NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

POSTGRADUATE STUDIES PROGRAM

MASTER THESIS

Optimization and inference under fuzzy numerical constraints

Vassileios-Marios Anastassiou

Supervisors:
Panagiotis Stamatopoulos, Associate Professor NKUA
Stasinos Konstantopoulos, Post-Doctoral Researcher NCSR Demokritos

ATHENS

OCTOBER 2014

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βελτιστοποίηση και συμπερασμός παρουσία ασαφών αριθμητικών περιορισμών

Βασίλειος-Μάριος Αναστασίου

Επιβλέποντες:
Παναγιώτης Σταματόπουλος,Επίκουρος Καθηγητής ΕΚΠΑ
Στασινός Κωνσταντόπουλος, Μετα-Διδακτορικός Ερευνητής ΕΚΕΦΕ Δημόκριτος

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2014

MASTER THESIS

Optimization and inference under fuzzy numerical constraints

Vassileios-Marios Anastassiou

RN: M1217

SUPERVISORS:

Panagiotis Stamatopoulos, Assistant Professor NKUA

Stasinos Konstantopoulos, Researcher NCSR Demokritos

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βελτιστοποίηση και συμπερασμός παρουσία ασαφών αριθμητικών περιορισμών

Βασίλειος-Μάριος Αναστασίου

Α.Μ: Μ1217

Επιβλέποντες:

Παναγιώτης Σταματόπουλος, Καθηγητής ΕΚΠΑ
Στασινός Κωνσταντόπουλος, Ερευνητής ΕΚΕΦΕ Δημόκριτος

ΠΕΡΙΛΗΨΗ

Εκτεταμένη έρευνα έχει γίνει στους τομείς της Ικανοποίησης Περιορισμών με διακριτά (ακέραια) ή πραγματικά πεδία τιμών. Αυτή η έρευνα έχει οδηγήσει σε πολλαπλές σημασιολογικές περιγραφές, πλατφόρμες και συστήματα για την περιγραφή σχετικών προβλημάτων με επαρκείς βελτιστοποιήσεις. Παρά ταύτα, λόγω της ασαφούς φύσης πραγματικών προβλημάτων ή ελλιπούς μας γνώσης για αυτά, η σαφής μοντελοποίηση ενός προβλήματος ικανοποί-ησης περιορισμών δεν είναι πάντα ένα εύκολο ζήτημα ή ακόμα και η καλύτερη προσέγγιση. Επιπλέον, το πρόβλημα της μοντελοποίησης και επίλυσης ελλιπούς γνώσης είναι ακόμη δυσκολότερο. Επιπροσθέτως, πρακτικές απαιτήσεις μοντελοποίησης και μέθοδοι βελτιστο-ποίησης του χρόνου αναζήτησης απαιτούν συνήθως ειδικές πληροφορίες για το πεδίο εφαρμογής, καθιστώντας τη δημιουργία ενός γενικότερου πλαισίου βελτιστοποίησης ένα ιδιαίτερα δύσκολο πρόβλημα.

Στα πλαίσια αυτής της εργασίας θα μελετήσουμε το πρόβλημα της μοντελοποίησης και αξιοποίησης σαφών, ελλιπών ή ασαφών περιορισμών, καθώς και πιθανές στρατηγικές βελτιστοποίησης. Καθώς τα παραδοσιακά προβλήματα ικανοποίησης περιορισμών λειτουργούν βάσει συγκεκριμένων και προκαθορισμένων κανόνων και σχέσεων, παρουσιάζει ενδιαφέρον η διερεύ-νηση στρατηγικών και βελτιστοποιήσεων που θα επιτρέπουν το συμπερασμό νέων ή/και αποδοτικότερων περιορισμών. Τέτοιοι επιπρόσθετοι κανόνες θα μπορούσαν να βελτιώσουν τη διαδικασία αναζήτησης μέσω της εφαρμογής αυστηρότερων περιορισμών και περιορισμού του χώρου αναζήτησης ή να προσφέρουν χρήσιμες πληροφορίες στον αναλυτή για τη φύση του προβλήματος που μοντελοποιεί.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ : βελτιστοποίηση, περιορισμός, αριθμητικός, συμπερασμός, ασάφεια, OWL

Abstract

Extensive research has been done in the areas of Constraint Satisfaction with discrete/integer and real domain ranges. Multiple platforms and systems to deal with these kinds of domains have been developed and appropriately optimized. Nevertheless, due to the incomplete and possibly vague nature of real-life problems, modeling a crisp and adequately strict satisfaction problem may not always be easy or even appropriate. The problem of modeling incomplete knowledge or solving an incomplete/relaxed representation of a problem is a much harder issue to tackle. Additionally, practical modeling requirements and search optimizations require specific domain knowledge in order to be implemented, making the creation of a more generic optimization framework an even harder problem.

In this thesis, we will study the problem of modeling and utilizing incomplete and fuzzy constraints, as well as possible optimization strategies. As constraint satisfaction problems usually contain hard-coded constraints based on specific problem and domain knowledge, we will investigate whether strategies and generic heuristics exist for inferring new constraint rules. Additional rules could optimize the search process by implementing stricter constraints and thus pruning the search space or even provide useful insight to the researcher concerning the nature of the investigated problem.

SUBJECT AREA: Artificial Intelligence

KEYWORDS: optimization, constraint, numerical, inference, fuzziness, OWL

Contents

List of Figures

List of Theorems

1. Introduction

In a world of complex systems, incomplete knowledge and changing requirements, the modeling and reasoning on numerical problems is often an intimidating task, requiring multiple modeling iterations and experimentations until a consistent model is created. Such a monolithic process may also lead to re-usability issues, e.g when a platform or formalism is not generalized or easily extensible to be used in multiple projects, or to wasted time overhead when requirements change or the investigator notices that the core of the problem was incompletely modeled. A generic and interactive method of modeling such problems may provide researchers with the appropriate tools to create a re-usable platform to model a problem, manage constraints and guide both the design and the automated reasoning/search processes.

A well researched area of Artificial intelligence is CSPs which are mathematical problems described as a set of variables with specific domains which are limited via constraints, modeling physical systems and the relations between them. A CSP solver is software which uses this information to infer new constraints in order to search for instantiations that satisfy the constrains. Due to the complexity of the systems they model, CSPs with finite or even real variables may be cumbersome or even inadequate to represent specific problems which contain incomplete or vague information. There are various methods of describing incomplete knowledge and applying reasoning on it, one of them being the well-researched area of Fuzzy Logic. Extensive work has been done on Fuzzy CSP (f-CSP) formalisms which give us the tools for integrating fuzzy variables in CSPs and the insight on utilizing fuzzy operators properly. Nevertheless, the process of modeling a CSP is a difficult one, possibly requiring multiple design iterations before landing on a successful description (otherwise, search performance suffers due to a vast search space or worse

the problem is so much constrained that there are no possible solutions). This means, of course, that the modeling of f-CSPs an even harder problem! Proper visual representations of such problems, interoperable representations and newer formalisms that retain adequate expressivity while yielding better performance would give CSP investigators an even better tool to implement and share their work. The developing application area of the Semantic Web is a promising candidate, providing a software stack that permits research on various levels (from low-level logic semantics to high-level software engineering practices).

In Chapter Two, we will study classical CSPs as well as the representation and solution process of dynamic and vague problems. We will investigate methods of modeling flexible, dynamic and fuzzy descriptions, as well as various methods of optimizing numerical constraints in CSPs. In Chapter three, we will investigate fuzzy logic and its semantics, we will describe Description Logics (DLs) and finally we will see how they are combined with fuzzy logic in the literature. Fuzzy Description Logics are a fairly recent formalism for describing fuzzy knowledge via ontologies, providing a common ground for describing and reasoning fuzzy knowledge on the Web, and will be described in chapter three. Finally, in Chapter four, we will investigate how constraints are modeled via ontologies and what reasoning capabilities are provided by such systems and will discuss the requirements of an ontological framework for describing constraint satisfaction problems with fuzzy relations and numerical constraints.

2. Constraint Satisfaction Problems

In this chapter, we will describe Constraint Satisfaction Problems (CSPs) and showcase their usefulness by careful investigation of their definition and features in sections 2.1 and 2.2. In section 2.3, we will provide valuable insight on the breadth of their applicability and their extensions, as well as their limits.

2.1 CSP definitions

A large number of problems in Artificial Intelligence and other fields of Computer Science can be represented, modeled and solved as CSPs. The formal definition of a CSP is the following:

Definition 2.1.1 (Constraint Satisfaction Problem)**.** A CSP is a tuple $\langle X, D, C \rangle$, where:

- $X = \{x_1, \ldots, x_n\}$ is a set of variables,
- $D = \{D_1, \ldots, D_n\}$ is a set of the respective domains of values such that $\forall x_i \in X$ there is a domain D_i, and
- $C = \{C_1, \ldots, C_m\}$ is a set of constraints such that the scope of each constraint is a subset of X

In other words, CSPs are the description of a problem containing finite sets of variables (each with its relevant domain of values) and a set of constraints each applied on some subset of variables.

Definition 2.1.2 (CSP Variable Domain and Constraints)**.** Each variable is defined by its domain, that is the set of all possible/considered values that can be assigned to the variable. These can vary according to the problem type, the arithmetic system used and the platform used for investigating a CSP, although finite discrete or real domains are the typical choice. A constraint defines a mathematical relation on and between CSP variables which must (hard) or should (soft) be satisfied.

Definition 2.1.3 (CSP Instantiation, Solution, Labeling and Consistency)**.** A solution to a CSP is a complete instantiation of the variables in X satisfying all the constraints in C. If a CSP has at least one solution, it is described as satisfiable or consistent, otherwise we say that it is inconsistent.

Solving a CSP requires searching the search space defined by $\langle X, D, C \rangle$ in order to find an appropriate assignment between variables X and their domain D, which at the same time satisfy constrains defined in C. In other words, variable literals are grounded during the process of labeling, checking for (full or partial) satisfiability of the constraint store in order to produce a satisfying assignment. At the same time, an additional goal may be not only to find any/all valid assignments for a specific problem, but to find an optimal solution based on specific cost criteria.

A specific instantiation is called locally consistent if it satisfies all the constraints that affect its instantiated variables. Thus, a solution is a locally consistent full instantiation of the CSP variables. In a globally consistent CSP, there is a locally consistent partial instantiation that can be extended to satisfy all constraints and thus lead to one or more solutions [Dechter, 1992]. An inconsistent CSP may be the result of overly strict constraints, in which case investigation is required in order to determine whether some constraints can be *relaxed* in order to widen the search space to include possible solutions [Epstein and Yun, 2010].

2.2 Backtracking and Constraint Propagation

The main methods of searching for a solution in a CSP is Back-Tracking via tree search and constraint propagation. Simple backtracking search guarantees that a solution will be found if one exists, but it suffers from redundant value assignments and possible thrashing (i.e. continuously exploring the same or similar search subtrees), duplicating effort and degrading performance [Lecoutre et al., 2006]. Constraint propagation disregards domain values that will definitely not satisfy the constraints but is usually insufficient for providing the actual solution by itself. The typical approach is a hybrid one, that is enhancing the typical Tree search method with a constraint propagation algorithm. In the search tree of the backtracking algorithm, whenever a node is visited, a constraint propagation algorithm is performed to attain a desired level of consistency by removing inconsistent values from the domains of the as yet uninstantiated variables. During the process of constraint propagation on a node, if the domain of any variable becomes empty then the node is pruned, as the current instance becomes inconsistent. An additional family of optimization methods for backtracking are called Look-back techniques, which avoid assignments that have lead to dead-ends during the search. These can be implemented as Back-Jumping (trying to continue the search from a previous ancestor rather than the parent node, that is to

the most recent variable that eliminated values from the dead-end variable) and No-Good learning (where additional constraints are introduced in order to detect similar dead-ends earlier in future searches) [Jussien and Boizumault, 2002].

