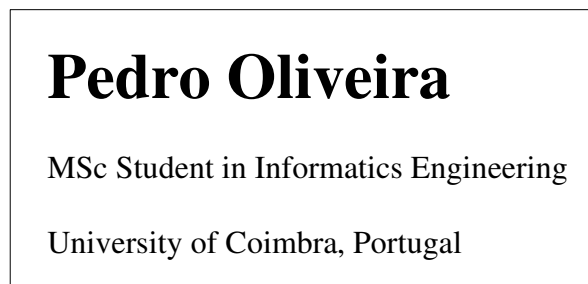


Figure 5. Visual representation of the previous HTML.

By seeing the page, the user understand that this page is dedicated to a person, named *Pedro Oliveira*, an *MSc* student in *Informatics Engineering* at the *University of Coimbra* in Portugal. But to the machine, the only information that it can understand is that there is a header that says *Pedro Oliveira*, with two more paragraphs. But what is *Pedro Oliveira*? Is it a person? A place? Or is just some random meaningless words that the web developer chooses to give to the header of the page? And what is the relation of the header with the proceeding paragraphs?

² <http://www.w3.org/html/>

This web is usually called the *Syntactic Web* (Breitman, Casanova, and Truszkowski 2006), where information presentation is carried out by computers, and the interpretation and identification of relevant information is delegated to human beings. There is no meaning associated with the text, being the markup metadata (e.g., *<h1>*) only used to visually represent the related text. For the machine it is only possible to know the *syntactic* information of the text, i.e., if it is a verb, a pronoun, a number, etc. There is no information about the meaning, i.e., the *semantics*, of the text.

In the last years, the scientific community put a lot of effort on trying to extract meaning from unstructured data, like those present in the WWW. Some fields like *Information Retrieval* (Baeza-Yates and Ribeiro-Neto 1999), *Machine Learning* (Bishop 2007), and *Natural Language Processing* (Jurafsky and Martin 2008) have produced increasingly complex systems for this task. Some of these systems are the base of very large companies like *Google*³, *Yahoo*⁴, and *SAS*⁵. Although, in all of those systems there is still currently a *knowledge gap* (Mika 2007) between what the machine understands and is able to work with, and what the user knows and infers about the data. This handicap from the machine is mainly consequence of technological difficulties in understanding the contents of a webpage. Since the contents are expressed in natural language, images, videos, or other complex representations, it is very difficult to the machine recognize the meaning of those elements.

Most of the times this handicap comes from the lack of background knowledge of the machine. In the previous example, we know that if something has a name and has some information about a scholar position, it is probable that this scholar position is of the person referred by the name. Although it is easy to make this assertion by a person, this task is difficult for a machine. Even if the machine can identify that a portion of text is a name or a scholar position, how can it determine that a scholar position is normally related to a person if that information is not stated anywhere? This is where the Semantic Web enters.

Envisioned in 1998 (Tim Berners-Lee 1998) by Tim Berners-Lee, the inventor of the WWW and HTML, the *Semantic Web* tries to fill the knowledge gap between human and machine. The main objective of the Semantic Web (T. Berners-Lee, J. Hendler, and Lassila 2001) is to “bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page

³ <http://www.google.com/>

⁴ <http://www.yahoo.com/>

⁵ <http://www.sas.com/>

can readily carry out sophisticated tasks for users.” With that meaningful knowledge, software agents while reading a page can not only assert the keywords and links of the document, like today search engines, but also complex relations between the elements of the page, just like those described in the later example.

This Semantic Web is not intended to be apart from the current Web. There is no effort on making a new Web, where all the information is somehow more intelligent and structured than it is today. The Semantic Web is not a separable Web, but an extension to the current one. In the future Web, there will coexist “normal” Web Pages and Semantic Web pages, with the only difference being the later ones more complete to the desires of the user. This way, Semantic Web is trying to expand a web of documents for people with a web of information for machines.

To realize this vision it is necessary to formalize technologies, tools, and standards that provide the ability to incorporate meaningful information onto Web pages. Those technologies need to express both data and rules for reasoning about data, allowing machines to execute complex tasks and better cooperate with humans. In the next sections, these technologies will be described.

2.4.2. Semantic Web Architecture

As stated before, a set of technologies, tools, and standards is needed to realize the Semantic Web vision. A *Layered Architecture* (Tim Berners-Lee 2000), composed by a series of standards structurally organized, was proposed (Figure 6). In this architecture, interrelationships of growing complexity between standards are structurally represented.

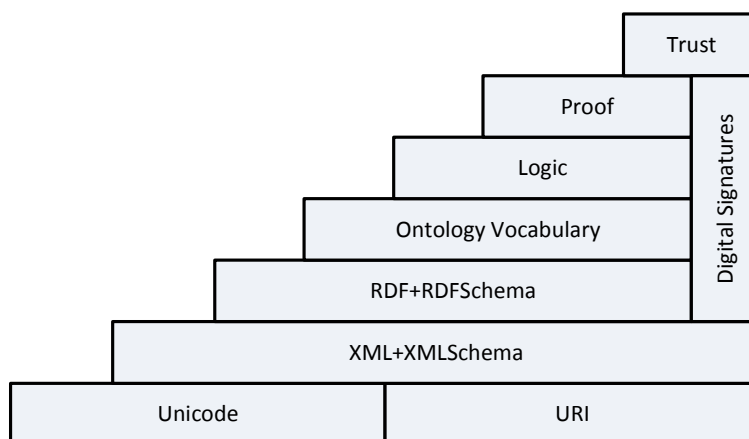


Figure 6. Semantic Web layered architecture (adapted from (Tim Berners-Lee 2000)).

Its foundation is composed by *Unicode* and *URI*, both responsible for the identification of resources. Above, there are 3 layers of representation: *XML+XMLSchema* that provides a syntactic operability layer, letting users write structured Web documents with a user-defined vocabulary; the *RDF+RDFSchemata* layer defines a basic data model to write statements about resources using XML; and the *Ontology Vocabulary*, composed of a more complex representation of

knowledge. At the top, the most complex layers: *Logic*, *Proof*, and *Trust*. Some layers rely on *Digital Signatures* to ensure security. In the next sections, all of these layers are explained.

2.4.3. URI and Unicode

Before explaining URIs and Unicode, it is useful to define the most basic unit in the Semantic Web: the resource. A *resource* (Breitman, Casanova, and Truszkowski 2006) is “anything that has an identity, be it a retrievable digital entity, a physical entity, or a collection of other resources”. All the information in the Semantic Web is represented as resources. So, it is necessary to define a language to refer to those resources.

A *Universal Resource Identifier* (URI) (Berners-Lee 2005) is a formatted string that globally identifies an abstract or physical resource. The most used type of URI is the *Universal Resource Locator* (URL), which identifies resources via an abstract identification of the resource location (e.g., <http://student.dei.uc.pt/pcoliv>).

URIs can have a *fragment identifier* attached, preceded by a #, meaning that a resource is subordinated to another, primary resource. In the Semantic Web these are normally used to identify resources in the same namespace, and are usually called *URIRefs* (URI References) or *Hash URIs* (Sauer mann and Cyganiak 2008). For example, <http://student.dei.uc.pt/pcoliv#a> and <http://student.dei.uc.pt/pcoliv#b> means that that resources *a* and *b* are in the same namespace, <http://student.dei.uc.pt/pcoliv>.

URIs can be used in two ways: by using the *absolute URI* that identifies a resource independently of the context in which the URI appears (e.g., <http://student.dei.uc.pt/pcoliv#a>); or by using a *relative URI* with some prefix, previously declared, omitted (e.g., [pcoliv#a](#)).

To promote interoperability between knowledge sources in the Semantic Web it is necessary that all the parts involved use the same encoding to refer to resources. This is done by using *Unicode* (The Unicode Consortium 2006), an encoding system that provides a unique number for every character, independently of the platform, program, or language.

2.4.4. XML

The *Extensible Markup Language* (XML) (Antoniou and Harmelen 2008) is a general-purpose markup language, designed to describe structured documents. In contrast to HTML, users can construct their own tags in XML (e.g., `<scholar_position>`). XML does not provide semantics indicating how to visualize documents. It is a language more suitable to machines, being normally used in data integration, interoperability, and exchange tasks (Jorge Cardoso 2007).

A XML document is composed of plain text and markups (in the form of *tags*). The example from Section 2.4.1 can be easily represented in XML:

-
1. <?xml version="1.0" encoding="UTF-8" ?>
 2. <page>
 3. <person personid="http://student.dei.uc.pt/pcoliv">
 4. <name>Pedro Oliveira</name>
 5. <scholar_position>MSc Student in Informatics Engineering</scholar_position>
 6. <address>University of Coimbra, Portugal</address>
 7. </person>
 8. </page>
-

A XML document usually starts with a *preprocessing* instruction (line 1) defining some general parameters like the XML version and encoding, being followed by a set of *elements* consisted by a *start tag* (e.g., <name>), the *content* (e.g., Pedro Oliveira) and an *end tag* (e.g., </name>). Elements can be nested, like <name> is nested in <person>, being a valid XML document a balanced tree of nested elements (Figure 7). An element can have one or more name-value pair attributes inside the opening tag, like *personid*="http://student.dei.uc.pt/pcoliv" in <person>.

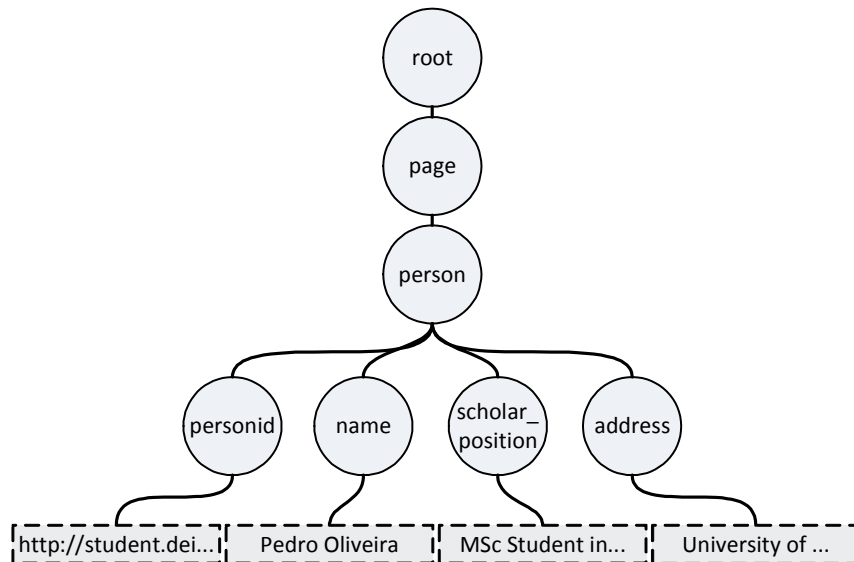


Figure 7. XML document tree of the previous XML example.

To be considered a well-formed XML document, the document must satisfy some syntactic constraints defined in a formal grammar, such as every start tag must have an end tag, attributes within an element have unique names, among others (Antoniou and Harmelen 2008). Besides those restrictions, users can impose its own restrictions in the structure of the document using a *Document Type Definition* (DTD) (Antoniou and Harmelen 2008).

A DTD is a XML-based language used to design constraints on the construction of a XML document. Using DTD, users can define all possible elements, their attributes, and even the allowed structure of a XML document. A simple DTD of the previous example can be easily defined:

-
1. <!ELEMENT page (person+)>
 2. <!ATTLIST person
 3. personid CDATA #REQUIRED
 4. >
 5. <!ELEMENT person (name)>
-

In this DTD, each page can have one or more persons (line 1). Each person must have an attribute *personid* (lines 2-4), and one *name* element (line 5).