Constraint propagation is a form of reasoning in which, from a subset of the constraints and the domains, more restrictive constraints or more restrictive domains are inferred. The inferences are justified by local consistency properties that characterize necessary conditions on values or set of values to belong to a solution. Arc consistency is currently the most important local consistency property in practice and has received the most attention in the literature. The importance of constraint propagation is that it can greatly simplify a constraint problem and so improve the efficiency of a search for a solution. Constraint propagation methods are online and parallelizable, meaning that one may interrupt propagation on a subtree and examine the intermediate instance if required, and it is also possible to incrementally add or remove constraints in order to check their effect on constraint propagation. There are two main approaches to performing constraint propagation: the rules iteration approach and the algorithmic approach. The former constitutes of reduction rules that specify conditions under which domain reductions can be performed for a constraint; the latter uses generic or special purpose algorithms specifically designed for constraints based on problem type [Kumar, 1992; Dib et al., 2010]

Using these heuristics, potential thrashing can be effectively reduced and the search process is directed towards possible ”bottlenecks” in the search space, where search is likely to fail, efficiently pruning the search space. Methods such as Forward Checking (FC) and Maintaining Arc-Consistency (MAC) are implementations satisfying said properties. FC achieves partial arc consistency while MAC guarantees full arc consistency by constraining the search space before proceeding in each search step. Additionally, multiple iterations of the main MAC algorithm have been implemented, as well as a number of heuristics enhancing the original functionality, providing multiple methods of inference that satisfy all possible types of constraint consistency [Bessiere et al., 1995]. Haralick and Elliott [1980] provide an in-depth investigation of said heuristics and strategies (and more such as Back-checking and Back-marking), demonstrating that efficient and consistent labeling is achievable without introducing significant overhead complexity. Selectively searching subtrees based on available label choices (effectively by sorting domains based on available domain values) also enhances search performance, as demonstrated by Bacchus and Run [1995].

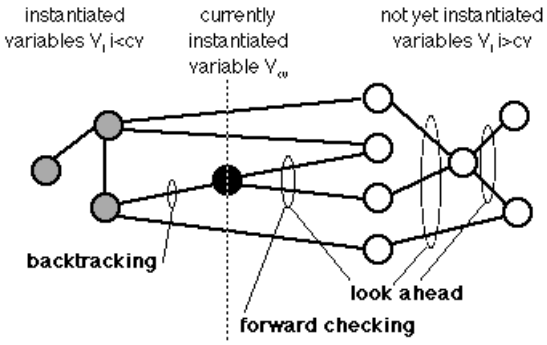

Figure 2.1: Comparison of the three basic CSP heuristic methods (source: Guide to Constraint Programming, Roman Barták, 1998)

2.2.1 Generalized frameworks for constraint planning

Bartak [1997] described constraint hierarchies which were introduced for describing over-constrained systems by labeling constraints as either required (called **hard**) or useful (called **soft**), and also with hierarchical strengths or preferences, allowing an arbitrary number of different strengths for the whole set. This enables the declarative specification of both hard and soft constraints which, in combination with existing efficient satisfaction algorithms, are the strengths of constraint hierarchies. Most of the modern satisfaction algorithms can be grouped into two separate categories: *refining* [Hosobe et al., 1996] and *local propagation* algorithms. The author introduces a generalized framework for constraint planning aiming to solve constraint hierarchies by leveraging generalized local propagation techniques.

A comparator is an irreflexive and transitive relation over partially-ordered valuations of variables that is consistent with the constraint hierarchy. One can easily deduce by the definition of constraint hierarchies that the stronger constraints are more influential as far a solution is concerned. However, comparators can parameterize the constraint hierarchy schemes thus allowing the comparison of different solutions and choosing the best among them. These may be locally-better (comparable on 1 level), regionally-better (comparable in multiple levels where there are no compatible locally-better comparators) or globally-better. All those comparators require error functions indicating the satisfaction degree of a specific valuation. Predicate-comparators use the trivial error function, returning 0 for satisfied constraints and 1 for violated constraints.

When investigating the limits of current satisfaction algorithms, one notices that the generality of refining method sacrifices the incremental updates of the solution but in return it allows using arbitrary comparators for all kinds of constraints. On the contrary, local prop-

agation is incremental and hence execution time is reduced but its major disadvantage is the preceding planning phase, required to decide constraint satisfaction order. Additionally, certain limits are imposed by the features of local propagation solvers. For instance, during the planning phase, weaker constraints may be disabled unfairly or conflicts cannot be resolved. The executing phase is linear, meaning that constraints expressed as systems of linear equations cannot be solved. Local propagation handles only equality (functional) constraints and it can identify just a single solution because functional constraints are satisfied uniquely.

The author, bearing in mind the aforementioned limits of current algorithms, proposes a new framework that encapsulates local propagation and refining approaches in order to combine their advantages, namely efficiency and generalization. The basic idea of this framework is the organization of the constraint hierarchy into constraint cells containing constraints of equal strength and partially ordering them in a constraint network. This network represents relationships between constraints and can be traversed by an executing local propagation algorithm based on the partial order of cells. It is shown that the same constraint hierarchy may lead to multiple constraint networks. Although the ideas of propagated valuations and order of constraint cells refer to local propagation solvers, solving all constraints of a cell in one step corresponds to the refining approach. The focus of the described method is on constraint network construction which is called planning stage. The authors set an important rule for solving a constraint hierarchy in the general case, namely that ”the satisfaction of a stronger constraint is strictly preferred to the satisfaction of an arbitrary number of weaker constraints”. This intuitive rule means that the satisfaction of a constraint should not cause the violation of an equally or more strongly preferred constraint after propagation in order to guarantee a monotonic behavior during search. More advanced methods are available that can be used to describe and solve over-constrained problems and thus be used as generalized CSP resolution frameworks [1].

2.3 Extending CSP with non-static variants

The classic, static model of CSPs we described earlier is a successfully-applied formalism, sufficient for defining many constraint-related problems, although by its nature is inadequate to describe dynamic situations or incomplete knowledge. Thus, various proposals

[1] `http://ktiml.mff.cuni.cz/~{}bartak/constraints/general.html`

try to enhance the CSP definition with constraints more appropriate for those kinds of problems. Dechter and Dechter [1988] described and demonstrated that constraint networks, generated from relevant CSP formalisms, provide ”an attractive paradigm for modeling dynamically changing environments”. Together with interactive CSP solving techniques [Madsen, 2003] and explanatory CSP solution methods [Jussien, 2001], these tools can greatly enhance the ability of an investigator to effectively describe a solvable constraint satisfaction problem and explore alternative descriptions.

In the following sections we will describe similar but also unique ways to deal with non-static constraint requirements. Even though we will only focus on Fuzzy CSP (f-CSP) in subsection 3.2.2, the interested reader may study more about non-static CSP formalisms via Miguel [2001], where he described how newly developed techniques for solving Dynamic CSPs (DCSPs) and Flexible CSPs (FCSPs) can be integrated together and provide powerful investigation and modeling capabilities.

2.3.1 Flexible CSPs and Preferences

FCSPs relax the requirement of having hard constraints (i.e. all constraints must be satisfied, otherwise there is no solution), partially relaxing some constraints and providing a partial solution that does not satisfy all of them. This is similar to preferences in preference-based planning. An early extension following these requirements is MAX-CSP where, given inconsistent or over-constrained CSPs, the problem is transformed to a maximization problem with the goal of finding a partial solution satisfying most constraints (or, inversely, a minimization goal of violating the least constraints possible). There a number of constraints are allowed to be violated, and the quality of a solution is measured by the number of satisfied constraints [Kask, 2000]. Weighted CSP (WCSP) can be thought as a more specific type of MAX-CSP where each violation of a constraint is weighted according to a predefined preference. Another extensively studied formalism, f-CSP (described in depth in section 2.3.3), models constraints as fuzzy relations in which the satisfaction of a constraint is a continuous function of its variables' values, going from fully satisfied to fully violated. While f-CSPs associate a level of preference with each tuple in each constraint, WCSPs assign a specific cost to search constraint, which facilitates the modeling of the CSP as a specific optimization problem, where the goal (and thus the aggregation function) is explicitly defined to minimize the total cost of the required solution. The cost function is defined by summing up the costs of all constraints, that is the cost of the chosen tuple for each

constraint accordingly. Thus, the goal is to find a set of tuples that minimize the sum of costs of their sub-tuples (one for each constraint). Two WCSPs are defined as equivalent when they contain the same variable set and also the same cost distribution over it. It logically follows that a problem with no solution has no locally consistent labeling with non-infinite cost. Typical consistency definitions (such as arc, node, pair etc) are extended to support weighted constraints, and various extensions have been proposed (such as Full Directional Arc consistency, Existential Directional Arc consistency, Virtual Arc consistency and Optimal Soft Arc consistency [Levy et al., 2007]). Even though the concept of weight is used to indicate the relative importance level between entities, the concept of priority can similarly be used to indicate the importance level of a constraint among some constraints. There are various definitions concerning WCSPs which deal with the way weight and constraint consistencies are treated, focusing on cost transfer operations and the ordering of constraint requirements [Brown, 2003], which we will not discuss here.

2.3.2 Dynamic CSPs

DCSPs are useful when there are no set requirements for a specific problem and is possible that its formulation changes during the search process. For example, a specific variable instantiation represents a real-world condition where requirements or facilities change, typically because the set of constraints to consider evolves because of the environment. These changes usually only affect a specific part of the instantiation and thus previous partial solutions may be re-used or slightly tuned without restarting the search process [Verfaillie and Schiex, 1994]. DCSPs are viewed as a sequence of static CSPs, where each CSP in the sequence is a transformation of the previous one in which variables and constraints can be relaxed (removed constraints) or restricted further (added constraints). Thus, information or partial solutions introduced in initial formulations of the problem can be used to refine following ones. Various methods of information transfer have been proposed, such as Local Repair (where each CSP is re-calculated from the partial solution of a previous one and any inconsistent constraints are repaired via local search methods), Oracles (where partial solutions are used as heuristics to guide the resolution of a completely new search for the current CSP) and Constraint recording (where new constraints are defined in each stage of the search to represent the learning of inconsistent group of decisions and carried over to newer formulations). Another way to deal with dynamic constraints is by tagging certain variables of the constraint network as assumption variables which are initially assigned

default values but may dynamically change their values during search without requiring backtracking to the root of the search tree, a process called contradiction resolution.

Mouhoub and Sukpan [2012] propose an interesting extension where they propose a framework for managing DCSPs with multiple types of preferences (unary, binary, composite and conditional). Their method provides enhanced modeling flexibility, as it can convert a given CSP into a constraint network where conditional constraints and composite variables control the addition of metadata to the constraint network in a dynamic manner during the resolution process. That way, preferences are associated to variable and constraint values as well as composite variables, in order to favor some solutions of the constraint problem but not sacrifice any local consistency. Composite variables are variables whose values are existing CSP variables, effectively representing disjunction possibilities and guiding the search accordingly. Conditional preferences define a preference function to dynamically determine whether a constraint is to be considered active or inactive, further guiding search pruning. Wallace et al. [2009] have experimented on possible heuristics and search methods in DCSPs and have shown that performance may vary significantly under certain circumstances even in the presence of subtle feature changes. Problem alterations have significant effect on which search subtree is possible to fail first, but mainly on promising alternatives. This fact means that adaptive contention-based search strategies (i.e. strategies which are evaluating sub-solution similarities and try to order them by how promising they are for leading to a solution) may not yield predictable search improvements. On the other hand, sub-solution caching is a robust and important step of the search process because it can effectively reduce variability when the problem changes, leading to more predictable performance.

2.3.3 Fuzzy CSPs

Similarly to CSPs, f-CSPs are defined by $\langle X, D, C, P\rangle$ tuples, where Constraints C are described as fuzzy relations between variables in X. The additional P vector contains flexibility (or priority) degrees to each constraint which indicate the minimum degree threshold that each constraint must be satisfied by. In other words, a constraint restricts the values these variables can simultaneously take and thus represents the possible combination of values of a consistent instantiation. Thus, a membership degree of 0 means a constraint is fully violated and a degree of 1 represents a crisp constraint that is fully satisfied. The feasible solutions in an f-CSP are complete instantiations satisfying all fuzzy constraints

simultaneously. The search may also focus on Min-optimal solutions, which are solutions where the satisfaction degree of the least satisfied constraint is optimal. Intuitively, solving a f-CSP involves finding a compound label of all variables such that the constraints involved are satisfied, to some extent, with the compound label.

Based on the goals specified by the fuzzy constraints, a measure is required to make a decision on which instantiation better satisfies the relevant constrains simultaneously [Bellman and Zadeh, 1970]. Assuming finite variable domains, the fuzzy sets are defined by a finite linearly ordered valuation set. Thus, in a partial instantiation, the membership degree of a specific domain variable represents the maximum satisfaction level of the relevant constraints, while in a full instantiation we know the specific degree by which the constraint is satisfied. The level of local consistency is the satisfaction level of the least satisfied constraint. Thus, a fully violated constraint means that the instantiation is locally inconsistent. It logically follows that a local consistency is always less than or equal to its global consistency, because additional variable instantiations can only possibly maintain the consistency level if not reduce it.

The set of feasible solutions is a fuzzy set defined over the set of potential solutions which satisfy the priority degrees of the constraints. Thus, the best solution is the one with the largest consistency degree over the set of possible solutions. As with classical CSPs, additions of constrains is expected to shrink the set of solutions but it may also rule out any previously best solutions, if the new constraint is satisfied with a too low a degree but search may continue as long as the instantiation is not completely inconsistent. A new constraint may be determined as redundant (when the set of best solutions remains the same), compatible with P (the new optimal solution set is a subset of the previous one), partially inconsistent with P (constraints are implicitly relaxed due to lower membership degrees) or totally incompatible (where the set of optimal solutions is empty). Hence, the set of best solutions does not decrease monotonically when new constraints are added. This non-monotonic behaviour makes the search process of f-CSPs more complex than classical CSPs. Constraint relaxation is now more intricate as it is intertwined with the flexibility of the constraints.

By using the f-CSP formalism, one can model a CSP with priorities on constraints or a CSP with preferences among the constraint tuples. Furthermore, as both the satisfaction scale and the priority scale are essentially ordinal scales, one can move between those to express the same ordering among the potential solutions. This duality is based on the duality between possibility and necessity already presented in Possibility Theory [Zadeh,

1978]. Moreover, the FCSP approach bypasses empirical relaxation techniques which are needed when a set of constraints is globally unfeasible. Constraint relaxation often happens to be more expensive, difficult to formulate, and leading to suboptimal solutions [Guo et al., 2006]. On the contrary, the FCSP approach is able to provide solutions to partially inconsistent problems because a solution (i.e. the instantiation with the maximal satisfaction degree) can be achieved as long as the specified problem is partially consistent [Dubois et al., 1996]. Thus, fuzzy constraints are a useful means of investigating problem modeling by guiding search towards solutions achieving higher satisfaction degrees.

Dubois et al. [1996] provide an extensive overview of the specifics of f-CSP modeling and search, as well as the handling of uncertainty in CSP with Possibility Theory. Thomas Schiex extended the previous work on f-CSPs by introducing Possibilistic Logic concepts in order to integrate soft constraints, preferences and modeling of incomplete knowledge [Schiex, 1992]. Chang and Mackworth [2005] have described the method of Constraint-Based Inference (CBI), where they propose a weaker condition of applying generalized arc consistency enforcing techniques, achieving local consistency in the presence of incomplete or soft constraints. Guesgen and Philpott [1995] provides an extensive review of heuristic methods for f-CSPs. Miguel [2003] examines methods for solving fuzzy dynamic CSPs and how classical planning can be extended via fuzzy sets to enable flexible goals and preferences to be placed on the use of planning operators for use with CSP-based planners.

We will investigate the concept of f-CSPs deeper in subsection 3.2.2.

2.3.4 CSPs with numerical or interval constrains

Even though a constraint or optimization problem is determined by the domain ranges of its variables, another way of modeling is by reasoning with interval values. Interval arithmetic is an arithmetic defined on sets of intervals, rather than sets of real numbers. Even though such formalisms focus on numerical analysis applications such as the automatic control of computational error (formally defined as interval analysis and introduced by Moore [1966]), they prove to be an intuitive formalism for describing temporal or spatial relations. Spatio-temporal reasoning can be achieved with various methods and CSPs are shown to be an adequate formalism for practical applications [Akplogan and Dury, 2011].

Davis [1987] provided one of the first instances of generalizing traditional constraint propagation of discrete values to intervals. His work was one of the earliest examples of

constraint propagation via interval labels, providing an extensive review of inference techniques and discussions on the nature and capabilities of using intervals to represent constraint values, where he also established that constraints of bounded differences can be efficiently solved by the Waltz algorithm [2]. He based his work on the concept of ”quantity knowledge bases”, Knowledge Bases (KBs) of quantitative facts which focus on physical reasoning and contain experimental or inferred relations between physical, real-valued quantities. One can easily understand that such instances provide inference possibilities on numerical relations that may readily lead to combinatorial explosion. These kinds of problems are also described as Numeric CSPs. After discussing the various inference methods and their capabilities and introducing the concepts of sign and interval labeling (e.g. applying metadata on data structures representing the expected sign or range of a variable or complex expression), the author investigates practical ways of expressing qualitative relations, important points in relation refinement, and the complexity of such operations. He also suggests that probabilistic metaheuristics, although intuitively useful in pruning the search space, are unreliable and experimentally shown to be slow and limiting legitimate alternative solutions. Even though theoretical results on the complexity of the required operations and expected performance were discouraging, the authors demonstrate that practical problems and their applications, as well as specific design considerations and locality considerations, prove to be adequate for practical applications. This position is alleviated in future work by Faltings [1994] and shown to work on a more general case. Lhomme [1993] has shown that consistency techniques used for CSP can be adapted to numeric CSPs over finite or continuous domains and enhanced with incremental labeling. Even though searching for a globally consistent instance of valid domain ranges is an NP-hard problem, the authors suggest a method of aiming for strong arc consistency by using convex, closed domain bounds and exploiting bound relations (e.g. B-consistency, an arc consistency restricted to the bounds of domains). Numerical integration methods such as the Simplex algorithm, the Grobner bases method, indexicals, interval splitting and approximation methods such as Newton-Raphson or Runge-Kutta are extensively used for solving such systems, although a complete and general CSP search strategy is still a difficult problem and also unavoidably prone to slow convergence [Bordeaux et al., 2007]. Carlson and Gupta [1997] demonstrated a hybrid language and system that could utilize many such methods when solving numerical CSPs.

[2] https://www.cs.cmu.edu/afs/cs/academic/class/46927-f97/slides/Lec5/sld021.htm

Hyvonen [1989] devised a generalized constraint propagation scheme based on interval instead of conventional arithmetic. He introduced the concept of tolerance propagation, where multiple alternative values (based on the interval bounds of a domain) are simultaneously propagated in the constraint network. That way, the search process can execute more effective constraint propagation without having to iterate through all possible discrete domain values or perform interval reasoning over real domains. Additionally, problem instances with non-exact input-output (e.g. where the output of a component may affect the input of another) which would otherwise prove difficult or intractable to model in a CSP, may be well-defined via appropriate dynamic variables and flexible value intervals.

Allen [1983] introduced an interval calculus for temporal reasoning, acknowledging that temporal knowledge in many applications is relative or imprecise and therefore representations based on time instants and dating were inadequate and providing the basis for many extensions and further research. He employed a temporal representation which takes the notion of a temporal interval as primitive and then represents the relationships between temporal intervals in a hierarchical manner. Allen's interval-based temporal logic scheme is particularly appealing for its simplicity and for its ease of implementation with constraint propagation algorithms. This interval algebra calculus for temporal reasoning defines thirteen possible relations between time intervals (namely *overlaps, starts, finishes, during, meets, takes place before*, their inverses, and *is equal to*) and provides a composition (transitivity) table that can be used as a basis for reasoning about temporal descriptions of events. This approach is useful for modeling processes and process interaction involving data bases of interactive systems, where the concept of "current time" is dynamic and techniques such as exact dating are not possible. Allen's Interval Algebra can be used for the description of both temporal intervals and spatial configurations. For the latter use, the relations are interpreted as describing the relative position of spatial objects. This also works for three-dimensional objects by listing the relation for each coordinate separately.

Using Allen's calculus, given temporal facts can be formalized and then employed for automatic reasoning. Relations between intervals are formalized as sets of base relations and relationships between intervals can be maintained in a network where the nodes represent individual intervals. Each network arc is labeled to indicate the possible relationship between the two intervals represented by its nodes. In cases where there is uncertainty about the relationship, all possible cases are entered on the arc (due to the fact that the thirteen possible relationships are mutually exclusive, there is no ambiguity in this notation). Allen's algorithm continues to operate as long as it is producing new further constrained

relationships between intervals, generating a complete constraint network of temporal relations.

In order to reduce the space requirements of the representation without greatly affecting the inferential power of the mechanism, reference intervals are introduced. Formally, a reference interval is simply another interval in the system, but it is endowed with a special property; it is used to group together clusters of intervals for which the temporal constraints between each pair of intervals in the cluster is fully computed. Such a cluster is related to the rest of the intervals in the system only indirectly via the reference interval. Reference intervals capture the temporal hierarchy implicit in many domains and enable the precise control of the amount of deduction performed automatically by the system (e.g, when planning the activities of a robot, one must model the effects of the robot's actions on the world to ensure that a plan will be effective).

2.3.5 Final thoughts

We have established the usefulness of CSP solvers and of various methods they implement in order to support varying constraints. Nevertheless, such systems typically require expert domain knowledge and experience in CSP modeling in order to promptly create fast and adequate representations. Integrating new technologies that would make the visual representation of CSP design easier and enhance interoperability is an interesting task which we will discuss in Chapter four, after we introduce relevant technologies and study Fuzzy Logic in-depth in Chapter three. Concerning interval arithmetic, an extensive overview not specific to CSPs is provided by Kearfott [1996]. For an extensive overview of CSP formalisms, as well as in-depth analysis of complete and incomplete specification solvers, we guide the interested reader to "Principles of Constraint Programming" by Apt [2003].

3. (Fuzzy) Description Logics

3.1 A brief history of fuzziness

The concept of non-classical logic using an additional indeterminate truth value was firstly introduced by Lukasiewicz [1930] in his work on trivalent logic. After 35 years, Fuzzy Logic (FL) was officially introduced as a concrete mathematical theory by Lofti A. Zadeh in his paper on "Fuzzy Sets" [Zadeh, 1965], and later extended with relevant algorithmic background [Zadeh, 1968]. Novak et al. [1999] write that fuzzy logic "[...] in narrow sense is a special many-valued logic which aims at providing formal background for the graded approach to vagueness". Due to the fact that the mathematics presented on the paper were unexplored, fuzzy logic met many objectors, which resulted in the slow development of applications, mainly in the west. On the other hand, the east community openly accepted this theory and started developing products based on fuzzy logic[1]. Until 1975, Zadeh continued his investigation into fuzzy set theory and fuzzy logic was revived in the US some years later, enhanced by other researchers and used in practical applications. Up until now fuzzy logic has been used on numerous applications that extend from hardware controllers and the development of products and sensors (collectively described as "fuzzy control systems"), to AI software techniques and implementation of programs dealing with vague information.

3.2 Fuzzy logic systems and fuzzy operators

Fuzzy logic is a form of non-classical logic. In contrast with classical propositional or predicate logic, the rule of excluded middle is rejected from its semantics Garrido [2010] and multiple truth values (not the traditional binary valuation of logical expressions) are used. Their arithmetic systems can be either discrete, such as three-valued logics with -1, 0 and 1 as truth values, or defined under a specific real interval such as [0,1]. Thus, we can describe vague information by defining (in addition to traditional crisp sets) the concept of fuzzy sets, where each propositional variable is mapped via a membership function into varying membership degrees.

[1] http://wing.comp.nus.edu.sg/pris/FuzzyLogic/HistoricalPerspectiveDetailed1.html

In the core of Fuzzy Logic and Fuzzy Sets is the membership function μ_A, used to describe degrees of truth (or simply degrees) which is the confidence of membership in a specific (fuzzy) set. The membership function acts as a generalization of the indicator (or characteristic) function in classical set theory. Degrees are described as a mapping to the real interval [0, 1], with 0 describing confident non-membership (false) and 1 describing confident membership (true). In that sense, a variable may be incompletely included to multiple sets, expressing varying degrees of truth depending on the membership functions used. Fuzzy Set/Algebra operators act as a generalization of classical logic operators and are based on the membership function. Fuzzy logics, together with many or infinite-valued logic, form the T-norm fuzzy logic family. Triangular norms (often noted as T in function notation and $\circledast$ in infix notation) act as a generalization of the conjunction operator in logics (or the intersection operation in lattice theory).