XMLSchema (Antoniou and Harmelen 2008) offers a significantly richer language for defining the structure of XML documents. The main innovation comparing to DTD is that XMLSchema uses a pure XML language to express the constraints on the structure. XMLSchema support the same restrictions of DTD, adding some other features like support for basic data types, constraints on attributes, sophisticated structures and the use of namespaces to allow the combination of multiple schemas. The same restrictions of the previous DTD can now be stated in XMLSchema:

-
1. <xsd:schema
 2. xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
 3. version="1.0">
 4. <xsd:complexType name="page">
 5. <xsd:element name="person" minOccurs="1">
 6. </xsd:complexType>
 7. <xsd:complexType name="person">
 8. <xsd:attribute name="personid" type="string">
 9. <xsd:element name="name">
 10. </xsd:complexType>
 11. </xsd:schema>
-

2.4.5. RDF

In the last section, we have seen that XML is a very good choice to express knowledge in a structured way. It provides a general framework, with a consistent syntax, for interchange of data and metadata between applications. However, XML does not provide any way of adding meaning (i.e., semantics) to the data. By nesting elements and defining properties, meaning is not being associated to those elements. They are just being structured; each application decides how to interpret those elements.

Resource Description Framework (RDF) (Breitman, Casanova, and Truszkowski 2006) (Antoniou and Harmelen 2008) is a data model proposed to fulfill the need of giving meaning to XML structured information. An RDF document is composed by a set of statements. Each statement is a triple composed of:

- *Subject*, the resource which the statement refers;
- *Predicate*, also called property, describing relations between resources;
- *Object*, which represents the value of the property, and can be a resource or an atomic value called *literal*.

There are two common ways of textually representing statements: *triples* of the form *(Subject, Predicate, Object)* or in a functional style with binary predicates *Predicate(Subject, Object)*. The later is adopted in this work. Visually, a statement can be seen as a *directed graph* (DAG).

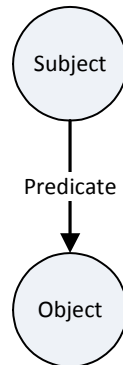


Figure 8. Visual representation of a RDF statement.

Since a RDF document is a set of statements, visually it is also a DAG. For example, the DAG of three statements modeling the domain presented in Section 2.4.1 can be seen on Figure 9.

-
1. Name(<http://student.dei.uc.pt/pcoliv#me>, "Pedro Oliveira")
 2. Course(<http://student.dei.uc.pt/pcoliv#me>, http://www.dei.uc.pt/courses#msc_ie)
 3. University(http://www.dei.uc.pt/courses#msc_ie, <http://www.uc.pt/univ#us>)
-

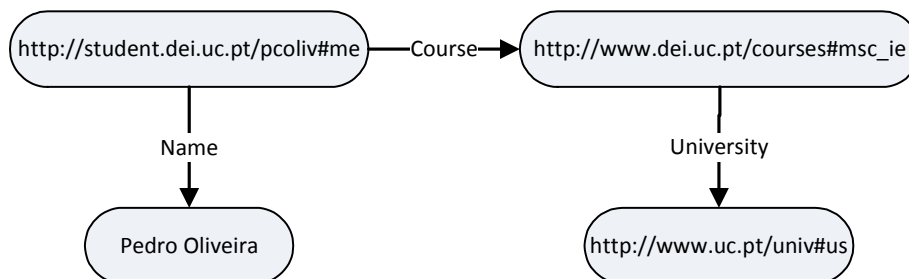


Figure 9. Visual representation of three RDF statements.

Although graphs and triples/functions are useful representations to human understanding, they are not so convenient to the storage, retrieval, and exchange of documents between machines. So, usually, RDF documents are stored in XML:

```
1. <!DOCTYPE rdf:RDF [  
2. <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">  
3. ]>  
4. <rdf:RDF  
5.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
6.   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"  
7.   xmlns:uni="http://www.university.org/uni-ns#">  
8.   <rdf:Description rdf:about="http://student.dei.uc.pt/pcoliv#me">  
9.     <uni:Name>Pedro Oliveira</uni:Name>  
10.    <uni:Course rdf:resource="http://www.dei.uc.pt/courses#msc_ie"/>  
11.  </rdf:Description>  
12.  <rdf:Description rdf:about="http://www.dei.uc.pt/courses#msc_ie">  
13.    <uni:University rdf:resource="http://www.uc.pt/univ#us"/>  
14.  </rdf:Description>  
15. </rdf:RDF>
```

First, *document type* (lines 1-3) and available *namespaces* (lines 4-7) are declared. Resources belonging to declared namespaces can be identified by a relative URI (e.g., *rdf:Description* instead of *http://www.w3.org/1999/02/22-rdf-syntax-ns#Description*). Statements are described in *rdf:Description* elements, using *rdf:about* to define the *Resource*, and representing the *Predicate* and *Object* as XML tags (lines 9-12 and 14-16). To refer an object that is also a resource (i.e., it is not a literal), we use the property *rdf:resource* (lines 11 and 15).

RDF is a very flexible data model, letting users describe resources using their own vocabulary. It doesn't provide any means for defining domain specific classes and properties. The only defined property is *rdf:type*, which defines the type of any object in the domain. Therefore, a way is needed to define the possible terms used to specify the resources, properties and values referred on RDF statements, and the way resources can be related to each other.

RDF Schema (RDFS) (Antoniou and Harmelen 2008) is an extension to RDF providing a vocabulary to specify classes (*rdfs:class*) and their relations (*rdfs:subClassOf*), properties (*rdfs:property*) and their relations (*rdfs:subPropertyOf*), literals (*rdfs:literal*), annotations (*rdfs:comment* and *rdfs:label*), property restrictions (*rdfs:range* and *rdfs:domain*), among others. This way, users can define classes, individuals and properties in a simple hierarchical structure, using a global agreed vocabulary.

RDFS is fully compatible with RDF, being a well formed RDFS document also a valid RDF document. So, RDFS can also be represented in XML in the same way as RDF.

2.4.6. Ontology Vocabulary

The expressivity of RDF and RDFS is very limited. RDF only provides a simple vocabulary to express structured knowledge, while RDFS is limited to model hierarchies with simple restricted properties. However, more expressivity is needed to model some complex domains. One of the ways to provide this expressivity is by using ontologies.

An *ontology* can be easily defined (James Hendler 2001) as a set of knowledge terms for some particular topic, including the vocabulary, semantic interconnections and rules of logic/inference of those terms. In general, ontologies provide a shared and common understanding of a domain that can be communicated between people and/or machines (Wahlster, Lieberman, and James Hendler 2002). They are normally used to make explicit conceptualizations that describe the semantics of the data with the intuition of reasoning about that data.

The most prominent markup language proposed by the W3C to model ontologies in the Semantic Web is the *Web Ontology Language* (OWL) (Breitman, Casanova, and Truskowski 2006) (Antoniou and Harmelen 2008). OWL, just like RDF and RDFS, is defined as a vocabulary, but with stronger semantics than its predecessors. Some of the new relations are, for example, the inclusion of special properties (*owl:TransitiveProperty*, *owl:SymetricProperty*, *owl:FunctionalProperty*, and *owl:InverseFunctionalProperty*), enumerations (*owl:oneOf*), versioning information (*owl:versionInfo*), logical properties (*owl:intersectionOf*, *owl:sameAs*, and *owl:differentFrom*), among others.

However, more expressivity means more complexity. So, there are currently three sublanguages in OWL, each one with a different complexity:

- *OWL Full* is OWL with no restrictions. It is fully compatible with RDF/S, both semantically and syntactically. The problem is that the language makes the reasoning over it *undecidable* (i.e., some statements cannot be shown to be either true or false), making almost impossible a complete (or efficient) reasoning.
- *OWL DL* is a sublanguage of OWL that tries to restrict it to the well known Description Logic *SHOIN(D)* (Baader et al. 2007). This logic is known of being both decidable (all computations will finish in finite time) and complete (all conclusions are guaranteed to be computable), making its reasoning possible, at least in theory. Those restrictions are made by disallowing the applications of constructors between classes. The problem with this sublanguage is that it is not fully compatible with RDF. The applied restrictions force to make some changes in a valid RDF document to be considered a legal OWL DL document. However, a legal OWL DL document is also a legal RDF Document.
- *OWL Lite* is an even more restricted version than OWL DL, corresponding to the Description Logic *SHIF(D)*. By excluding enumerated classes, disjoint statements and arbitrary cardinality, a more easy (for both human and machine) sublanguage is created. Although, the expressivity is largely reduced compared to the other OWL sublanguages.

These three sublanguages create an internal sub layer in the Ontology Vocabulary layer in the Semantic Web Architecture (Figure 10). This sub layer shows an upward

compatibility (e.g., an OWL Lite ontology is also a valid OWL DL one) of growing complexity.



Figure 10. OWL sub layer.

More recently, a new extension to OWL, called *OWL2* (Grau et al. 2008) has been proposed. This new language extends OWL DL with a new set of features that have been requested by users. Extra syntactic constructors like *owl:DisjointUnion* and *owl:DisjointClasses*, new constructors for properties (e.g., *owl:ReflexiveProperty* and *owl:AsymmetricProperty*), extended datatypes capabilities, and new annotating features are some of the new features. Its Description Logic, *SROIQ(D)*, is also sound and complete.

2.4.7. Logic

One of the most powerful capabilities of the Semantic Web is *reasoning*, i.e., derive new facts and relations from existing knowledge. The main mechanism behind this reasoning is *logic* (Section 2.1). In fact, the biggest advance made by the Semantic Web is to add logic to the current Web, providing agents with the capability of using rules to make inference, choose courses of action and answer questions.

Take for example the three RDF statements of Figure 9. An agent confronted with this knowledge will infer that *Pedro Oliveira* is connected to the *University of Coimbra* through his *MSc* course, i.e., the *MSc* course serves as a transitive path between *Pedro Oliveira* and the *University of Coimbra*. So, when asked about individuals in the *University of Coimbra*, the agent will return *Pedro Oliveira*, even if that information is not explicitly represented.

As seen in the previous section, the semantics of some OWL languages are based on Description Logics (see Section 2.1.4), a well studied group of logical languages. Even if OWL Full does not have a predefined logic, it can be interpreted as belonging to the set of *high-order languages* (S. Shapiro 2001), which are very expressive logical languages that usually do not provide either sound or complete reasoning. These OWL languages allow us to understand one of the most important conclusions of logic as a knowledge representation mechanism: the difficulty of reasoning increases with the expressive power (Levesque and Brachman 1985). Since there is a tradeoff between the expressiveness of the representation language and its computational tractability, there is an increasing need in finding logical languages that are expressive enough, but remain with some interesting capabilities like soundness and completeness. The Semantic Web community has been extensively

researching on this theme, and the new language, OWL2 (Grau et al. 2008), is one of these attempts.

2.4.8. Digital Signatures, Proof and Trust

As stated in the previous section, reasoning is one of the most important capabilities of the Semantic Web. However, different reasoners can produce different results. Since in a fully distributed Semantic Web most of the time the reasoning could be made in a different machine than ours, for example using a *Semantic Web Service* (SWS) (Jorge Cardoso 2007), how can agents trust that the inference process was reliable and the result is what they were expecting? For example, if there is a SWS that gives the address of a person given its name, how can an agent be sure that the returned address is from the people that he want and not from another person with the same name?

This can be done by accompanying the results with a justification that allows the receiving agent to assert the quality of the results. This is the purpose of the *Proof layer*. One of the proposed languages to describe justification of results in the Semantic Web is the *Proof Markup Language* (PML) (Silva, McGuinness, and Fikes 2006), using OWL to exchange proofs between machines.

Trust is the top layer of the Semantic Web architecture. Since anyone can contribute to the Semantic Web, some form of trustworthiness must be given to the information. This can be achieved by using *Digital Signatures* (e.g., public key cryptography), trusted agent recommendations, and ratings from certificated agencies. This way a *Web of Trust* (Matthew Richardson, Agrawal, and Pedro Domingos 2003) could be created, were agents can be aware of the credibility and reliability of statements, presenting the most relevant information to the user.