Definition 3.2.1 (Monoidal T-Norm-based propositional fuzzy logic)**.** MTL is an axiomatization of logic where conjunction is defined by a left continuous T-Norm, and implication is defined as the residuum of the T-Norm. Its models correspond to MTL-algebras that are prelinearly-ordered, commutative, bounded integral, residuated lattices.

The appropriate connectives for defining generalized intersection and union operations were a class of associative monotonic connectives known as triangular norms (T-Norms for short), together with their De Morgan dual triangular co-norms. These operations are at the basis of the semantics of a class of mathematical fuzzy logical systems that have been thoroughly studied [Klement and Navara, 1999; Gottwald, 2000; Mazeika et al., 2007; Noguera et al., 2010].

T-norms are a generalized extension for fuzzy logics of the typical two-valued classical logical conjunction and are used to construct the intersection of fuzzy sets. We know that the Boolean conjunction is both commutative and associative. The monotonicity property ensures that the conjunction degree of truth does not decrease if the conjuncts truth values increase. The requirement of 1 as an identity element corresponds to the interpretation of 1 as true and consequently 0 as false. Continuity in the framework of fuzzy conjunction, expresses the important idea that, very small changes in truth values of conjuncts should not macroscopically affect the truth value of their conjunction.

The connectives defined through the continuous T-Norm conjunctions (continuity with respect to the left argument is sufficient) are special. Accordingly, there are algebraic procedures relating them with implications, which have very interesting metalogical properties.

Any such implication is defined as residuum of a given T-Norm BaczyE•ski and Jayaram [2008].

Definition 3.2.2 (Fuzzy Logic Residuum)**.** Residuum is an FL operation which acts as the logical implication operator and is mathematically defined as $c \circledast a \leq b \iff c \leq (a \Rightarrow b), \forall a, b, c \in [0, 1]$

Hajek introduced the basic fuzzy propositional logic, BL-logic, as the logic of continuous T-Norms on interval [0,1]. The language of BL comprises the connectives of conjunction, implication and the constant of falsity. The semantics of BL is established by T-Norm functions and all other functions corresponding to the connectives are derived. A formula is a BL tautology if and only if under each valuation of propositional variables, compatible with the functions of connectives, takes the value 1.

Definition 3.2.3 (Basic propositional fuzzy logic)**.** BL is an extension of MTL logic where conjunction is defined by a continuous T-Norm, and implication is also defined as the residuum of the T-Norm. Its models correspond to BL-algebras.

In order to clarify the notion of T-Norms, let's look at the truth function of conjunction, on which the following constraints are imposed by T-norm fuzzy logics. Commutativity ensures that the order of fuzzy propositions, i.e. conjuncts in a conjunction (and disjuncts in a disjunction) is immaterial, despite the introduction of intermediary truth degrees. Associativity warrants that the grouping in an iterated conjunction (or disjunction) does not affect the truth-conditions, i.e. that the order of the conjunction operation is immaterial. Monotony evinces the important fact that augmenting the truth degree of a conjunct should not lessen the truth degree of the conjunction. Monotonicity in formal logics means that adding a formula to a theory never produces a reduction of its set of consequences. Intuitively, it indicates that acquiring a new piece of knowledge cannot reduce the fuzzy set of what is known. However, despite the fact that the most prevalent formal logics have a monotonic consequence relation, one must bear deep in mind, that a monotonic logic cannot handle various reasoning tasks, such as reasoning by default (consequences may be derived only because of lack of evidence of the contrary), abductive reasoning (consequences are only deduced as most likely explanations), some important approaches to reasoning about knowledge (the ignorance of a consequence must be retracted when the consequence becomes known), and likewise, belief revision (when new knowledge may contradict old beliefs.)

The assumption of Neutrality of 1, is in accord with regarding the truth degree 1 as full truth, without decreasing the truth value of the other conjunct. Combined with the previous conditions neutrality ensures that the truth degree 0 corresponds to full falsity and also that any conjunction with it is always fully false.

Continuity (the previous constraints relax the continuity requirement to either argument) expresses the crucial necessity that microscopic changes of the truth degrees of conjuncts should not result in a macroscopic change of the truth degree of their conjunction. Continuity, among other things, ensures the good behavior of (residual) implication derived from conjunction; to guarantee this good behavior, however, left-continuity (in either argument) of the function is sufficient. Therefore, only left-continuity is essential, which expresses the requirement that a microscopic decrease of the truth degree of a conjunct should not macroscopically decrease the truth degree of conjunction.

The aforementioned assumptions ensure that the truth function of conjunction is a left-continuous T-Norm. In various other logics further assumptions may be made about the behavior of conjunction (for example, Godel Logic requires its idempotent) or other connectives (for example, Involutive Monoidal T-Norm based Logic (IMTL) requires the involutiveness of negation).

Definition 3.2.4 (Interpretation of Many-Valued Logics)**.** Semantically, a many-valued interpretation I maps each basic proposition p_i into $[0, 1]$ and is then extended inductively to all propositions as follows:

- $I(a \wedge b) = I(a) \otimes I(b)$
- $I(a \vee b) = I(a) \oplus I(b)$
- $I(a \Rightarrow b) = I(a) \rhd I(b)$
- $I(\neg a) = \ominus I(a)$

where the four operators above are called combination functions. More specifically, these are the triangular norms (or T-Norms), triangular co-norms (T-Conorms, also called S-Norms), implication functions, and negation functions respectively, which extend the classical Boolean conjunction, disjunction, implication, and negation, accordingly, to the Many-Valued logics Lukasiewicz and Straccia [2008].

Definition 3.2.5 (Properties of FL systems)**.** Juxtapositions between the systems serving as a base for particular constructions directed the focus of research towards strong connectives whose corresponding truth functions are associative, commutative, non-decreasing and have 1 as its neutral (unit) element Behounek [2010]. In order for a fuzzy logic to be valid, it must satisfy the following basic mathematical properties which derive from monoidal and lattice algebras:

- Commutativity: $a \circledast b = b \circledast a$
- Monotonicity: $a \leq b \Rightarrow a \circledast z \leq b \circledast z, \forall a, b, c \in [0, 1]$
- Associativity: $a \circledast (b \circledast c) = (a \circledast b) \circledast c$
- Neutral/Identity element: $a \circledast 1 = a$

Additionally, residuated lattice logics (MTLs) should satisfy an additional property:

- (Left-)Continuity: $a \circledast b \leq c \iff a \leq b \to c$

Furthermore, all left-continuous T-Norms have a unique residuum, that is interpreted as a fuzzy version of the modus ponens rule of inference. The residuum of a left-continuous T-Norm is the weakest function that makes the fuzzy modus ponens valid, and indeed it is an indispensable truth function for implication in fuzzy logic. Left-continuity of the T-Norm is the necessary and sufficient condition for this relationship between a T-Norm conjunction and its residual implication to hold.

One may define truth functions of other propositional connectives via the T-Norm and its residuum, such as the residual negation or the bi-residual equivalence. In this way, a left-continuous T-Norm, its residuum, and the truth functions of additional propositional connectives determine the truth values of complex propositional formulas on the interval $[0, 1]$.

Tautologies with respect to any left-continuous T-Norm are the formulas that always evaluate to 1. The set of all tautologies is called the logic of the T-Norm, as these formulas typify the laws of fuzzy logic (determined by the T-Norm) which hold regardless of the truth degrees of atomic formulas. Some formulas are tautologies with respect to a larger class of left-continuous T-Norms. The set of such formulas is called the logic of the class.

A T-Norm is described as continuous, provided it is continuous (as a function) on the interval $[0, 1]$. (Similarly for left- and right-continuity.) It is also strict if it is continuous and strictly monotone.

Furthermore, a T-Norm $\circledast$ is called nilpotent if it is continuous and each x in the open interval (0,1) is its nilpotent element, that is, there is a natural number n where $\underbrace{x \circledast \ldots \circledast x}_{n} = 0$. Moreover, an Archimedean T-Norm $\circledast$ satisfies the Archimedean property, where $\forall y \in (0,1) \exists x \in (0,1) : \underbrace{x \circledast \ldots \circledast x}_{n} \leq y$.

The standard partial ordering of T-Norms is pointwise and, being functions, pointwise larger T-Norms are sometimes termed stronger as compared to the pointwise smaller ones. Nevertheless, in the semantics of fuzzy logic, the larger a T-Norm is, the weaker conjunction it represents, in terms of logical strength.

A T-Norm is continuous if and only if it is continuous in one variable. Analogous theorems hold for left- and right-continuity of a T-Norm. A continuous T-Norm is Archimedean if and only if 0 and 1 are its only idempotents. A continuous Archimedean T-Norm is strict if 0 is its only nilpotent element; otherwise it is nilpotent. By definition, moreover, a continuous Archimedean T-Norm T is nilpotent if and only if each $x < 1$ is a nilpotent element of T. Therefore with a continuous Archimedean T-Norm T, either all or none of the elements of $(0,1)$ are nilpotent.

If it is the case that all elements in $(0,1)$ are nilpotent, then the T-Norm is isomorphic to the Lukasiewicz T-Norm. If there are no nilpotent elements of T, the T-Norm is isomorphic to the Product T-Norm. In other words, all nilpotent T-Norms are isomorphic, the Lukasiewicz T-Norm being their prototypical representative; and all strict T-Norms are isomorphic, with the Product T-Norm being their prototypical example.

Finally, in the typical semantics of T-Norm based fuzzy logics, where conjunction is interpreted by a T-Norm, the residuum is a unique binary operation on $[0,1]$ and plays the role of implication, which is often termed:

Definition 3.2.6 (R-implication)**.** $x \Rightarrow y = max\{z | x \circledast z \leq y\}$

Residuated implications make sense and are well-defined only if and only if the generating T-Norm is left-continuous. Implication is of fundamental importance for fuzzy logic in the narrow sense. There is another straightforward, though logically less attractive possibility, in which implication may be defined from conjunction and negation (or disjunction and negation) using the corresponding tautology of classical logic, called S-implications Tick and Fodor [2005].

As far as T-conorms (also called S-norms) are concerned, they are dual to T-Norms under the order-reversing operation which assigns $1-x$ to x on the $[0,1]$ interval, an obvious

generalization of De Morgan's laws. T-conorms also satisfy Commutativity, Monotonicity, Associativity and have number 0 as their Identity element. The aforementioned conditions can be used for an equivalent axiomatic definition of T-Conorms independently of T-Norms. The main point about T-Conorms is that they are used to represent logical disjunction in fuzzy logic and union in fuzzy set theory. It is interesting to note that **any properties of T-Conorms can be obtained by dualizing the properties of T-Norms**, e.g. for any T-Conorm the number 1 is the annihilating element. Also, dually to T-Norms, all T-Conorms are bounded by the Maximum and the Drastic T-Conorm. **Maximum T-Conorm**, is dual to the Minimum T-Norm, and is the smallest of all T-Conorms. **Probabilistic sum** is dual to the Product t-norm and is the standard semantics for strong disjunction in extensions of product fuzzy logic over which it is defined, and also expresses the probability of the union of independent events in Probability theory. **Bounded sum** is dual to the Lukasiewicz t-norm and thus is the standard semantics for strong disjunction in Lukasiewicz fuzzy logic. Finally, **Einstein sum** is dual to one of the Hamacher T-Norms and, interestingly, is similar to the velocity-addition formula of special relativity and the law of addition of hyperbolic tangents. The interested reader may study these operators in depth in Gassert [2004][2].

We may visualize T-Norms as 3D plots, as surfaces in the unit cube or as contour plots showing the curves (or, more generally, the sets) where the function in question has constant (equidistant) values, and, occasionally, as diagonal sections. Since T-Norms are just functions from the unit square into the unit interval, their comparison is done in the usual way (e.g. pointwise).

	$x * y$	$x \Rightarrow y$ for $x \leq y$	$x \Rightarrow y$ for $x > y$	$\neg x$
Łukasiewicz	$\max(0, x+y-1)$	1	$1-x+y$	$1-x$
Gödel	$\min(x, y)$	1	y	$\neg 0 = 1$
product	$x \cdot y$	1	y/x	$\neg x = 0$ for $x > 0$

Figure 3.1: Comparison of popular T-Norms (source: plato.stanford.edu)

Any system of propositional logic determined by a T-Norm may be appreciated as a strengthening of BL. For instance, Lukasiewicz, Godel and Product logics result from BL by one addtional axiom schema:

[2]Also, a fine summary is available on: `http://www.nicodubois.com/bois5.2.htm` and `http://www.inside-r.org/packages/cran/sets/docs/.N`.

- Lukasiewicz: $\neg\neg a \Rightarrow a$
- Godel: $a \Rightarrow (a \wedge b)$
- Product: $\neg\neg a \Rightarrow ((a \Rightarrow (a \wedge b)) \Rightarrow (b \wedge \neg\neg b))$

Some salient examples of T-Norms are presented below:

3.2.1 Typical T-Norm systems

Definition 3.2.7 (Minimum T-Norm)**.** Minimum T-Norm (also termed as Godel T-Norm) is the standard semantics for conjunction in Godel fuzzy logic. It occurs in most T-Norm based fuzzy logics as presenting the typical semantics for weak conjunction. It is the pointwise largest T-Norm and also the only T-Norm where each element in [0,1] is an idempotent element.

It is defined as $A \circledast_{min} B = min\{A, B\}$, with residuum $A \Rightarrow_{min} B = \begin{cases} 1, if A \leq B \\ B, otherwise \end{cases}$

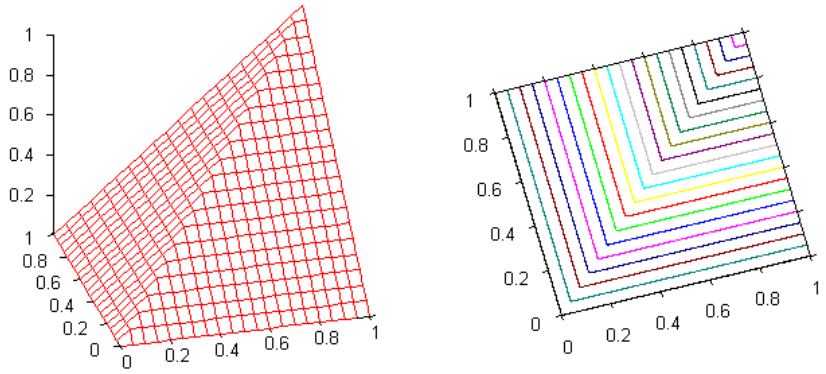

Figure 3.2: Graph of the Minimum T-Norm (3D and contours) (credits: Libor Behounek)

Definition 3.2.8 (Product T-Norm)**.** Product T-Norm (the ordinary product of real numbers). Besides other uses, the Product T-Norm is the standard for strong conjunction in product fuzzy logic. It is a strict, Archimedean T-Norm.

It is defined as $A \circledast_{\Pi} B = AB$, with residuum $A \Rightarrow_{\Pi} B = \begin{cases} 1, if A \leq B \\ B/A, otherwise \end{cases}$

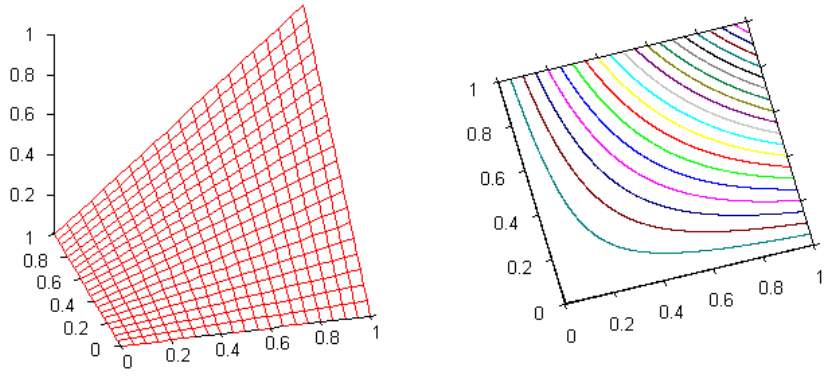

Figure 3.3: Graph of the Product T-Norm (3D and contours) (credits: Libor Behounek)

Definition 3.2.9 (Lukasiewicz T-Norm)**.** Lukasiewicz T-Norm is named after the fact that it is the standard T-Norm used for strong conjunction in Lukasiewicz fuzzy logic. Sometimes it is referred to as the Linear T-Norm. It is a nilpotent, Archimedean T-Norm, pointwise smaller than the Product T-Norm.

It is defined as $A \circledast_{L} B = max\{0, A+B-1\}$, with residuum $A \Rightarrow_{L} B = \begin{cases} 1, if A \leq B \\ 1 - A + B, otherwise \end{cases}$

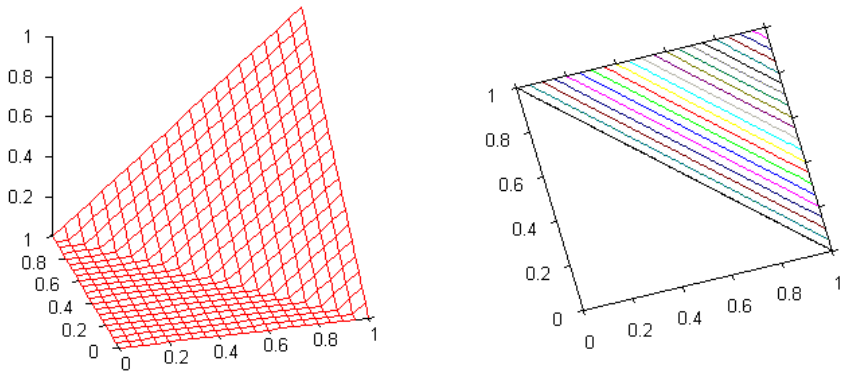

Figure 3.4: Graph of the Lukasiewicz T-Norm (3D and contours) (credits: Libor Behounek)

Definition 3.2.10 (Drastic T-Norm)**.** Drastic T-Norm is the pointwise smallest T-Norm (hence the emphatic name drastic.) It is a right-continuous Archimedean T-Norm.

It is defined as $A \circledast_D B = \begin{cases} B, if A = 1 \\ A, if B = 1 \\ 0, otherwise \end{cases}$

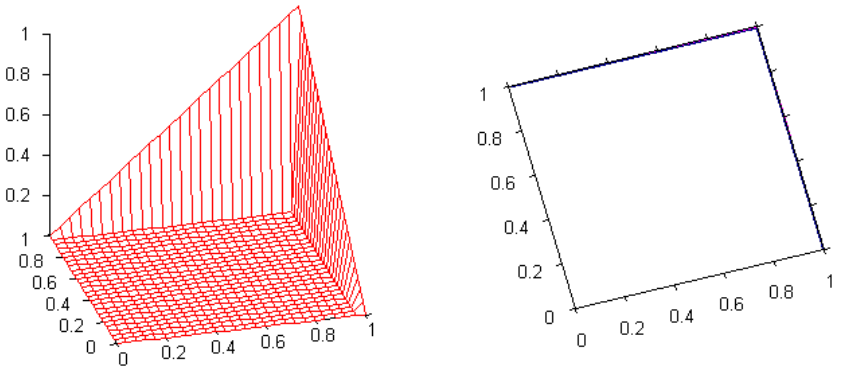

Figure 3.5: Graph of the Drastic T-Norm (3D and contours) (credits: Libor Behounek)

Definition 3.2.11 (Nilpotent T-Norm)**.** Nilpotent minimum is a standard illustration of a left-continuous T-Norm which is, not continuous. Notwithstanding its name, the nilpotent minimum is not a nilpotent T-Norm. An extensive overview of its importance and semantics is

presented by Fodor [2004].

It is defined as $A \circledast_{nM} B = \begin{cases} min\{A, B\}, if A + B > 1 \\ 0, otherwise \end{cases}$

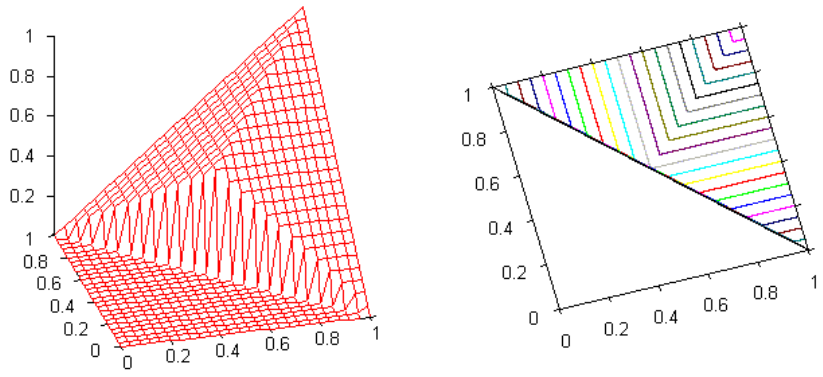

Figure 3.6: Graph of the Minimum T-Norm (3D and contours) (credits: Libor Behounek)

Definition 3.2.12 (Hamacher T-Norm)**.** Hamacher product is a strict, Archimedean T-Norm, and a principal example of the parametric classes of Hamacher T-Norms and Schweizer-Sklar T-Norms.

It is defined as $A \circledast_{D} B = \begin{cases} 0, if A = B = 0 \\ \dfrac{AB}{A + B - AB}, otherwise \end{cases}$

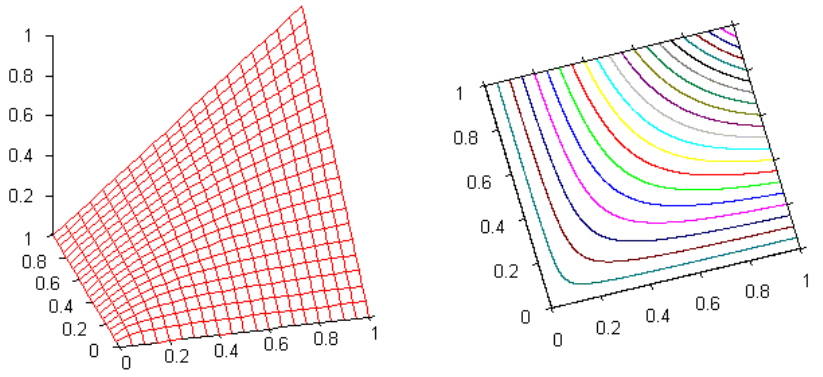

Figure 3.7: Graph of the Hamacher T-Norm (3D and contours) (credits: Libor Behounek)

3.2.2 Fuzzy CSPs and fuzzy operators

In section 2.3.3, we laid the foundations of f-CSPs, which we can now combine with a deeper understanding of fuzzy logic semantics.

Luo et al. [2003] extend the work of Dubois et al by providing deeper investigation of axioms concerning the local satisfaction degree of a prioritized constraint, as well as providing an axiomatic discussion about possibilistic aggregation, trying to distinguish the scale for priorities from the scale for constraint satisfaction degrees. Finally, they investigate the relationship between prioritized and weighted f-CSP schemes. Their results indicated that a Prioritized Fuzzy CSP (PFCSP) is isomorphic to a typical f-CSP and permit the adoption of existing solution techniques. Additionally, they state and prove a theorem which demonstrates that new, valid priority operators can be constructed from existing priority operators, guaranteeing that such a definition is reasonable according to the axiomatic framework of fuzzy logic. That way, even though many existing systems focus on the Gödel (or minimum) T-Norm, a system can easily be extended to use any valid T-Norm, enhancing the efficiency or applicability of typical constraint solving techniques such as arc-consistency, even though it still remains a hard problem.

Zadeh's initial proposal was to use a minimum T-Norm as the generic f-CSP global satisfaction degree formula. Bellman and Zadeh [1970] later coined this operator as the confluence of constraints with the possibility of it acquiring different meanings in different contexts of the problem formulation. Similarly, Zimmermann [2001] makes a case that the choice of an appropriate aggregation operator depends mostly on the context of the problem being investigated, with different operators requiring empirical experimentation to determine their fitness on particular applications. For example, if one wants to find out the degree to which a compound label satisfies all constraints, a T-norm could be used as the conjunction operator. Instead, if one intends to get the degree to which a compound label satisfies at least one of constraints, a T-Conorm is more appropriate.

The generic PFCSP global satisfaction degree outlines the common structure of various PFCSP global satisfaction degree formulae for aggregating operations on the local satisfaction degrees of all prioritized fuzzy constraints, where the role of the priority of a fuzzy constraint can be regarded as a prioritized factor to the local satisfaction degree of the constraint. Further axioms about priority operators capture some intuitive conclusions, such that the local satisfaction degree of a prioritized constraint should increase with its non-prioritized counterpart, and that the satisfaction degree of the corresponding prioritized

constraint decreases when the priority increases (as the problem becomes more difficult or over-constrained). This means that the higher the priority of a constraint, the more sufficiently the constraint should be satisfied in order to focus on a solution with a higher global satisfaction degree. When a constraint has complete priority, it is practically equal with its non-prioritized counterpart, while a completely non-prioritized constraint is not required to be satisfied, so it can be treated as having local satisfaction degree of 1.