2.5. Markov Logic

Handling uncertainty and complexity in the same representation has been a long goal of Artificial Intelligence. Markov logic is a novel language that provides that capability, joining in the same representation probabilistic graphical models (Markov networks) to handle uncertainty, and first-order logic to handle complexity. By attaching weights to first-order logic formulas, a Markov logic network can be built. This network can be used as a template to construct Markov networks, providing the full expressiveness of probabilistic graphical models and first-order logic.

In the next sections, Markov logic representation and capabilities are described.

2.5.1. Markov Logic Networks

Markov logic (M. Richardson and P. Domingos 2006) (Pedro Domingos, Stanley Kok, et al. 2008) combines first-order logic and Markov networks in the same representation. The main idea behind Markov logic is that, unlike first-order logic, a world that violates a formula is not invalid, but less probable. This is done by attaching weights to first-order logic formulas: the higher the weight, the bigger is the difference between a world that satisfies the formula and one that does not, other things being equal. Using these weighted formulas, a Markov logic network can be built.

A *Markov logic network* (MLN) (M. Richardson and P. Domingos 2006) L is a set of pairs (F_i, w_i) where F_i is a formula in first-order logic and w_i is a real value representing the weight of the formula. If a set of constants $C = \{c_1, \dots, c_n\}$ is provided, a Markov network $M_{L,C}$ can be defined as follows:

- A binary node is created for each possible grounding of each atom in L , being its value 1 if the ground atom is true, 0 otherwise.
- Each possible grounding of each formula F_i in L generates a distinct feature, being its value 1 if the ground formula is true, 0 otherwise. The weight of the feature is the w_i associated with the formula.

This way, it is created a node for each ground atom and an edge if two ground atoms appear in the same formula. Suppose a simple MLN with two formulas.

Formula	Weight
$\forall x Steal(x) \Rightarrow Prison(x)$	3
$\forall x \forall y CrimePartners(x, y) \wedge Steal(x) \Rightarrow Prison(y)$	1.5

Table 8. Markov logic network example.

Using the previous algorithm, if we have two constants, *Anna* and *Bob*, the resulting Markov network has eight variables, corresponding to eight grounded atoms.

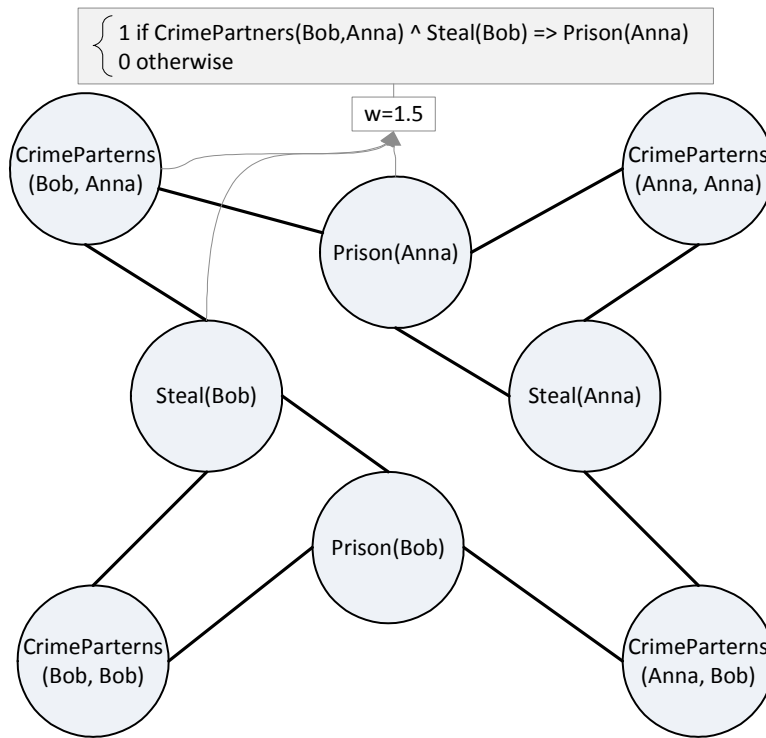


Figure 11. Markov network, with an example feature, built from the previous Markov logic network.

An MLN can be viewed as a template to construct Markov networks. With different sets of constants, different Markov networks are created. However, all of those networks have some kind of similarity in the structure or in the parameters. For example, all the groundings of a formula have the same weight, making the Markov networks of the different groundings of the same formula similar in structure. These grounded networks are called *grounded Markov networks*.

The probability distribution of a ground Markov network can be defined as

$$P(X = x) = \frac{1}{Z} \exp \left(\sum_{i=1}^F w_i n_i(x) \right), \quad 2.11$$

where F is the number of formulas in the MLN, $n_i(x)$ is the number of true groundings of F_i in the world x , and w_i is the weight of F_i .

Markov logic can easily represent many of the models usually used in Artificial Intelligence. For example, an MLN with only grounded formulas is a Markov network, and an MLN with all the weights equal and tending to infinity is a first-order logic knowledge base. Other models that use products of potentials, like Bayesian networks, can also be represented in Markov logic (M. Richardson 2004).

There are two main tasks that can be done by MLNs: inference and learning.

2.5.2. Inference

Since MLNs create Markov networks, the inference algorithms presented in Section 2.2.3 can be used in Markov logic. However, given the specific properties of MLNs, more efficient algorithms were developed. *Most probable explanation* (MPE) and *maximum a posteriori* (MAP) queries reduce to find the truth assignment that maximizes the sum of weights of satisfied clauses. This can be done using *MaxWalkSAT* (Selman, Kautz, and Cohen 1993), a local-search weighted satisfiability solver algorithm. However, this algorithm consumes many resources in sparse domains, since it requires the full grounding of the MLN. Lazy versions of MaxWalkSAT, like *LazySAT* (P. Singla and P. Domingos 2006), solve this problem by gradually grounding the MLN when its distinct parts are needed.

Another inference task is to find the marginal and conditional probabilities of a formula given an MLN and possibly other formulas as evidence. Since a probability of a formula is the sum of the probabilities of the world where it holds, the more ground atoms exist, the more difficult is the task. Approximate inference algorithms, like the previous referred *Markov Chain Monte Carlo* (MCMC) (Section 2.2.3), are usually used. However, MCMC is not efficient in domains where formulas with deterministic or near-deterministic dependencies exist (e.g., formulas with infinite weight) because these areas of the search space can be very difficult to traverse by simple flipping the value of the non-evidence variables. To solve this problem, we can use *MC-SAT* (H. Poon and P. Domingos 2006), a combination of MCMC and the *SampleSAT* satisfiability solver (Wei, Erenrich, and Selman 2004). MC-SAT uses *slice sampling* to help capturing the dependencies between variables, allowing jumping from these difficult areas.

2.5.3. Learning

There are two things that can be learned in a MLN: *parameters* (i.e., the weights) and *structure* (i.e., the features). Both types are learned from example data.

Weights can be learned generatively by maximizing the *pseudo-likelihood* (Besag 1975) of the data. This is done by adjusting the weight of each variable by the likelihood of the variable and his neighbors with the given data. If the model predicts that a feature is true less often than it really is in the data, then the weight is increased; otherwise, it is decreased. Since this procedure is made taking only in account the variable and his neighbors, sometimes weights can be overestimated. A solution when it is known a priori which atoms will be evidence and query is to use *discriminative learning* (P. Singla and P. Domingos 2005). It uses a similar approach to the former, but only takes in account the evidence and query atoms to do the learning, maximizing the conditional likelihood of the query predicates.

Features can be learned from an empty knowledge base (KB) or from an existing KB (making possible to refine an existing KB). The idea is to add all the single atoms to the MLN and iteratively try to joining them until finding an MLN that maximizes a certain evaluation metric based on the provided data. This evaluation can be done by

using a weighted version of the already defined pseudo-likelihood algorithm. Since this is a very intensive task, some improvements (S. Kok and P. Domingos 2005) are made to reduce the number of calculations, normally using approximation techniques that avoid the update of all the weights in each iteration. Instead of randomly joining atoms, relationship graphs between variables can be used to guide the learning (Mihalkova and Mooney 2007).

2.6. Related Work

This section is dedicated to related work in the field of reasoning in the Semantic Web. There are three main research themes relevant to this topic: *deterministic reasoning in the Semantic Web*, the most researched area by the Semantic Web reasoning community; *uncertainty in the Semantic Web*, where probabilistic reasoning is applied to the Semantic Web; and *vagueness in the Semantic Web*, where Semantic Web languages are extended with fuzzy logic concepts. At the end of this section, it is provided a summary with our most relevant conclusions about the related work.

2.6.1. Deterministic Reasoning in the Semantic Web

Since Semantic Web languages like RDF and OWL are based on crisp logic, deterministic reasoning is the main area of research in Semantic Web reasoning. In this section are presented some of the best reasoners in this area.

Most of the reasoners in this section use *tableau-based* (also called *tableaux*) decision procedures for inference (Ralf Moller and Volker Haarslev 2008). These methods check the *satisfiability* (i.e., if there is an assignment of variables that make a statement evaluate to true) of an ontology by constructing a graph representation from the logical model of the ontology. This way, satisfiability can be checked by traversing the graph and checking for inconsistencies. This graph representation is also useful to do more complex tasks, like concept satisfiability, classification, realization, and individual retrieval.

One of the most used reasoners is *Pellet*⁶ (Sirin et al. 2007), an open source *Java* OWL DL Reasoner. It provides sound and complete reasoning with OWL DL, being capable of reasoning with assertional and terminological knowledge. It also provides reasoning with user defined datatypes, support for ontology debugging and integration with rules formalisms like *Semantic Web Rule Language*⁷ (SWRL). It uses tableau Description Logic algorithms for consistency checking, making possible OWL DL concept satisfiability, classification, and realization.

*FaCT++*⁸ (Tsarkov and Ian Horrocks 2006) is a Description Logic reasoner supporting OWL DL and OWL 2. It is an open source reasoner made in C++. The main

⁶ <http://clarkparsia.com/pellet/>

⁷ <http://www.w3.org/Submission/SWRL/>

⁸ <http://code.google.com/p/factplusplus/>

characteristics of *FaCT++* are the use of highly optimized tableau algorithms, making it a very efficient assertional and terminological knowledge reasoner. Reasoning in *FaCT++* is segmented in two tasks: preprocessing, when the ontology is loaded and some rewriting optimizations are made; and classification, where the taxonomy of named concepts is calculated using satisfiability checking techniques.

(Stocker and Smith 2008) proposed *Owlgres*⁹, an open source scalable reasoner for OWL 2 DL-Lite. It provides efficient reasoning and querying over scalable persistent data storages, like *relational database managements systems* (RDBMS) (e.g., *PostgreSQL*¹⁰). By combining Description Logic reasoning techniques with efficient data management and retrieval of RDBMS, *Owlgres* provides conjunctive query answering using a subset of *SPARQL*¹¹, *SPARQL-DL*, transforming *SPARQL* queries in *SQL* queries.

*SHER*¹² (Scalable High Expressive Reasoner) (Dolby et al. 2007) is an OWL DL reasoner developed by *IBM*¹³. It is optimized to high performance reasoning over millions of facts in OWL DL ontologies. It is mainly used to answer conjunctive and membership queries over ontological resources. Its high performance is obtained by summarizing ontological data into a very compact representation, and by refining this data by filtering unnecessary facts to the queries.

*Hermit*¹⁴ (Boris Motik, Shearer, and Ian Horrocks 2007) is a *Java* OWL reasoner that uses a novel approach to Description Logic reasoning. This novel approach, called *hypertableau reasoning*, is a hybrid of resolution and tableau algorithms, being more efficient in some subsets of OWL ontologies.