3.2.3 (Fuzzy) Constraint Logic Programming and optimization problems

Constraint Logic Programming (CLP) generalizes logic programming by enhancing traditional unification with constraint modeling and solving over a particular domain (e.g. integer, real etc.). This enhancement of logic programming preserves the declarative nature of logic programming while utilizing existing mature implementations and well-understood semantics. A CLP engine provides facilities for describing traditional constraint satisfaction constructs such as domains, constraints, search strategies and others as clauses.

The usual procedure for solving a Constraint Satisfaction Problem using CLP is to define logical clauses for the $\langle X, D, C \rangle$ tuple in order to model all relevant information about the problem. Afterwards, additional clauses may operate on the constraints in C and drive the search process on how to find a relevant solution. The last step typically consists of an enumeration procedure which, combined with the backtracking mechanism inherent in Prolog engines and possible constraint propagation or editing mechanisms, tries to find an admissible solution to the problem specified. For these reasons, various research projects use a representation of a CLP knowledge base as a common representation of knowledge which can easily be transformed in a solvable Prolog program (such systems will be investigated in the next chapter).

A typical area of its application is combinatorial optimization problems, such as physical simulation, operational optimization, timetable scheduling, circuit verification and many others. Researchers and decision makers have systematically employed classical optimization methods to tackle the so-called hard systems (a class of the well-defined optimization problems) which are solved via the accurate mathematics of crisp objective functions that are subject to appropriate constraints. However, the random, uncertain and/or vague nature of real-life situations (such as randomness of occurrence of events, imprecision of sensors and ambiguity of system data, linguistic vagueness, diverse sources, deficits in

statistical data, subjectivity and preference of human judgment) require a more stochastic approach to modeling and solving such problems. Stochastic systems of this type which can deal with incomplete knowledge can be solved by stochastic optimization techniques using fuzzy logic, probability theory and other incomplete reasoning methods.

Fuzziness can be further classified into vagueness or ambiguity. Vagueness is associated with the difficulty of making sharp or precise distinctions, i.e. when the information cannot be valued sharply or cannot be described clearly in linguistic terms, as in preference related information. This type of fuzziness is usually represented by a membership function that reflects the investigator's subjectivity and preference on the objects. On the other hand, we have ambiguity when the choice between two or more alternatives is left unspecified, and the occurrence of each alternative is unknown owing to lack of tools and knowledge.

Ambiguity may be further classified into preference-based ambiguity and possibility-based ambiguity (or just imprecision), from the standpoint of where the ambiguity arises from. When ambiguity emanates from subjective knowledge or objective tools, it is a preference-based ambiguity, and is usually described by a membership function. If the ambiguity is due to incompleteness, it is a possibility-based ambiguity characterized by a possibility distribution, that simply reflects the possibility of occurrence of an event or an object. A soft system with vague and ambiguous information has an ill-defined structure, reflecting human subjectivity and ambiguity/imprecision. It cannot be articulated and solved efficiently by the traditional mathematics of optimization methods nor by stochastic optimization methods, which are based upon probability. Nevertheless, fuzzy set theory and fuzzy optimization techniques are functional and effective tools for modeling and optimizing such soft systems. Modeling and optimization within a fuzzy domain is termed fuzzy modeling and fuzzy optimization Fuller [1998].

The objective of fuzzy modeling is to create a befitting model that stems from the understanding of the problem and the analysis of the available fuzzy information, whereas fuzzy optimization aspires to an optimal solution of the fuzzy model via optimization methods and tools on the basis of formulation of the fuzzy information in terms of membership functions and/or possibility distribution functions, etc.

An optimization problem consists of two fundamental elements, i.e. an objective function (specifying the type of goal that needs to be achieved, such as the maximization of fitness or minimization of cost) and a feasible space (the search space containing feasible solutions to the problem). Fuzzy optimization refers to problems where the search goal

is an extremum of a real function with fuzzy coefficients, where the variable domains are fuzzily bounded, and/or when the abstract goal of the problem is fuzzy and thus there is no clear optimization objective. Similarly to deterministic optimization problems, in general, the FOP may be classified into two different types, namely, fuzzy extremum problems and fuzzy mathematical programming problems. In any forms of the fuzzy extremum problems, the extremum of the function is not a unique one, and there are no unique relationships between the extremum of the objective function and the notion of the optimal decision. The concepts of maximizing set, maximum and minimum of fuzzy numbers and some integral methods for fuzzy ranking can be applied to solve the fuzzy extremum problems.

Although there are no clear-cut boundaries between fuzzy modeling and fuzzy optimization, they are separate processes. The entire process of fuzzy optimization as applied to solve intricate problems may be decomposed into the following seven stages, as described by Yung et al TANG et al. [2004]:

- Understanding the problem. The state, constraints and goals of the system, as well as the relationships among them are understood and expressed by sets.
- Fuzziness analysis. The collected information is expressed in a semantic but fuzzy way. The researcher needs to determine which kind of fuzzy information is involved and what the position (e.g. fuzzy goal, fuzzy system of constraints, fuzzy coefficients) it takes, as well as the way (e.g. ambiguity/imprecision in quantity, vagueness in linguistics) in which it is expressed.
- Development of fuzzy model. Utilizing all the collected information is enough to build a proper fuzzy optimization prototype model by using appropriate mathematical tools and taking into account the characteristics of the problem. There are two methods for developing fuzzy models, e.g.. using the principles of cause-and-effect and transition, or using ordinary equations to express the cause-and-effect relationships. During model evolution, sets and logic relationships are first laid down and expressed as fuzzy terms. The optimization model may take the form of fuzzy linear programming, fuzzy nonlinear programming, fuzzy dynamic programming, fuzzy multi-objective programming, possibilistic linear programming or another optimization method Floudas and Pardalos [2008].
- Description and formulation of the fuzzy information. This stage takes care of the transition from fuzzy modeling to fuzzy optimization. The fuzzy information including

ambiguity and vagueness has been differentiated and needs to be quantified in terms of appropriate tools and theory using fuzzy mathematics. Taking into account the way fuzzy information is expressed, a membership function or a possibility distribution function can be selected. The membership function is subjectively determined, and preference-based. It usually applies to situations involving the human factor with its concomitant vagueness of perception, subjectivity, goals and conception, e.g. fuzzy goals with aspiration, fuzzy constraints with tolerance. Such goals and constraints are expressed vaguely without shard thresholds in order to provide the necessary flexibility of expression. Nevertheless, the possibility distribution function expresses the possibility measure of occurrence of an instance and can be constructed in either an objective or subjective way. This usually applies to the cases where ambiguity in natural language and/or values is involved, e.g. ambiguous coefficients/parameters in the objective function and/or the system of constraints. These coefficients are considered as possibilistic variables restricted by a possibility distribution. The membership function or the possibility distribution function may take a linear or non-linear form, depending upon the investigator's preferences and interpretation of the problem.

- Transformation of the fuzzy optimization model into an equivalent or an approximate crisp optimization model. It consists of three procedures, i.e. determination of the optimal solution, interpretation and transformation. The type of the optimal solution is determined, depending on preferences and the understanding of the problem. That is to say, selection of the type of the optimal solution to a fuzzy model depends absolutely on understanding and definition of the optimal solution in a fuzzy sense. The subsequent task is to propose an appropriate interpretation method and some new concepts to support the understanding and definition of the optimal solution, based on theories and principles on fuzzy mathematics, such as fuzzy ranking, extension principle, fuzzy arithmetics, etc. Finally, the fuzzy model is transformed into an equivalent or approximate crisp optimization model on the basis of the interpretation. For a fuzzy model, different forms of crisp optimization models may be built depending on different types of the optimal solution and interpretations applied.

- Solving the crisp optimization model. Depending on the characteristics of said model (i.e. whether it is linear or non-linear, single or multi-objectives or containing decision variables with continuous, discrete or mixed domains) appropriate optimization

techniques and algorithms such as metaheuristics, rule-based system approaches or hybrid algorithms, can be utilized for solving the model.

- Validity examination. The obtained optimal solution may not always be reasonable or applicable, thus further checks may be required to confirm its validity. If the solution is unreasonable, the fuzzy modeling process and/or the subsequent optimization process requires iterative improvements to better simulate the problem.

Fuzzy Optimization and decision making is not the focus of this thesis, so the interested reader may research this issue further in the following resources: Ramik [2001]; Vojtas [2005]; Lodwick and Kacprzyk [2010]

3.3 (Fuzzy) Description Logics

3.3.1 A quick introduction

In contrast to First-Order Logic (FOL), which is very expressive but semi-decidable (i.e. if an axiom is entailed then there is an algorithm to produce it, but if an axiom is not entailed then the reasoning process may not terminate), it is easier to select specific semantic features to create a logic which is tractable with polynomial-time decision procedures; complete expressiveness (e.g. via using any available boolean operator) makes satisfiability NP-complete. Description Logics (DLs), also known as terminological languages or concept languages, are a family of logic-based knowledge representation formalisms. A DL is a logical reconstruction of frame-based knowledge representation languages and describes the knowledge domain in terms of concepts (called classes, similarly to FOL unary predicates), roles (i.e. properties of classes, similarly to FOL binary predicates) and individuals (which are specific instantiations, or objects, of classes, similarly to FOL constants). DLs are used for inferencing and generating well-defined declarative semantics for most features of structured representation of knowledge. The fundamental modeling concept of a DL is the axiom, which is a logical statement relating roles and/or concepts.

Conjunction is an absolute requirement because it is the only way to reason with multiple properties, while value or existential restrictions are also required for basic reasoning. In that respect, DLs practically act as decidable fragments of FOL. Two minimal DLs satisfying those requirements are called $\mathcal{FL}$ and $\mathcal{EL}$ accordingly. Another one is $\mathcal{ALC}$, the smallest propositionally-closed DL, which is a combination of $\mathcal{EL}$ and $\mathcal{FL}$. $\mathcal{S}$ is used to describe $\mathcal{ALC}$ extended with role transitivity axioms, with additional letters indicating other extensions [3]. Of course, a more expressive DL comes at a semantic and computational overhead and requires additional support to implement as part of a DL reasoner.

A Knowledge Base (KB) contains all the knowledge statements described in an ontology, described as a tuple $\langle ABox, TBox, RBox \rangle$:

- The Assertional Box (ABox) is the set of assertion axioms, containing role assertions between individuals ($isStudentIn(VASSILIS, DIT\dot{}UOA)$) and membership assertions ($Student(VASSILIS)$), describing the **instance knowledge** of the KB.

[3] An extensive overview of naming conventions for DLs is available here: `https://en.wikipedia.org/wiki/Description_logic#Naming_convention`

- The Terminological Box (TBox) is the set of terminological axioms describing class axioms such as subsumptions between concepts ($C_1 \sqsubseteq C_2$) and equivalence (concept definition, i.e. $(C_1 \equiv C_2) \Leftrightarrow (C_1 \sqsubseteq C_2 \sqcap C_2 \sqsubseteq C_1)$), as well as datatype definitions, describing the **schema knowledge** of the KB.
- The Role Hierarchy (RBox) is the set of role axioms that describe relations, such as Object Properties or Data Properties. Most of the time authors implicitly consider RBox as a part of TBox.

A specific instantiation, i.e. an interpretation $\mathcal{I}$, is a model of a KB if it satisfies the assertions of ABox and the inclusions in TBox, collectively represented $\mathcal{I} \vDash KB$. An interpretation has a domain (with individuals being its elements), concept names (which describe subsets of said domain) and role names (which describe binary relations over the domain).