*KAON2*¹⁵ (Boris Motik and Ulrike Sattler 2006) is an OWL DL management system that provides efficient Description Logic reasoning. Unlike most of the Description Logic reasoners, *KAON2* does not use any tableau methods. Instead, the ontology is reduced to a *disjunctive Datalog program* and the inference is made in this representation. This way, some well know deductive database techniques like *magic*

⁹ <http://pellet.owldl.com/owlgres>

¹⁰ <http://www.postgresql.org/>

¹¹ <http://www.w3.org/TR/rdf-sparql-query/>

¹² http://domino.research.ibm.com/comm/research_projects.nsf/pages/iaa.index.html

¹³ <http://www.ibm.com/>

¹⁴ <http://www.hermit-reasoner.com/>

¹⁵ <http://kaon2.semanticweb.org/>

sets and *join-order optimization* can be applied in OWL DL reasoning, improving the results on answering some types of queries.

*RACER*¹⁶ (Renamed ABox and Concept Expression Reasoner) (V. Haarslev and Möller 2003) is a Description Logic reasoner that can be used to reason over OWL DL ontologies. *RACER* uses optimized tableau calculus algorithms, supporting tasks like satisfiability, consistency, subsumption, and querying of ontological resources. The commercial version, *RacerPro*¹⁷, also supports rules using SWRL and can be used as an HTTP reasoning server.

(Tsarkov et al. 2004) describe work on using *Vampire*¹⁸ (A. Riazanov 2002), a general purpose first-order logic (FOL) reasoner, to reason with OWL DL. Since OWL DL is a subset of FOL, the authors translate the ontology to FOL and apply FOL reasoning techniques to it. They also translate SWRL rules into FOL and reason with those rules. The results were not as good as a general Description Logics reasoner, but since FOL is way more expressive than OWL DL, this approach can be useful in some more restrictive tasks, like testing and debugging of new tests.

2.6.2. Uncertainty in the Semantic Web

Some domains are uncertain by nature. Since Semantic Web languages, like OWL and RDF, are based on crisp logic, it is very difficult, if not impossible, to represent those domains in the Semantic Web. Most of the times, this uncertainty comes from our incapacity of asserting the veracity or falsity of a statement. This uncertainty is usually represented by a probability, i.e., a quantity representing our uncertainty.

There are currently two distinct approaches (Thomas Lukasiewicz and Umberto Straccia 2008) to add probabilistic knowledge in the Semantic Web: *Probabilistic Description Logics* tries to expand and modify the logic behind Description Logics with probabilistic knowledge; and *Probabilistic Semantic Web Languages* which tries to combine Semantic Web languages with probabilistic formalisms like Bayesian networks.

¹⁶ <http://www.racer-systems.com/>

¹⁷ <http://www.racer-systems.com/products/racerpro/index.phtml>

¹⁸ <http://www.voronkov.com/vampire.cgi>

Probabilistic Description Logics

The most expressive Description Logic with probabilistic knowledge is $\mathcal{P}\text{-}\mathcal{SHOIN}(\mathcal{D})$ (T. Lukasiewicz 2008) (Klinov and Bijan Parsia 2008). $\mathcal{P}\text{-}\mathcal{SHOIN}(\mathcal{D})$ is a probabilistic extension to $\mathcal{SHOIN}(\mathcal{D})$, the Description Logic behind OWL DL. Probabilities are represented by a new kind of axiom, called conditional constraints. *Conditional constraints* are composed by expressions of the form $(D|C)[l,u]$, where D is the evidence, C the conclusion, and $[l,u]$ is a probability interval. There are two types of conditional constraints: *generic constraints*, representing probabilistic relations between classes; and *individual constraints*, representing probabilistic information about the belonging of individuals to certain classes. Currently, there are two reasoners that implement this Description Logic: *Pronto*¹⁹ (Klinov 2008), a probabilistic OWL reasoner developed by the Pellet team, and *ContraBovemRufum*²⁰ (Nath and R. Moller 2008), a simple OWL probabilistic reasoner built on top of Racer. Recently, a tool²¹ for the evaluation of probabilistic Description Logic reasoners was also developed. Since OWL Lite is a subset of OWL DL, its Description Logic, $\mathcal{SHIF}(\mathcal{D})$, can also be extended with probabilistic knowledge, $\mathcal{P}\text{-}\mathcal{SHIF}(\mathcal{D})$ (T. Lukasiewicz 2008).

Very similar to the previous work is $\mathcal{P}\text{-}\mathcal{SHOQ}(\mathcal{D})$ (Giugno and Thomas Lukasiewicz 2002). This Description Logic is based on $\mathcal{SHOQ}(\mathcal{D})$, a Description Logic very similar to $\mathcal{SHOIN}(\mathcal{D})$. The only difference between them is that $\mathcal{SHOQ}(\mathcal{D})$ doesn't allow inverse properties.

There are also other Description Logics augmented with probabilistic knowledge, but none of those have the needed expressivity to comport the expressiveness of Semantic Web languages. An overview can be found on (Thomas Lukasiewicz and Umberto Straccia 2008).

Probabilistic Semantic Web Languages

Since the emergence of the first Semantic Web languages, some work has been done in representing probabilistic knowledge in those languages. Most of this work tries to add probabilistic capabilities to those languages without changing their logical foundations or their syntax, by combining them with known probabilistic formalisms.

¹⁹ <http://pellet.owldl.com/pronto>

²⁰ <http://www.sts.tu-harburg.de/~t.naeth/#Software>

²¹ <http://www2.cs.man.ac.uk/~klinovp/projects/prevaldl/index.html>

In (Fukushige 2005) is proposed a simple vocabulary to represent probabilistic knowledge in RDF. This vocabulary is composed by a set of classes and properties representing elements of Bayesian networks, making possible to link RDF statements to those elements. This way, marginal and conditional probabilities about statements can be easily represented and reasoned using Bayesian networks. *pRDF* (Udrea, Subrahmanian, and Majkic 2006) is a formal probabilistic extension to a subset of RDF/S, allowing probabilistic knowledge about classes and properties of individuals. Unlike the previous work, *pRDF* implements its own probabilistic logic, making possible to reason over assertional knowledge in acyclic RDF graphs. (Holi and Hyvönen 2006) presented a framework for representing uncertainty in simple RDFS taxonomies. They were particular interested in computing the degrees of *subsumption*, i.e., overlap, between concepts. By attaching weights, called *masses*, to concepts, a Bayesian network is built. Using the Bayesian network prior and conditional probabilities, an overlap table between concepts is easily built by using *evidence propagation* algorithms.

*PR-OWL*²² (Costa and Laskey 2005) is a probabilistic generalization of OWL based on *multi-entity Bayesian networks* (MEBNs) (Laskey and Costa 2005). MEBN logic combines Bayesian probability theory with first order logic, constructing Bayesian networks from parameterized fragments representing the probabilistic knowledge about a collection of related hypotheses. This way, probability distributions can be encoded in first-order theories. The main contribute of PR-OWL is the definition of an upper ontology that guides the development of probabilistic ontologies in OWL using MEBNs.

(Ding, Peng, and Rong Pan 2006) proposed *BayesOWL*, a framework to represent and reason OWL uncertain knowledge using Bayesian networks. They provide a set of rules and procedures to translate OWL DL concept taxonomies (i.e., class axioms and logical relations between classes) into Bayesian networks. The main idea in this translation is to transform all the classes in variables and all the predicates in arcs between the respective classes. Special variables are inserted to facilitate the modeling of relations between concepts, and to avoid cycles. A simple approach to annotate OWL DL statements with conditional and marginal probabilities is also provided, as methods to automatically construct and refine conditional probability tables. The resulting Bayesian network preserves the semantics of the original ontology, and supports ontology reasoning both within and across ontologies. This framework was successfully applied in ontology mapping tasks (Rong Pan et al. 2005).

²² <http://www.pr-owl.org/>

Similar to the previous work is *OntoBayes* (Yang and Calmet 2005), which combines OWL and Bayesian networks. They provide a simple method to annotate OWL with conditional, marginal, and full disjoint probabilities. Unlike the previous work, the Bayesian network is not automatically built. Users must annotate in OWL the dependencies between variables. This way, users are not restricted to a subset of OWL, like *BayesOWL*, but have the burden of annotate more elements. Given the OWL ontology, the probabilities and dependency annotations, a Bayesian network is easily built, and the inference is made on it. (Gu et al. 2004) also propose a similar approach to the last one, being more focused on reasoning over uncertain contexts represented in OWL.

(Henrik and Norbert 2006) describe work on probabilistic reasoning in two subsets of OWL Lite. They translate restricted OWL Lite ontologies into *Datalog*, a subset of first-order logic, and use a probabilistic extension of *Datalog*, *pDatalog*, to do probabilistic inference over the ontology. This approach was successfully used in automatic ontology matching tasks (Nottelmann and Umberto Straccia 2006). In (Predoiu and Stuckenschmidt 2007) a probabilistic framework for information integration and retrieval on the Semantic Web is proposed. Their approach uses *Bayesian Description Logic Programs*, a formalism that joins *Description Logic Programs* (DLP), a subset of *Datalog* without negation and without equality, with a fragment of *Bayesian Logic Programs*. In this representation, statements are translated to DLP rules with an attached probability. This way, a Bayesian network can be built from those annotated rules, providing a complete specification of the desired probability distribution.

2.6.3. Vagueness in the Semantic Web

Sometimes, real world domains are composed by imprecise or vague information. Again, Semantic Web languages are not ready to deal with this type of information. For example, in those languages, how can we model the fact that some resource is “fast” or “tall”?

The problem with this situation is that we are dealing with *vague concepts*, that are more or less true, but we do not have a precise definition of them. For example, saying that *Ferrari is fast* depends on the velocity of the Ferrari, but we cannot say that this statement is completely true or false because we do not know which velocity is considered fast (i.e., fast is a vague concept). One way of representing vague concepts is by assigning a value to them, saying that this concept is true to some degree represented by that value. For example, by saying that *Ferrari is fast* with the degree of truth 0.8, we are saying that Ferrari is relatively fast but not completely fast.

One way of dealing with vagueness is by using *fuzzy set theory* and *fuzzy logic* (Klir and Yuan 1995). Instead of having a true/false value, statements with vague concepts have a truth value, usually between $[0, 1]$, where 0 and 1 represents binary false and true, respectively. In the last years, many approaches have been proposed to extend Description Logics with fuzzy set theory. Next, the most relevant

approaches to this work are described. For a more complete overview see (Sanchez 2006) and (Thomas Lukasiewicz and Umberto Straccia 2008).

(Stoilos et al. 2005a) propose *f-OWL*, a fuzzy extension to OWL DL. In this extension, degrees of truthiness are added to OWL facts, representing the fuzziness of those facts. When a fact doesn't have any degree, it is interpreted as a binary true fact (i.e., degree of 1). In this approach, OWL syntax must be changed to cope with the addition of the membership degree. The reasoning is made by a new logic, called *f-SHOIN*. This logic extends *SHOIN* (without datatypes) to deal with fuzzy set theory. (Stoilos et al. 2005b) also proposed *f-SHIN*, a simpler fuzzy description logic that is base of *FIRE*²³, a fuzzy reasoner for the Semantic Web. A more complete fuzzy extension of *SHOIN* was proposed by (U. Straccia 2006a). Although the principles are very similar, this approach subsumes the previous one, being capable of supporting all the OWL DL language.

Another fuzzy Description Logic reasoner is *fuzzyDL*²⁴ (Bobillo and U. Straccia 2008), a fuzzy extension to *SHIF(D)*, the Description Logic behind OWL Lite. This reasoner supports various membership functions and distinct fuzzy logics, like *Zadeh semantics* and *Lukasiewicz logic*. This reasoner also supports classical Description Logic reasoning, being not restricted to fuzzy reasoning. In (U. Straccia 2006b), a fuzzy extension to a restricted subset of OWL DL, called *DL-Lite*, is proposed. This extension, called *f-DL-Lite*, supports fuzzy queries over fuzzy knowledge bases. (J. Z. Pan et al. 2007) extended the previous work, proposing new query answer languages to query fuzzy knowledge bases, by extending SPARQL to support membership degrees. The work is implemented in *ONTOSEARCH2*²⁵, a search and query engine for the Semantic Web.

2.6.4. Conclusions

Through the analysis of the previous related work, some general conclusions can be made:

- *There is little work on dealing with uncertainty and vagueness in the Semantic Web.* Even if these two areas are of the most importance to the future of the Semantic Web, only recently the scientific community has started to research them (the first publications about the subjects are from 2005). In the same

²³ <http://www.image.ece.ntua.gr/~nsimou/FiRE/>

²⁴ <http://gaia.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html>

²⁵ <http://www.ontosearch.org/>

year, a dedicated workshop about those subjects (*International Workshop on Uncertainty Reasoning for the Semantic Web*²⁶) was created;

- *Deterministic reasoning is more computationally efficient than all the other approaches.* Some of the systems seen in Section 2.6.1 support efficient reasoning over millions of ontological facts, while other systems, like *Pronto*, are restricted to only hundreds of them. The main reason is that deterministic reasoning is the main area of research in Semantic Web reasoning, having already developed very efficient algorithms, mainly those based on tableau procedures. These algorithms were already being improved in other older fields, like Description Logics;
- *Probabilistic Semantic Web languages are, usually, more computationally efficient than the other uncertain and vague approaches.* The main reason is that most of them use formalisms like probabilistic graphical models, which are well known and provide efficient reasoning mechanisms. Probabilistic and fuzzy Description Logics usually lack of efficient reasoning mechanisms, and their applications to real world domains are also not well studied;
- *All the works on uncertainty and vagueness in the Semantic Web rely on the principle that the uncertainty or vagueness of the ontology is already asserted.* To our knowledge, there is no work on extracting automatically this information from the ontology, or from other knowledge representations. However, there are many ontologies that are uncertain or vague by nature, but do not have any type of information denoting that fact. This fact leads to the need of develop efficient mechanisms to learn this information.
- *All the works on Probabilistic Semantic Web languages rely on probabilistic formalisms, like Bayesian networks, which do not allow cycles.* However, in many domains, knowledge is cyclic (e.g., the relations *FatherOf* and *SonOf* are cyclic). This fact limits the usability and expressiveness of these approaches in real world domains.

These conclusions identify the main problems and debilities of the related work. This information will be used in the definition of our proposed approach (Chapter 4).

²⁶ <http://c4i.gmu.edu/ursw/>

3. Technologies and Tools

In this chapter, we present the technologies and tools that will be used in this thesis. We define our programming platform and our supporting tools in the fields of the Semantic Web and Markov logic.

3.1. Programming Platform

This project will be developed using the *Java*²⁷ programming platform. *Java* is the largest and most active programming platform existent, mainly for its versatility, efficiency, platform portability, and security. *Java* provides not only a stable and simple platform to software development, but also a large community with hundreds of applications freely available.

This choice is not only result of the technical capabilities of *Java*, but also for being the most used language in the Semantic Web community (J. Cardoso 2007). In this work, we will use the *Sun's Java Platform Standard Edition*²⁸ (J2SE), which provides all the basic mechanisms to develop our application.

3.2. Supporting Tools

Next, we describe the supporting tools that will be used during this work. There are two main groups of tools: *Semantic Web Tools*, dedicated to the development and processing of OWL ontologies, and the *Markov Logic Tools*, dealing with the inference and learning procedures of Markov logic.

3.2.1. Semantic Web Tools

We plan to use two tools of the Semantic Web domain: *OWL API*, an OWL ontology processor, and *Protégé*, an ontology editor.

²⁷ <http://www.java.com>

²⁸ <http://java.sun.com/javase/>

OWL API

*OWL API*²⁹ (Horridge, S. Bechhofer, and Noppens 2007) is a *Java* open-source tool to process OWL ontologies. It provides a simple and easy *Application Interface* (API) to deal with OWL ontologies, providing fast and efficient in-memory ontology processing. The last version supports OWL2, the new Semantic Web Language proposed by the W3C. It can read and write OWL ontologies in various formats (XML, Functional Syntax, Turtle, among others), and provides a simple interface for integration with Semantic Web reasoners like *Pellet* and *FaCT++*.

The main difference of *OWL API* to others ontology processors, like *Jena*³⁰, is that it uses a functional syntax representation, representing ontological knowledge as *axioms* instead of triples. This way, a much cleaner and easier to understand representation is achieved, providing a frame style modeling, easier to access programmatically. *OWL API* also provides a *black box* OWL debugger, which can be used by reasoners to explain their results, and a fragment detector/expressivity checker, capable of determining ontologies' OWL subset and correspondent Description Logics expressivity.

OWL API is being developed by the *University of Manchester* under the international funded projects *CO-ODE*³¹ and *TONES*³².

Protégé

*Protégé*³³ (Gennari et al. 2003) is *Java* open source ontology editor. It provides a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Two ways of editing ontologies are available: *Protégé Frames*, to develop frame-based domains ontologies; and the *Protégé OWL*, to develop RDF and OWL ontologies. It can be accessed by a graphical user interface, or programmatically by the provided API. Protégé is constituted by three main versions:

- *Protégé 3*, supporting OWL, RDF(S), and Frames;
- *Protégé 4*, supporting OWL2;

²⁹ <http://owlapi.sourceforge.net/>

³⁰ <http://jena.sourceforge.net/>

³¹ <http://www.co-ode.org/>

³² <http://www.tonesproject.org/>

³³ <http://protege.stanford.edu/>

-
- *WebProtege*, a lightweight web-based ontology editor.

Protégé is mainly developed by the *Stanford University School of Medicine*, with the cooperation of the *University of Manchester*.

3.2.2. Markov Logic Tools

During our research, we found three tools about Markov logic: *Alchemy*, *PyMLNs*, and *Markov thebeast*. All of them implement different reasoning and learning algorithms.

Alchemy

*Alchemy*³⁴ (S. Kok et al. 2007) is a software package providing a series of algorithms for statistical relational learning and probabilistic logic inference, based on the Markov logic representation. It provides a series of state of the art algorithms for inference and learning in Markov logic:

Feature	Algorithms
Weight Learning	Generative and Discriminative (Voted Perceptron, Conjugate Gradient, and Newton's Method)
Structure Learning	Top-Down Learning
Inference	MAP/MPE inference and Probabilistic inference (MC-SAT, Gibbs Sampling, Simulated Tempering, and Belief Propagation)

Table 9. Alchemy features and algorithms.

Other interesting features, like support of continuous domains and online inference are also supported. The application is developed in *C++*, being provided both an API and a console interface. Currently it only supports natively *Unix* systems, but a porting to *Windows* systems with *Cygwin*³⁵ is possible.

Alchemy is developed by the *University of Washington* by the team headed by Pedro Domingos, one of the inventors of Markov logic.

³⁴ <http://alchemy.cs.washington.edu/>

³⁵ <http://www.cygwin.com/>

PyMLNs

*PyMLNs*³⁶ is a *Python* toolkit to work with Markov logic networks. It provides a *Python* implementation of many algorithms for inference and learning in Markov logic:

Feature	Algorithms
Weight Learning	Maximum likelihood (Log-likelihood and Pseudo-log-likelihood)
Inference	Probabilistic inference (Exact Inference, MC-SAT, and Gibbs Sampling)

Table 10. PyMLNs features and algorithms.

Some interesting features, like constraints on formulas probabilities and support for domains where the weights are in the interval $[0-1]$, are also provided. *PyMLNs* also provides a graphical user interface to work with Markov logic, using *Alchemy* or the intern engine as the Markov logic engine.

PyMLNs is developed by the *Technische Universität München*.

Markov thebeast

*Markov thebeast*³⁷ (Riedel 2008) is a statistical relational learning software based on Markov logic. It is programmed in *Java*, and provides a set of algorithms for Markov logic reasoning and learning:

Feature	Algorithms
Weight Learning	Online Discriminative Training (MIRA, Perceptron, and Passive Aggressive Learning)
Inference	MAP Inference (Cutting Planes combined with Integer Linear Programming (or Max-Walk-Sat))

Table 11. Markov thebeast features and algorithms.

³⁶ <http://www9.cs.tum.edu/people/jain/mlns/>

³⁷ <http://code.google.com/p/thebeast/>

Markov thebeast is specially designed to deal with domains with many features with individual weights, through the use of parameterized weights. It is developed by the *University of Tokyo*.

4. Proposed Approach

In this chapter, we describe our proposed approach. First, our solution strategy to the problem of probabilistic reasoning in the Semantic Web using Markov logic is described. Two distinct approaches to this task were identified: *reasoning in uncertainty-annotated ontologies*, and *reasoning with weight learning*. Next, we demonstrate the main features of the system that will be developed, and also how our work will be evaluated.

4.1. Solution strategy

The objective of this thesis is to study mechanisms to perform probabilistic reasoning in the Semantic Web (T. Berners-Lee, J. Hendler, and Lassila 2001). For this purpose, we will use Markov logic (Pedro Domingos, Stanley Kok, et al. 2008), a novel representation formalism that combines logic and probabilistic graphical models. Before explain our approach, we first need to clarify why we are using Markov logic.

As previously seen (Section 2.4.6), the most relevant Semantic Web languages to describe ontologies are based on Description Logics (Baader et al. 2007). These Description Logics follow a model-theoretic semantics, and therefore can be interpreted as a set of formulas in first-order logic. To perform probabilistic reasoning over those languages, we need formalisms that combine first-order logic and probability in the same representation. This requirement leads us to the field of statistical relation learning (Lise Getoor and Ben Taskar 2007). In this field, there are two main approaches that combine the full power of first-order logic and probabilistic graphical models: Markov logic and Bayesian logic (BLOG) (Milch et al. 2007). Between those two, Markov logic was chosen for a set of reasons:

- BLOG models are based on Bayesian networks, and therefore do not allow cycles. Since the Semantic Web domain is characterized by cyclic relations between entities (e.g., the relations *FatherOf* and *SonOf* are cyclic), this fact can restrict the use of BLOG in some ontologies. It is studied (Ding, Peng, and Rong Pan 2006) that these cycles can be removed by the introduction of auxiliary variables. However, this is a difficult task that increases the model complexity, and can be only used on a small subset of some Semantic Web languages;
- Even if BLOG allows first-order knowledge, models are procedurally defined by programs, an unnatural way of defining first-order knowledge. To model a simple fact, complex constructors like *guaranteed object statements* and *dependency statements* must be declared, while in Markov logic it suffices to declare the first-order logic formulas of the domain, and their weights;
- Several algorithms for learning and reasoning in Markov logic were studied (e.g., (P. Singla and P. Domingos 2005) (S. Kok and P. Domingos 2005) (P. Singla

and P. Domingos 2006) (H. Poon and P. Domingos 2006)). On other side, in BLOG only reasoning was deeply studied (Milch and S. Russell 2006) (Milch et al. 2008), and there is no work on learning the parameters or the structure of BLOG models.

After defining the reasons behind the decision for Markov logic, we are prepared to describe the use of Markov logic to perform probabilistic reasoning in the Semantic Web. As seen in Section 2.5, a Markov logic network is composed by a set of weighted first-order logic formulas. Since ontologies described through Semantic Web languages can be interpreted as first-order formulas, their weights must be found. We identified two approaches to acquire those weights: the ontologies already got their axioms annotated with some kind of uncertainty values that can be used as weights, or the weights have to be learned from example data.