	Syntax	Semantics
TBox axiom	$C_1 \sqsubseteq C_2$	${C_1}^I \subseteq {C_2}^I$
RBox axiom	$R_1 \sqsubseteq R_2$	${R_1}^I \subseteq {R_2}^I$
	Trans(R_1)	${R_1}^I = ({R_1}^I)^+$
	$U_1 \sqsubseteq U_2$	${U_1}^I \subseteq {U_2}^I$
ABox axiom	$i:C$	$i^I \in C^I$
	$i_1 = i_2$	${i_1}^I = {i_2}^I$
	$i_1 \neq i_2$	${i_1}^I \neq {i_2}^I$

Figure 3.8: Axioms in $\mathcal{SHOIN}$ (Source: Xu et al. [2006])

3.3.2 The Semantic Web stack

A lot of standards have been proposed and defined by the World Wide Web Consortium (W3C) with various degrees of support and integration. Even when a standard is not officially finalized or integration between standards is not complete, a collection of popular ones may become the de-facto standard which the community is mostly focused on. A quick overview of such technologies follows:

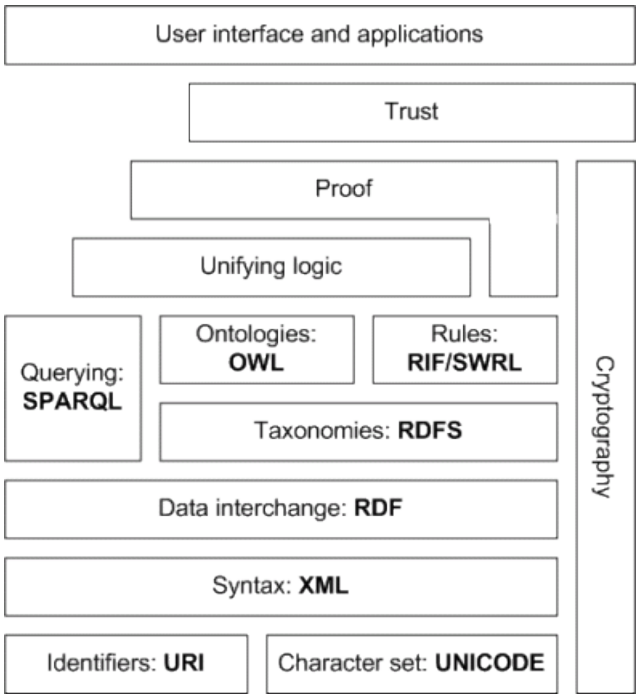

Figure 3.9: De-facto Semantic Web stack (source: semanticweb.org)

- Extensible Markup Language (XML) is used due to its extensibility and its applicability to both human readers and machine agents, providing the means of describing specific-purpose schemas useful in higher level ontologies[4].
- Resource Description Framework (RDF) is used to define metadata concerning online resources (described via Uniform Resource Identifiers (URIs)). RDF expressions are $\langle subject, predicate, object \rangle$ triples between URIs, where $subject$ is the resource being described, $predicate$ is the property or definition being attached to the $subject$, and $object$ the value being assigned to the $predicate$.

[4] A very promising and recent alternative, which provides data manipulation improvements compatible with RDF, is JSON for Linked Data.

- Queries on RDF data are executed via the SPARQL Protocol and RDF Query Language (SPARQL).
- In order to provide useful descriptions on specific applications (called RDF Vocabularies), extensions to RDF are provided via RDF Schema (RDFS).
- When generating RDF data and vocabularies, we aim to provide formal specifications to generate semantic meaning via agents working on shared conceptualizations. Initial work on this regard involved Ontology Inference Layer (OIL), DARPA Agent Markup Language (DAML) as well as other technologies. These were eventually superseded by the Web Ontology Language (OWL).
- This ontological description of data gives agents the ability to infer knowledge via a reasoning process. In order to enhance the reasoning process with Horn-like logical clauses, standards are introduced such as Rule Interchange Format (RIF) and Semantic Web Rule Language (SWRL) which are used by rule engines. Rule engines provide transformation utilities: whenever the conditions specified in the antecedent part of a rule hold, then the conditions specified in the consequent must also hold, effectively introducing new knowledge in the KB.

3.3.3 OWL and Rules

Over the past years, DLs have gained popularity mainly due to their expressiveness and algorithmic completeness, as well as their usability in the Semantic Web. One of the top abstraction layers of the Semantic Web is the popular OWL, the de-facto standard for knowledge representation and ontologies. The initial OWL specification described three sub-languages of increasing complexity and expressiveness:

- OWL Lite supported basic classification hierarchies and constraints
- OWL DL provided a more complete correspondence with description logics while maintaining decidability and basic performance guarantees
- OWL Full provided full expressiveness and syntactic freedom without computational guarantees while also providing facilities to enhance existing RDF and OWL vocabulary.

These have now been superseded by OWL 2 which provides more complete facilities and various fragments/profiles concerning different performance requirements [5].

As description logics focus on describing relations between concepts, standards such as OWL lack facilities for defining object and data properties through axioms or via reasoning on specific individuals. In other words, OWL addresses classification problems that can be expressed using Description Logic (i.e. tractable subsets of First Order Logic). Practical applications often need to augment OWL with rules that either cannot be implemented with OWL or appear prohibitively difficult or counter-intuitive to implement [Rudolph and Hitzler, 2008]. Work on rule engines such as SWRL can help overcome such shortcomings, although they add considerable overhead and possible decidability implications (SWRL is semi-decidable) [Parsia et al., 2005].

Nevertheless, the reason that a rule engine is, for example, appropriate to implement complex arithmetic reasoning and integrity constraint relations is that OWL is not expressive enough to reason about certain attribute values, function symbols are excluded, the arity of predicates are restricted, and most importantly the interpretation of negation assumes that the data is incomplete and/or the model inconsistent. For example, the definition of a relation ”forbidden” between two object classes would require a new operator, while a flexible cardinality variable is not feasible since this requires the definition of complex relation descriptions (i.e. property descriptions in OWL) or data/individual-centric property descriptions (e.g. ”no individuals should be included in this object class”), which is not supported by OWL and either not possible with built-in SWRL methods or impractical without additional extensions (e.g. the makeOWLThing predicate).

Even though SWRL is the typical rule engine demonstrated, decidability concerns limit its expressive power and reasoners usually implement the DL-Safe SWRL subset. One of the main expressivity restriction is that variables bind only to explicitly named individuals, practically downgrading it from a powerful TBox and RBox axiom to an ABox axiom [6]. Some typical approaches, based on the implementing **reasoner**, are presented:

- **Hoolet**: translate SWRL into First Order Logic and demonstrate reasoning tasks with a theorem prover [Alecha et al., 2009]
- **Bossam**: translate OWL-DL into rules and feed them to a backward/forward chaining (RETE-based) inference engine [Jang and Sohn, 2004]

[5] http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/

[6] http://weblog.clarkparsia.com/2007/08/27/understanding-swrl-part-2-dl-safety/

- **Pellet**: expanding the tableaux algorithm to handle rules

Data interchange languages such as RDF and even knowledge representation formalisms such as OWL, although they are the basic building blocks of the majority of current implementations, have some limitations which introduce difficulties when aiming for interoperability or when requiring features readily available in custom systems. For example, Vanekova et al. [2005] demonstrated a system which introduced a Fuzzy RDF specification which integrated fuzzy logic. This specification was used to translate the RDF Graph data (i.e. collection of RDF triples) into database data and queries, which in turn were integrated into two different Inductive logic programming (ILP) systems (Aleph and GOLEM) by transforming the background knowledge into a Prolog-compatible representation. The aim of their project was to assign ranking metadata to subjects using fuzzy sets transformed into appropriate database schema and queries, in order to be fed as background knowledge for the ILP system. There were differences between the systems (e.g. Golem would not support database queries as rules) and implementation requirements (RDF data had to be transformed into N-ary or binary predicates, resulting in additional work and harder representation). Additionally, the lack of a single, coherent formalism or platform produced a very specific-purpose system, lacking interoperability and extensibility. Eiter et al. [2008] provide an extensive overview of using rules to reason with RDF, RDFS and Constraint Interchange Formalism (CIF) knowledge bases which will not be covered here.

An ontology language such as OWL may be used to describe different classes of relations, and connections between data concepts can be achieved by utilizing such relations via a rule engine such as SWRL or RuleML. When managing large or diverse knowledge bases with the aim of creating interoperable representations, it is apparent that the successful modeling of axioms is important, as it alleviates the cumbersome manual generation of concepts and properties, provides additional inference opportunities, and can lead to significant semantical enrichment of KBs [Staab and Maedche, 2000]. It was the case though that systems often failed to visually demonstrate rule axioms similarly to concepts and properties, which can be a deterrent to their use. This is mainly due to the diverse specifications for describing rules, further complicated by the different knowledge representation technologies on which they are applied. Additionally, the recognition of axiom semantics is an undecidable task and so axiom specification translation and representation may not be effectively recognized. Another reason that rule axioms have not been extensively worked on is that their expected functionality ranges from being considered a subset of FOL calculus (to be used only when required, due to its complexity and decidability

costs), to being knowledge-based application development paradigm by itself (used to provide readable and extensible axiomatic descriptions). [Staab et al., 2003] argue that rules should be viewed as a knowledge representation language in order to promote diverse implementations, although that would mean that maximizing the utility and interoperability of rule engines would require a standard rule language layer. Meech [2010] discussed that in order for rule engines to be considered relevant, they have to be usable enough to be easily integrated and used to describe business rules. He notes that the OWL and SWRL standards have some adequately complete implementations as well as good community support, suggesting that their combination has the expressive power required for Business Rule Engines, providing a usable alternative to existing rule engine systems although they still lack mechanisms for dynamically accessing and updating external data sources. There is no standard system of rules in use and the performance considerations are important, as such systems typically use exponential time for the inference process. At the same time, during the modeling and querying phases, fast system response is important to guide the ontology engineer, but useful time may be wasted on slow implementations which eventually lead to inconsistent answers.

Mas et al. [2005] discussed how SWRL can be used to add integrity constraints in an ontology and how such information can be shared among different schemas. Yaguinuma et al. [2014] introduced “Fuzz-Onto”, a meta-ontology for describing fuzzy terms and rules in a Semantic Web context. These kinds of problems and relevant work are a demonstration of ontological engineering, which we will discuss in section 4.2.

Additionally, due to the relevant immaturity of the Semantic Web field, there is great fragmentation on the tools and standards used to implement tools, and some are promoted in specific areas and applications while others become stagnant. For example, frameworks such as SPIN [7], Jena [8], RIF [9], the recently proposed SWRL2 [10] and other rule engines can be considered as ”competing” recommendations or defacto standards. Other extensions, such as the fuzzy extension SWRL-F [Wlodarczyk and Rong, 2011], may never gain enough traction in order to become practical and user-friendly tools. Only recently there has been a push to promote the RuleML and bridge implementations with other standards in a way that can provide a common ground for the community [Boley et al., 2010].

[7] http://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/
[8] https://jena.apache.org/
[9] http://www.w3.org/TR/rif-overview/
[10] http://wiki.ruleml.org/index.php/SWRL2

A lot of work on the semantics of description logics and their inference services, as well as practical performance requirements when creating DL Reasoners such as Hermit, Pellet and Fact++, have produced a wide range of optimizations and heuristics for reasoning with description logics. Many such methods are mentioned in the referenced bibliography [Sirin et al., 2006; Tsarkov and Horrocks, 2006; Motik et al., 2007; Sirin et al., 2007; Bobillo et al., 2008b; Glimm et al., 2010]. For an extensive overview and in-depth analysis of all relevant history, theory, formalisms and practical applications, the "The Description Logic Handbook" is a highly suggested, complete and useful resource for further study [Baader et al., 2007][11].

3.3.4 DLs vs Fuzzy DLs

As we described in previous sections, Description Logics (DLs) are used on a wide range of applications and, during the last decade, have increasing popularity due to their application on the Semantic Web Baader et al. [2005]. However, typical description logics are designed to deal with crisp domains and concepts, while real-world applications deal with incomplete information. The notion of vagueness on the web means that concepts encountered in the real world do not usually have specifically defined criteria of membership. This is a major issue for the traditional DLs since they cannot handle problems that support probabilistic domains. In addition, current systems pose syntactic or reasoning limitations that require extensions and enhanced semantics to support said applications. To address this issue, Straccia [1998] introduced the first Fuzzy Description Logic (f-DL) and the popular FuzzyDL reasoner [Bobillo and Straccia, 2008] to manipulate and reason with fuzzy domains. There have been additional reasoners for fuzzy ontologies, before and since then (described in-depth in later sections), as well as extensive but on-going research on the semantics and decidability of f-DLs for specific and generalized instances [Stoilos et al., 2006a, 2007; Liu et al., 2011], as well as integrating or merging multiple ontologies into fuzzy knowledge bases (e.g. the work of Qi et al. [2004]) which we will not discuss in this thesis. Lukasiewicz and Straccia [2008] provide an extensive overview concerning the description of imprecise knowledge via ontologies and according semantics, as well as describing the concepts of probability, possibility, uncertainty as well as incomplete, vague and fuzzy knowledge. Additionally, f-DLs have been shown to be isomorphic to classical DLs [Straccia, 2004],

[11] Another very quick introduction is available here: `http://www.isi.edu/~blythe/cs541/Readings/loom.pdf`

which means that one may either focus on enhancing existing methods for supporting fuzzy semantics or produce an equivalent crisp representation of the fuzzy KB and use existing methods. Examples of both strategies will be discussed in the following section.