4.1.1. Probabilistic Reasoning in Uncertainty-annotated Ontologies

Ontology axioms can be annotated with a value representing its uncertainty (usually a weight or a probability) (e.g., a certain class has $X\%$ probability of being subclass of another). This allows ontology engineers to build probabilistic ontologies with their own knowledge about the domain. However, this kind of reasoning is not only interesting in this case. There are already domains with ontologies with some kind of uncertainty associated:

- In the field of *ontology learning* (Maedche 2002), mainly from text corpus, the resulting ontologies usually have a probability associated with the confidence on the asserted relation (e.g., based on the corpus, there is $X\%$ probability that two concepts are related);
- In *ontology mapping/alignment/matching* tasks (Euzenat and Shvaiko 2007), most of the relations found between distinct ontologies are probabilistic (e.g., two concepts in two different ontologies are the same with $X\%$ probability);
- Ontologies are being used to express *user context* when using a certain application (e.g., (Kersten and Murphy 2006)). For example, an application that recommends actions to the user can use ontologies to assert what are the most interesting concepts to the current context. This is usually done by assigning weights or probabilities to the ontology concepts and relations (e.g., in a word processor, when the user is writing about dogs, the concept *Dog* is $X\%$ more important than the concept *Cat*).

In most of the previous cases, the uncertainty is represented as a probability. However, Markov logic networks receive weights and not probabilities as its parameterization. In fact, these two concepts are very different. While a probability has a comprehensible meaning (a value between 0 and 1 representing our belief in a certain statement), weights are real values that only have a meaning when

compared to other weights (a statement is more plausible than another if it has a higher weight). So, to use those probability-annotated ontologies, probabilities must be transformed in weights.

In Markov logic, weights have a direct correspondence with probabilities if they are interpreted as log odds

$$w_i = \log \frac{p_i}{1 - p_i}, \quad 4.1$$

where p_i is the probability of the formula F_i , and w_i is its corresponding weight. However, if F_i shares variables with other formulas, as typically is the case, this correspondence cease to hold, since the weight of F_i is influenced not only by its probability, but also by the other formulas that share the same variable. In this case, the probabilities of all formulas collectively determine all the weights. One way of solving this situation is to treat formulas probabilities as empirical frequencies and learn the maximum likelihood weights using the algorithms previously demonstrated (Section 2.5.3). However, this solution can dramatically increase the complexity of the reasoning, since formulas' weights must be learned before the reasoning, a process that is computationally demanding. More recently, an extension to Markov logic (Jain, Kirchlechner, and Beetz 2007) has been proposed that somehow tries to tackle this problem by the specification of *a priori independent* attributes. Since this problem is very relevant to the success of our work, we will also try to explore specific solutions to our domain.

4.1.2. Probabilistic Reasoning with Weight Learning

In the last section, we took the principle that ontologies were somehow annotated with some kind of uncertainty information. However, these situations only occur in restricted domains, mainly those where these ontologies were built automatically by machines. In fact, it is studied (Tversky and Kahneman 1974) that humans are not good at either producing or perceiving concepts like probability. And even if humans were capable of doing that, creating and maintaining large annotated ontologies can be a cumbersome and difficult task, invalidating all the gains that could arise from the annotation. So, it is important to develop mechanisms to learn this uncertainty automatically. This can be useful not only to help users when creating probabilistic ontologies, but also to gain access to the vast number of non-annotated ontologies already available.

As previously seen (Section 2.5.3), in Markov logic, formulas' weights can be learned generatively or discriminatively through example data. This example data usually is composed by individuals of the domain and their relations. In the Semantic Web, the concept of *individual* already exists, and therefore can be used to learn the formulas weights. However, in our preliminary studies, we found that many ontologies (approximately 75%) do not have individuals, and even if they have, their number is reduced.

Number of ontologies	Number of ontologies with individuals	Number of ontologies with more individuals than formulas
216	54 (25%)	15 (7%)

Table 12. Preliminary study on ontology individuals.

So, other ways of gather evidence data must be developed. The most promising techniques are those used in the field of *ontology population* (Cimiano 2006). In this field, it is studied techniques to instantiate existing ontologies, mainly through the analysis of textual corpus. By using previously trained classifiers or general syntactic rules, we can extract information about ontology individuals and their relations. The main question in these techniques is what corpus use to learn that information: documents about the domain of the ontology, or global repositories, like web search engines. We plan to explore both approaches in our work.

4.2. System

To demonstrate the feasibility of our approach, we will develop a system that provides the capabilities referred in the previous section. This section describes the features and functionalities, requirements, and architecture of the system.

4.2.1. Features and Functionalities

The main features and functionalities of the system are:

- **Load ontologies with or without uncertainty annotations.** The system must read annotated ontologies (with weights or probabilities) and ontologies without annotations.
- **Selection of ontology axioms.** The system must provide means to select subsets of the ontology to perform the reasoning.
- **Weight learning.** The system must provide mechanism to learn the weights automatically.
- **Parameterization of algorithms.** The system must allow the change of the parameters of the main algorithms.
- **Programmatic and visual access.** The system must provide programmatic access and a visual interface.

4.2.2. Requirements

In this section, we define the system's requirements. These are divided in *functional requirements*, describing the system from the user's perspective, and *non-functional requirements*, describing required properties and constraints of the system.

The functional requirements of the system are:

- **Probabilistic reasoning using Markov logic in OWL ontologies.** Given a valid OWL ontology, the system must be able to perform probabilistic reasoning on it.
- **Intuitive result representation.** The reasoning results must be presented in an intuitive and understandable way.

The non-functional requirements, related to the usability, performance, and supportability of the system, are:

- **Simple access and installation.** Novice users must be allowed to install and operate with the system with little or no training. However, the system must be also prepared to deal with the needs of more experienced users.
- **High performance.** Since probabilistic reasoning is a very hard task in general, the system must be optimized to our specific domain. The parameters of performance definition of the system should be also visible to the user.
- **Easy modification and expansion.** The system shall allow users to easily modify or expand the current system to their concrete needs.

4.2.3. Architecture

In this section, we present the system's architecture and describe its main components. There are two main components in the system: the *System Core*, responsible for the reasoning, and the *Interface Layer*, responsible for the interaction with the user.

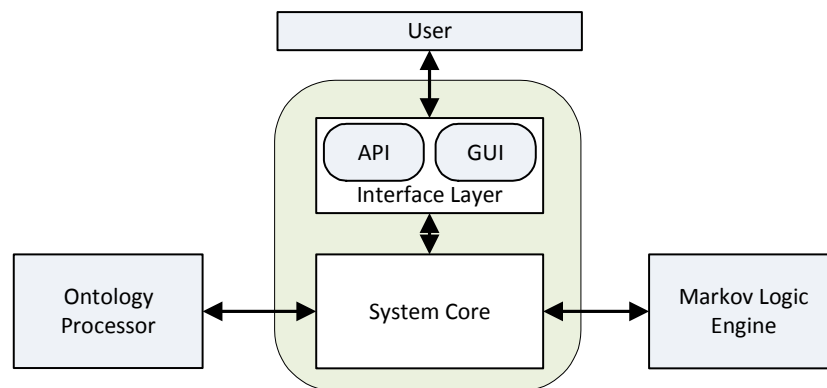


Figure 12. System architecture.

Next, these two main components of our system are explained.

System Core

The *System Core* is the main component of the system. It interacts with two main external components: the *Ontology Processor*, responsible for the reading, processing, and writing of OWL ontologies; and the *Markov Logic Engine*, responsible

for the reasoning and learning process. The selection of these two components will be made taking in account the tools specified in Section 3.2. Whenever it is possible, efforts will be made to allow the system to work independently of the components (i.e., if one component is substituted by a similar one, the system will have the same behavior).

The general function of the system is defined in the following flow chart.

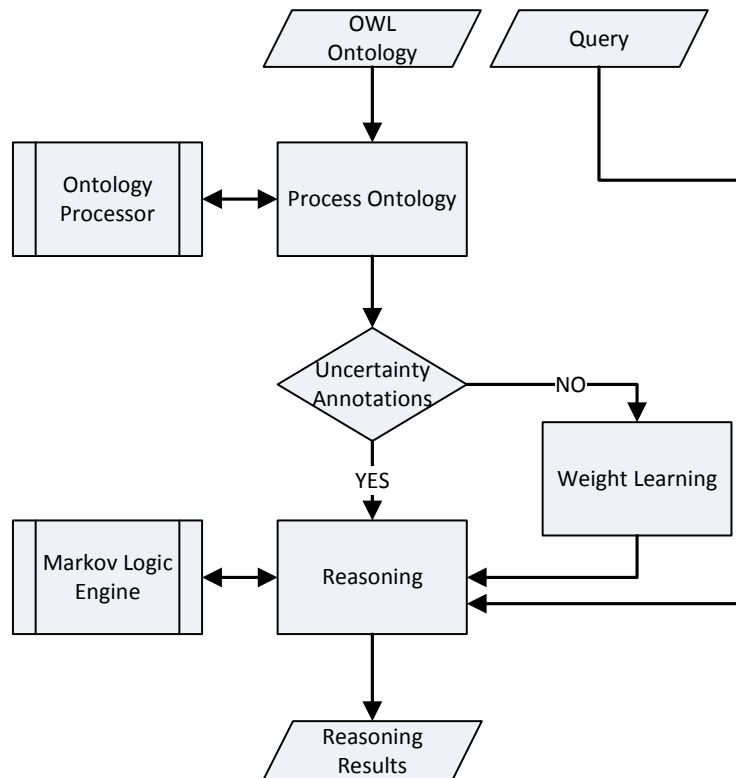


Figure 13. Flow chart representing the behavior of the System Core.

In the *Process Ontology* process, a valid OWL ontology is transformed in an efficient computational representation using an *Ontology Processor*. In this phase, the ontology is also transformed in first-order logic. The *Weight Learning* process is responsible for applying the techniques defined in Section 2.5.3. In the *Reasoning* process, the system will have all the information needed for reasoning: a set of weighted formulas (i.e., a Markov logic network) and a query. Using a *Markov Logic Engine*, the reasoning results are returned to the interface layer.

Interface Layer

The interface layer is responsible for the interaction with the user. There are two ways of communicating with the system:

- **Application Programming Interface (API)**, providing a programmatic access to the system. This allows the incorporation of the system in other applications.
- **Graphical User Interface (GUI)**, providing a visual access to the system. This allows the system to be intuitively used by the user.

Efforts will be made to provide both interfaces with the most number of similar features. However, given the specific properties of this work and comparing to similar systems, the API will be the most used interface to the system, and therefore could have more capabilities than the GUI.

4.3. Evaluation

In this section, we describe the main methods to evaluate our approach. Based on the features of the system, three main evaluation methods were identified:

- **Comparison with other Semantic Web probabilistic reasoners.** This evaluation allows comparing not only the reasoning results, but also the computational efficiency of our system. In Section 2.6.2, several other approaches to probabilistic reasoning in the Semantic Web have been demonstrated. However, in most of these approaches, there are no available systems that can be used to compare with our system. The only systems that are freely available are Pronto³⁸ (Klinov 2008) and ContraBovemRufum³⁹ (Nath and R. Moller 2008), both based on the probabilistic Description Logic $\mathcal{P}\text{-}\mathcal{SHOIN}(\mathcal{D})$.
- **Classification improvements.** In this evaluation, we try to explore the benefits of learning the uncertainty of an OWL ontology. For this purpose, the results of classification (i.e., predict the class of an individual given its attributes) of our system are compared against a non-probabilistic reasoner (e.g., Pellet⁴⁰ (Sirin et al. 2007)). These classification results can be measured with techniques used in the field of Pattern Recognition (Marques de Sá 2001), like measuring the *accuracy*, *specificity*, and *sensibility* of both approaches using *holdout* and *partition* methods.
- **Domain-specific tests.** Our system can be applied in many domains relevant to the Semantic Web, like *ontology mapping* (Euzenat and Shvaiko 2007) and *learning* (Maedche 2002), *social network analysis* (Mika 2007), and *ontology individual clustering* (Grimnes, Edwards, and Preece 2008). Our system can be evaluated in comparison to the state of the art systems developed in those areas.

³⁸ <http://pellet.owldl.com/pronto>

³⁹ <http://www.sts.tu-harburg.de/~t.naeth/#Software>

⁴⁰ <http://clarkparsia.com/pellet/>

5. Conclusions

To realize the Semantic Web vision of a world where knowledge is the most important element, mechanisms must be developed to represent and reason about the uncertainty that this knowledge could arise. In this thesis, we will explore the use of Markov logic, a unifying representation of logic and probability, to learn and reason about uncertainty in the Semantic Web. We think that our approach is of most importance to the field of the Semantic Web by a set of reasons:

- We apply the ideas of a new field, statistical relational learning, which has been developing mechanisms to learn and reason in large uncertain relational domains. These characteristics are the same of the Semantic Web;
- Unlike other approaches, we plan to study not only reasoning about uncertainty in the Semantic Web, but also how we can learn that uncertainty from various sources, like ontology individuals and textual corpus;
- Unlike other approaches, our approach is based on undirected probabilistic models, allowing cyclic knowledge;
- Our system can be used in many crucial domains for the Semantic Web, like ontology mapping and learning, social networks analysis, and ontology individual clustering.

In fact, since Markov logic is a so broad approach, we think that our approach of using Markov logic reasoning in the Semantic Web can be seen as an introductory step in providing a general Markov logic framework for the Semantic Web (Pedro Domingos, Lowd, et al. 2008), providing services like ontology learning, reasoning, mapping, refining, among others.

The thesis will comprise several deliverables to be concluded at different stages of the project. At the end of the project it is expected the deliverance of the following milestones:

- *Bibliographic Revision*, describing the most relevant concepts to this thesis (Chapter 2);
- *Technologies and Tools Study*, describing the tools and technologies that can be used in this thesis (Chapter 3);
- *Thesis Proposal Elaboration*, describing the proposed approach and the architecture of the system (Chapter 4);
- *First System Prototype*, a simple version of the system, composed mainly by the System Core (see Section 4.2.3);
- *Second System Prototype*, the final version of the system, with all the features described in Chapter 4;
- *Final Thesis Elaboration*, describing all the work done in this thesis.

The underlined deliverables were already concluded. The *First System Prototype* is currently being developed. The work plan defined for this thesis, with tasks and

respective schedule, is represented in Figure 14. The *Bibliographic Revision* and the *Technologies and Tools Study* suffered some delays compared to the schedule, but they are now completely finished.

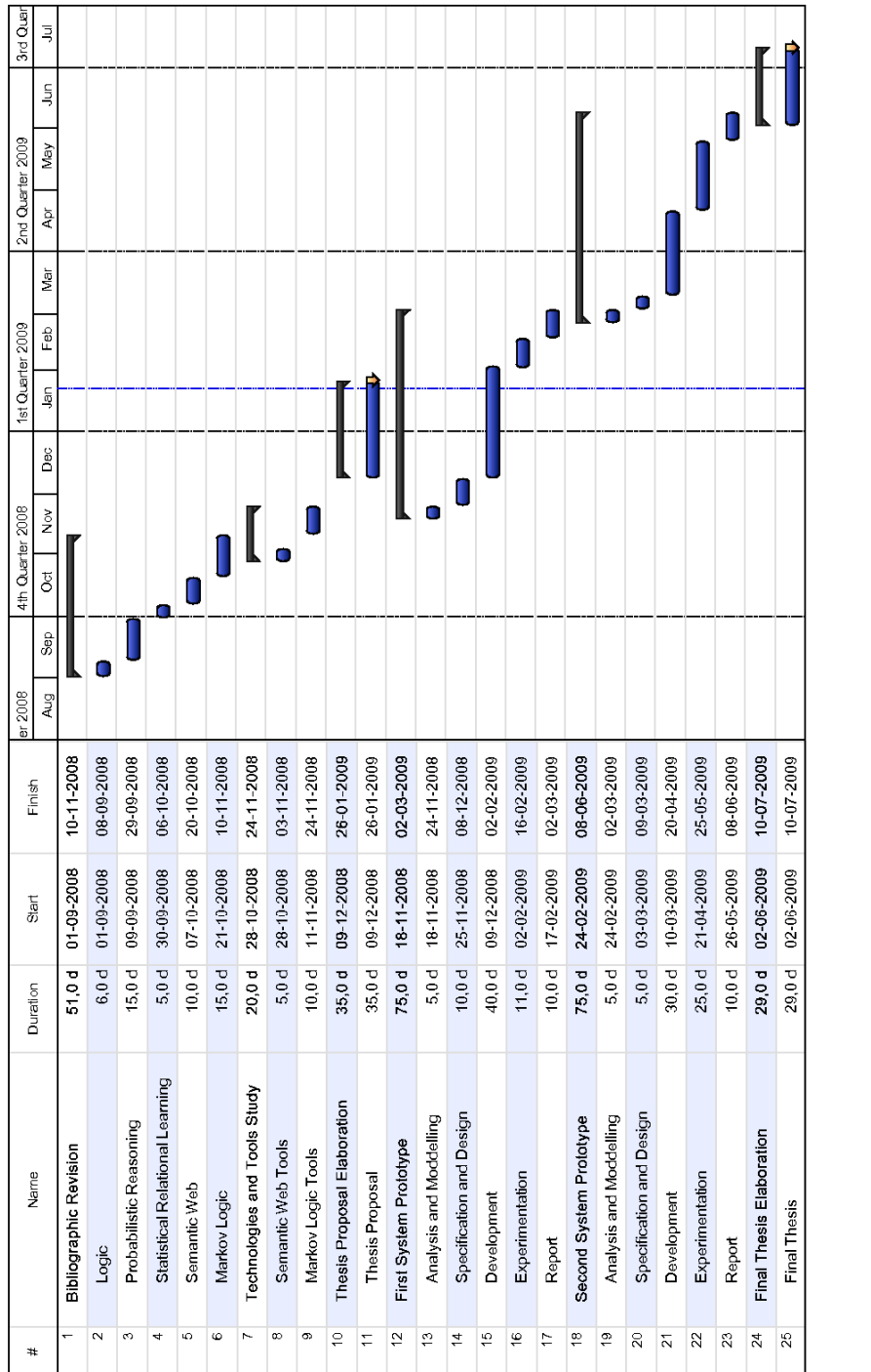


Figure 14. Project planning.

6. References

- Antoniou, Grigoris, and Frank van Harmelen. 2008. *A Semantic Web Primer, 2nd Edition*. 2nd ed. The MIT Press.
- Baader, Franz, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. 2nd ed. Cambridge University Press.
- Baeza-Yates, Ricardo, and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. 1st ed. Addison Wesley.
- Barwise, Jon, and John Etchemendy. 2002. *Language, Proof and Logic*. Center for the Study of Language and Inf.
- Berners-Lee, Tim. 2005. RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax. January. <http://tools.ietf.org/html/rfc3986>.
- Berners-Lee, T., J. Hendler, and O. Lassila. 2001. The Semantic Web. *Scientific American* 284, no. 5: 28-37.
- Berners-Lee, Tim. 1998. Semantic Web roadmap. <http://www.w3.org/DesignIssues/Semantic.html>.
- . 2000. Semantic Web. In *Invited Talk at XML 2000 Conference*. <http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html>.
- Besag, J. 1975. Statistical analysis of non-lattice data. *The Statistician* 24, no. 3: 179-195.
- Bishop, Christopher M. 2007. *Pattern Recognition and Machine Learning*. 1st ed. Springer.
- Bobillo, F., and U. Straccia. 2008. fuzzyDL: An expressive fuzzy description logic reasoner. In *Proceeding of the IEEE International Conference on Fuzzy Systems*, 923-930. doi:10.1109/FUZZY.2008.4630480.
- Breitman, K.K., M.A. Casanova, and W. Truszkowski. 2006. *Semantic Web: Concepts, Technologies and Applications*. 1st ed. Springer.
- Cardoso, J. 2007. The Semantic Web Vision: Where Are We? *IEEE Intelligent Systems* 22, no. 5: 84-88. doi:10.1109/MIS.2007.4338499.
- Cardoso, Jorge. 2007. *Semantic Web Services: Theory, Tools and Applications*. IGI Global.
- Cimiano, Philipp. 2006. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. 1st ed. Springer.
- Costa, P. C. G., and K. J. Laskey. 2005. PR-OWL: A Bayesian Ontology Language for the Semantic Web. In *Proceedings of the 1st Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2005)*, 6-10.

-
- Ding, Zhongli, Yun Peng, and Rong Pan. 2006. BayesOWL: Uncertainty Modeling in Semantic Web Ontologies. In *Soft Computing in Ontologies and Semantic Web*, 3-29. http://dx.doi.org/10.1007/3-540-33473-4_1.
- Dolby, J., A. Fokoue, A. Kalyanpur, A. Kershenbaum, E. Schonberg, K. Srinivas, and L. Ma. 2007. Scalable Semantic Retrieval through Summarization and Refinement. In *Proceedings of the National Conference on Artificial Intelligence*, 22:299. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- Domingos, Pedro, Stanley Kok, Daniel Lowd, Hoifung Poon, Matthew Richardson, and Parag Singla. 2008. Markov Logic. In *Probabilistic Inductive Logic Programming*, 92-117. http://dx.doi.org/10.1007/978-3-540-78652-8_4.
- Domingos, Pedro, Daniel Lowd, Stanley Kok, Hoifung Poon, Matthew Richardson, and Parag Singla. 2008. Just Add Weights: Markov Logic for the Semantic Web. In *Uncertainty Reasoning for the Semantic Web I*, 1-25. http://dx.doi.org/10.1007/978-3-540-89765-1_1.
- Euzenat, Jérôme, and Pavel Shvaiko. 2007. *Ontology Matching*. 1st ed. Springer.
- Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.
- Fukushige, Y. 2005. Representing Probabilistic Relations in RDF. In *Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*.
- Gennari, J. H., M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu. 2003. The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies* 58, no. 1: 89-123.
- Getoor, Lise, and Ben Taskar. 2007. *Introduction to Statistical Relational Learning*. The MIT Press.
- Getoor, U., N. Friedman, D. Roller, A. Pfeffer, and B. Taskar. 2007. Probabilistic Relational Models. In *Introduction to Statistical Relational Learning*, 129-174. The MIT Press.
- Giugno, Rosalba, and Thomas Lukasiewicz. 2002. P-SHOQ(D): A Probabilistic Extension of SHOQ(D) for Probabilistic Ontologies in the Semantic Web. In *Proceedings of the European Conference on Logics in Artificial Intelligence*, 86-97. Springer-Verlag. <http://portal.acm.org/citation.cfm?id=646333.756800>.
- Grau, B. C., I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. 2008. OWL 2: The next step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web*.
- Grimnes, Gunnar, Peter Edwards, and Alun Preece. 2008. Instance Based Clustering of Semantic Web Resources. In *The Semantic Web: Research and Applications*, 303-317. http://dx.doi.org/10.1007/978-3-540-68234-9_24.

-
- Gu, T., H. K. Pung, D. Zhang, and G. Kotsis. 2004. A Bayesian approach for dealing with uncertain contexts. In *The Proceeding of the Second International Conference on Pervasive Computing*.
- Haarslev, V., and R. Möller. 2003. Racer: A Core Inference Engine for the Semantic Web. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools*, 27-36.
- Heckerman, D., C. Meek, and D. Koller. 2007. Probabilistic Entity-Relationship Models, PRMs, and Plate Models. In *Introduction to Statistical Relational Learning*, 201-238. The MIT Press.
- Hendler, James. 2001. Agents and the Semantic Web. *IEEE Intelligent Systems* 16, no. 2: 30-37.
- Henrik, Nottelmann, and Fuhr Norbert. 2006. Adding Probabilities and Rules to Owl Lite Subsets Based on Probabilistic Datalog. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 14, no. 1: 17-42.
- Holi, Markus, and Eero Hyvönen. 2006. Modeling Uncertainty in Semantic Web Taxonomies. In *Soft Computing in Ontologies and Semantic Web*, 31-46. http://dx.doi.org/10.1007/3-540-33473-4_2.
- Horridge, M., S. Bechhofer, and O. Noppens. 2007. Igniting the OWL 1.1 Touch Paper: The OWL API. *Proc. of the Third International OWL:Experiences and Directions Workshop (OWLED- 2007)* 258.
- Jain, Dominik, Bernhard Kirchlechner, and Michael Beetz. 2007. Extending Markov Logic to Model Probability Distributions in Relational Domains. In *KI 2007: Advances in Artificial Intelligence*, 129-143. http://dx.doi.org/10.1007/978-3-540-74565-5_12.
- Jordan, Michael I. 2004. Graphical Models. *Statistical Science*. Special Issue on Bayesian Statistics.
- Jurafsky, Daniel, and James H. Martin. 2008. *Speech and Language Processing (2nd Edition)*. 2nd ed. Prentice Hall.
- Kersten, M., and G. C. Murphy. 2006. Using task context to improve programmer productivity. In *Proceedings of the 13th ACM SIGSOFT 14th International Symposium on Foundations of Software Engineering*, 1-11. ACM Press New York, NY, USA.
- Klinov, Pavel. 2008. Pronto: A Non-monotonic Probabilistic Description Logic Reasoner. In *The Semantic Web: Research and Applications*, 822-826. http://dx.doi.org/10.1007/978-3-540-68234-9_66.
- Klinov, Pavel, and Bijan Parsia. 2008. Probabilistic Modeling and OWL: A User Oriented Introduction to P-SHIQ(D). In *Proc. of the Fifth OWL: Experiences and Directions Workshop 2008 (OWLED'08)*. October 26.
- Klir, George J., and Bo Yuan. 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. 1st ed. Prentice Hall PTR.

-
- Kok, S., and P. Domingos. 2005. Learning the structure of Markov logic networks. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 22:441-448.
- Kok, S., P. Singla, M. Richardson, and P. Domingos. 2007. *The Alchemy system for statistical relational AI*. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. <http://alchemy.cs.washington.edu>.
- Laskey, K. B., and P. C. G. Costa. 2005. Of Klingons and Starships: Bayesian Logic for the 23rd Century. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence*. AUA Press.
- Levesque, H. J., and R. J. Brachman. 1985. A fundamental tradeoff in knowledge representation and reasoning. *Readings in Knowledge Representation* 1.
- Lindley, Dennis V. 2006. *Understanding Uncertainty*. 1st ed. Wiley-Interscience.
- Lukasiewicz, T. 2008. Expressive probabilistic description logics. *Artificial Intelligence* 172, no. 6-7: 852-883.
- Lukasiewicz, Thomas, and Umberto Straccia. 2008. Managing Uncertainty and Vagueness in Description Logics for the Semantic Web. *Web Semantics Sci Serv Agents World Wide Web*.
- Maedche, Alexander. 2002. *Ontology Learning for the Semantic Web*. 1st ed. Springer.
- Marques de Sá, J.P. . 2001. *Pattern Recognition: Concepts, Methods and Applications*. 1st ed. Springer.
- Mihalkova, L., and R. J. Mooney. 2007. Bottom-up learning of Markov logic network structure. In *Proceedings of the 24th International Conference on Machine Learning*, 625-632. ACM Press New York, NY, USA.
- Mika, Peter. 2007. *Social Networks and the Semantic Web*. 1st ed. Springer.
- Milch, B., B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Rolobov. 2007. BLOG: Probabilistic Models with Unknown Objects. In *Introduction to Statistical Relational Learning*, 373-398. The MIT Press.
- Milch, B., and S. Russell. 2006. General-purpose MCMC inference over relational structures. In *Proc. 22nd Conference on Uncertainty in Artificial Intelligence*, 349-358.
- Milch, B., L. S. Zettlemoyer, K. Kersting, M. Haimes, and L. P. Kaelbling. 2008. Lifted Probabilistic Inference with Counting Formulas. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 1062-1068.
- Moller, Ralf, and Volker Haarslev. 2008. Tableau-based Reasoning. In *Handbook of Ontologies*. 2nd ed. Springer.
- Motik, Boris, and Ulrike Sattler. 2006. A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In *Logic for Programming, Artificial*

-
- Intelligence, and Reasoning*, 227-241.
http://dx.doi.org/10.1007/11916277_16.
- Motik, Boris, Rob Shearer, and Ian Horrocks. 2007. Optimized Reasoning in Description Logics Using Hypertableaux. In *Automated Deduction – CADE-21*, 67-83. http://dx.doi.org/10.1007/978-3-540-73595-3_6.
- Muggleton, S., and N. Pahlavi. 2007. Stochastic Logic Programs: A Tutorial. In *Introduction to Statistical Relational Learning*, 323-338. The MIT Press.
- Nath, T. H., and R. Moller. 2008. ContraBovemRufum: A System for Probabilistic Lexicographic Entailment. *Proceedings of the 21st International Workshop on Description Logics (DL2008)*.
- Neville, J., and D. Jensen. 2007. Relational Dependency Networks. *The Journal of Machine Learning Research* 8: 653-692.
- Nottelmann, Henrik, and Umberto Straccia. 2006. A probabilistic, logic-based framework for automated Web directory alignment. *Soft computing in ontologies and the semantic Web. Studies in fuzziness and soft computing*: 47--77. doi:10.1.1.60.6575.
- Pan, J. Z., G. Stamou, G. Stoilos, and E. Thomas. 2007. Expressive Querying over Fuzzy DL-Lite Ontologies. In *Proceedings of the International Workshop on Description Logics (DL 2007)*.
- Pan, Rong, Zhongli Ding, Yang Yu, and Yun Peng. 2005. A Bayesian Network Approach to Ontology Mapping. *Proceedings of the International Semantic Web Conference 2005*: 563--577. doi:10.1.1.93.5004.
- Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. 1st ed. Morgan Kaufmann.
- Poon, H., and P. Domingos. 2006. Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In *Proceedings of the National Conference on Artificial Intelligence*, 21:458-463. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- Predoiu, Livia, and Heiner Stuckenschmidt. 2007. A probabilistic Framework for Information Integration and Retrieval on the Semantic Web. In *Proc. of 3rd International Workshop on Database Interoperability (InterDB) in conjunction with the VLDB conference*, 23-28. Vienna, Austria.
- Rersting, R., and L. De Raedt. 2007. Bayesian Logic Programming: Theory and Tool. In *Introduction to Statistical Relational Learning*, 291-322. The MIT Press.
- Riazanov, A. 2002. The design and implementation of VAMPIRE. *AI Communications* 15, no. 2: 91-110.
- Richardson, M. 2004. Learning and Inference in Collective Knowledge Bases. PhD Thesis, University of Washington.
- Richardson, M., and P. Domingos. 2006. Markov logic networks. *Machine Learning* 62, no. 1: 107-136.

-
- Richardson, Matthew, Rakesh Agrawal, and Pedro Domingos. 2003. Trust Management for the Semantic Web. In *The SemanticWeb - ISWC 2003*, 351-368. <http://www.springerlink.com/content/dm0n8r10rl4u430q>.
- Riedel, S. 2008. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*.
- Roller, D., N. Friedman, L. Getoor, and B. Taskar. 2007. Graphical Models in a Nutshell. In *Introduction to Statistical Relational Learning*, 13-55. The MIT Press.
- Russell, Stuart, and Peter Norvig. 2002. *Artificial Intelligence: A Modern Approach (2nd Edition)*. 2nd ed. Prentice Hall.
- Sanchez, Elie. 2006. *Fuzzy Logic and the Semantic Web*. 1st ed. Elsevier Science.
- Sauermann, Leo , and Richard Cyganiak. 2008. Cool URIs for the Semantic Web. March 31. <http://www.w3.org/TR/cooluris/>.
- Selman, B., H. A. Kautz, and B. Cohen. 1993. Local search strategies for satisfiability testing. *Proceedings of the Second DIMACS Challenge on Cliques, Coloring, and Satisfiability*.
- Shapiro, S. 2001. Classical logic II—Higher-order logic. *The Blackwell Guide to Philosophical Logic*.
- Silva, Paulo Pinheiro da, Deborah L. McGuinness, and Richard Fikes. 2006. A proof markup language for Semantic Web services. *Information Systems* 31, no. 4-5 (July): 381-395. doi:10.1016/j.is.2005.02.003.
- Singla, P., and P. Domingos. 2005. Discriminative Training of Markov Logic Networks. In *Proceedings of the National Conference on Artificial Intelligence*, 20:868. Menlo Park, CA; Cambridge, MA; London; AAI Press; MIT Press; 1999.
- . 2006. Memory-Efficient Inference in Relational Domains. In *Proceedings of the National Conference on Artificial Intelligence*, 21:488. Menlo Park, CA; Cambridge, MA; London; AAI Press; MIT Press; 1999.
- Sirin, E., B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web* 5, no. 2: 51-53.
- Sterling, Leon, and Ehud Shapiro. 1994. *The Art of Prolog, Second Edition: Advanced Programming Techniques*. 2nd ed. The MIT Press.
- Stocker, Markus, and Michael Smith. 2008. Owlgrs: A Scalable OWL Reasoner. In *Proceedings of the Fifth International Workshop OWL: Experiences and Directions (OWLED 2008)* . Karlsruhe, Germany.
- Stoilos, G., G. Stamou, V. Tzouvaras, J. Z. Pan, and I. Horrocks. 2005a. Fuzzy OWL: Uncertainty and the Semantic Web. In *Proceedings of the International Workshop on OWL: Experiences and Directions*.

-
- . 2005b. The fuzzy description logic f-SHIN. In *Proc. of the International Workshop on Uncertainty Reasoning for the Semantic Web*, 67–76.
- Straccia, U. 2006a. A Fuzzy Description Logic for the Semantic Web. *Fuzzy Logic and the Semantic Web*: 73-90.
- . 2006b. Answering vague queries in fuzzy DL-Lite. In *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-06)*, 2238–2245.
- Taskar, B., P. Abbeel, M. F. Wong, and D. Roller. 2007. Relational Markov Networks. In *Introduction to Statistical Relational Learning*, 175-200. The MIT Press.
- The Unicode Consortium. 2006. *Unicode Standard, Version 5.0, The*. 5th ed. Addison-Wesley Professional.
- Tsarkov, Dmitry, and Ian Horrocks. 2006. FaCT++ Description Logic Reasoner: System Description. In *Automated Reasoning*, 292-297. http://dx.doi.org/10.1007/11814771_26.
- Tsarkov, Dmitry, Alexandre Riazanov, Sean Bechhofer, and Ian Horrocks. 2004. Using Vampire to Reason with OWL. In *The Semantic Web – ISWC 2004*, 471-485. <http://www.springerlink.com/content/6huhpyqdt7yw4pv2>.
- Tversky, Amos, and Daniel Kahneman. 1974. Judgment under Uncertainty: Heuristics and Biases. *Science* 185, no. 4157: 1124-1131. doi:10.1126/science.185.4157.1124.
- Udrea, O., V. S. Subrahmanian, and Z. Majkic. 2006. Probabilistic RDF. In *IEEE International Conference on Information Reuse and Integration*, 172-177.
- Wahlster, Wolfgang, Henry Lieberman, and James Hendler. 2002. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. The MIT Press.
- Wei, W., J. Erenrich, and B. Selman. 2004. Towards Efficient Sampling: Exploiting Random Walk Strategies. In *Proceedings of the National Conference on Artificial Intelligence*, 670-676. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- Yang, Y., and J. Calmet. 2005. OntoBayes: An Ontology-Driven Uncertainty Model. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, 1:457-463. IEEE Computer Society Washington, DC, USA.