3.3.5 Current Fuzzy DL reasoners and implementations

3.3.5.1 yadlr

Konstantopoulos and Apostolikas [2007] described a fuzzy DL inference system used for vague ontological reasoning on semantic metadata provided by user feedback. Their modular Prolog system[12] integrates variable fuzzy degrees and is able to use multiple deduction methods and fuzzy arithmetic systems (i.e. algebraic norms), providing ABox reasoning services via SL-resolution extended with tabling (SLG Resolution) over the $\mathcal{SHROIQ}$ description logic. The aim of the authors was to identify fuzzy degrees that are close to the initial system's results and accommodate user input (used to determine what is considered a satisficing result), by fine-tuning the acceptance levels (i.e. threshold fuzzy degrees for membership in an abstract concept) based on concrete feedback and back-propagating this information to the reasoner (for iterative optimization of feature detection). During SLG Resolution, transformations are applied only when the new rules satisfy the acceptance threshold. The consistency of the ABox can be readily verified if all fuzzy degrees are known. If there are unknown fuzzy degrees, the inference engine iteratively calculates derivative degrees and integrates more restricted ranges in the KB by applying linear $clp(\mathcal{Q}, \mathcal{R})$ constraints and determines the lower satisfying bounds.

3.3.5.2 FuzzyDL

FuzzyDL [Bobillo and Straccia, 2008] is one of the most popular fuzzy reasoners, mainly due to the FuzzyOWL2 Protege plugin, its fuzzy ontology specification and the extensive work of its authors on f-DL semantics. It is a Java inference engine providing full reasoning services over $\mathcal{SHIF}$ DL[13]. Different ***membership functions*** are pre-defined (namely Trapezoidal, Triangular, L-function, R-function, Crisp interval and Linear) and fuzzy degrees are mapped via ***fuzzy modifiers*** according to the investigator's needs, permitting explicit definition of fuzzy concepts. Additional features implemented are concrete datatypes and

[12] `http://sourceforge.net/projects/yadlr/`

[13] `http://gaia.isti.cnr.it/straccia/software/fuzzyDL/download/old/documents/documents.html`

various concept constructs, such as threshold concepts (which can be used as acceptance thresholds during reasoning) and weighted sum concepts (for describing complex fuzzy concepts). In addition to traditional, crisp operators, Zadeh and Lukasiewicz semantics are supported for fuzzy reasoning. Fuzzy degrees are quantized in a discretized set of n values in order to accommodate its Mixed-integer linear programming (MILP) engine for its numerical operations, which also permits degree restrictions and constraint relations between concepts. An additional interesting feature is the defuzzification operation, where a fuzzy set is aggregated to produce a single output variable via an operation on its degrees (such as selecting the smallest, the largest or middle of its maxima). All these non-classical features are implemented via XML annotation properties and can be shared via the FuzzyOWL2 specification [Bobillo and Straccia, 2010], making it a system with extensive fuzzy syntax support and semantics and an appropriate platform for extension and collaboration.

3.3.5.3 FiRE

The FiRE[14] fuzzy reasoner [Simou and Kollias, 2007] is a Java system implementing the authors' f$_{KD}\mathcal{SHIN}$ f-DL specification, and constitutes the first tableau reasoner for f-DLs. The system was designed to accommodate, additionally to entailment and subsumption, TBox and RBox inferencing services and cardinality restrictions, although it does not support (custom) fuzzy datatypes. ABox consistency is maintained until the first tableau clash, showcasing the relevant tableau expansion. Optimization queries can be used in order to determine the most appropriate bounds and support degrees after ABox inferencing. This prototype implementation is an interesting graphical testbed for extending known tableau optimizations over fuzzy logic although it utilizes its own, non-standard syntax and file-saving format.

3.3.5.4 LiFR

A very recently introduced system, LiFR[15], is a Java lightweight reasoner extending Pocket-KRHyper Kleemann [2006] with fuzzy semantics sacrificing some expressive power in order to be usable on mobile platforms for the LinkedTV cloud project, used to implement the LinkedTV User Model Ontology and perform content filtering based on user profiles

[14] http://www.image.ece.ntua.gr/~nsimou/FiRE/
[15] http://mklab.iti.gr/project/lifr

[Tsatsou et al., 2014]. An interesting feature of this reasoner is that, in order to achieve the required performance, its implementation supports the expressive fragment of Description Logic Program (DLP) [Grosof et al., 2003] (for the interested reader, a DLP is a combination of DLs under the general FOL semantics with logic programs under the answer set semantics, covered extensively in [Eiter et al., 2004]). Although the system is restricted to Zadeh logic operations and crisp rules, its main advantage is performance, as it is shown to be significantly faster to FuzzyDL and even FiRE.

3.3.5.5 DeLorean

The DeLorean system[16], introduced in Bobillo et al. [2008a], optimizes a fuzzy $\mathcal{SHOIN}$ or $\mathcal{SROIQ}$ KB and transforms it into an equivalent crisp representation. Although not a reasoner per se and limited to a trapezoidal fuzzy datatype with a finite chain of degrees (with Zadeh or Gödel semantics), it permits the reuse of classical tools and resources for operating on a fuzzy KB and also provides a user friendly interface for combining reduction and optimization methods with crisp reasoning procedures.

3.3.6 Optimization strategies for Fuzzy DLs

During the development of FiRE, Simou et al describe some basic but effective optimizations implemented in their system that enhance the reasoning performance of the tableau algorithm (which is the prominent method implemented in DL reasoners), described in the following three subsections, with all being of polynomial complexity and agnostic to the fuzzy logic being used [Simou and Mailis, 2010]. Other recent work provides useful insight on possible ways to parallelize the reasoning capabilities for ABox and even TBox inferences [Bock, 2008; Urbani, 2010; Ren et al., 2012]. Recent work has also aimed to provide web-scale capabilities to f-DL reasoning via the MapReduce algorithm [Liu et al., 2012].

3.3.6.1 Degree Normalization

Fuzzy ontologies may contain assertions where an individual participates in the same concept with different degrees, via different rules, without forming a contradiction. The existence of multiple such instances means there are superfluous degrees of truth which

[16] `http://webdiis.unizar.es/~fbobillo/delorean`

inevitably lead to additional clash checks in the tableau algorithm and degradation of performance. Since a fuzzy ABox may contain multiple assertions with varying degrees, it is important to compute the optimal upper bound (Least Upper Bound (LUB)) and more importantly lowest bounds (Greatest Lower Bound (GLB)) of the degrees of a fuzzy assertion, depending on whether T-Norms or S-Norms are used; Straccia defined this requirement as Best Truth-Value Bound (BTVB). Thus, a process of degree normalization may compact the ABox and improve the performance of the algorithm by keeping the least negative and largest positive assertions. This method can be enhanced with rules for early clash detection during tableau reasoning by utilizing these two values (because the sum of the degrees of a fact and its negation cannot be larger than 1).

3.3.6.2 ABox Partitioning

Due to the hierarchical data structures implemented in the tableau algorithm, expansion rules may generate independent subtrees. Thus, the assertional component of a fuzzy knowledge base can be divided in smaller partitions, which can be examined independently by the algorithm. If all partitions are consistent then the ABox will also be consistent, but if even one of the partitions is inconsistent then the ABox will be inconsistent and the search process can break early. During search, any subtrees that are not connected to the node being partitioned are not affected by the search and can temporarily be disregarded by the reasoner.

3.3.6.3 Greatest Lower Bound

Efficiently determining the GLB value for each individual's membership degree to any concept is an important operation, as it is used in most fuzzy operations. Such a decision process was initially proposed by Straccia [2011], where he determined that the BTVB and the subsumption problems can be reduced to the entailment problem (i.e. determining whether a fuzzy assertion is true or, in other words, has an adequate truth degree). This is achieved, quite intuitively, by partially ordering the assertions and performing binary search to minimize the required satisfiability checks. Simou et al propose two additional optimizations to this method: using ABox partitioning to select on selections containing the affected individual, and caching previous tableau tree expansions to prevent similar expansions and satisfiability checks.

3.3.6.4 General Concept Inclusion absorption

Stoilos et al. [2006b] initially described the required semantics for General Concept Inclusion axiom (GCI) absorption methods on f-DLs and provided a more extensive overview in a recent study [Bobillo and Straccia, 2013] (this concept and some further optimizations were also discussed by Steigmiller et al. [2014]). GCI formulas are the subsumption and equivalence axioms involving complex (i.e. non-atomic) concepts, a very useful, expressive and thus expensive feature of Description Logics. Even though the family of tableau algorithms are easily adaptable and extensible, they have high worst-case complexity and may easily lead to large fan-out in the reasoning tree due to disjunction or even infinite clause expansion due to a non-acyclic KB; checks and optimizations are required in order to rewrite GCI axioms and optimize performance, as these cannot be handled by simple tableau expansion. This practically means that $C \sqsubseteq D$ must be transformed into $\top \sqsubseteq D \sqcup \neg C$ which the classical algorithm can check[17]. An effective optimization for this problem is the absorption method, where axioms are transformed so that this process of "internalization" and the processing of disjunctions during tableau node expansion (leading to fan-out during the reasoning process) are avoided as much as possible. This is achieved by extracting easily checked conditions for each axiom of the TBox and only then adding the rest of the complex expression to the expanded node's label (otherwise, the axiom is trivially satisfied). The Tableau internals and its internal optimizations can be studied in [Ding, 2008]. A simpler optimization enhanced by absorption is decreasing the number of GCI axioms via composition (e.g. when two axioms can be replaced with one, such as $A \sqsubseteq B$ and $A \sqsubseteq C$ combined into $A \sqsubseteq B \sqcap C$) or simplification (e.g. eliminating trivially satisfied concepts or transforming trivial complex concepts to atomic ones). Even though GCI are a very important feature of practical ontologies, there are intuitive and theoretical indications that they may render such DLs, in the general case, undecidable [Baader and Penaloza, 2011; Borgwardt and Distel, 2014], although specific assumptions and checks (i.e. on the T-Norm operators used) may be integrated in order to permit practical applications with adequate performance.

[17] this feature is not straightforward in a fuzzy context and requires some fuzzy tableau extensions to be implemented which will not be discussed here

3.3.6.5 Parallelization of ABox and TBox reasoning

An obvious optimization for reasoning systems is to execute threaded processing of reasoning subtrees for faster performance. This is especially true in big data applications where large data sources and multiple systems are available. When the ABox or TBox data are internally partitioned in such a way that its formulas are independently distributed, each tableau subtree can be expanded independently. When executing queries on possibly vast number of RDF triples, other methods are required, such as executing the query via MapReduce compatible algorithms or using appropriate subset fragments of DL which may not support the full width of expressivity required. Additional assumptions such as implementing monotonic reasoning (e.g. not retracting known facts) or caching sub-graph results may provide an additional performance boost but with non-guaranteed consistency.

3.3.7 Final thoughts

A lot of reasoners are available over broad areas of application, but unfortunately they still lack practical tools, extensive optimizations and well agreed standards for implementing and sharing work on f-DLs. Fuzzy Description Logic (f-DL) technology has only recently matured enough to show its potential as a viable alternative, so new developments and applications are expected. It is now time to see how one could implement Constraint Satisfaction Problems (CSPs) in f-DL, to investigate the effect or potential of higher-level tools and what possibilities lie ahead. Dentler et al. [2011] provide an description of performance characteristics that should be considered important for selecting a reasoner (based on the performance and expressivity requirements of the system).

4. Constraints and Ontologies

As described in chapter 2, constraint satisfaction and optimization problems are applicable in many areas and, although CSPs are a very mature formalism for describing and solving such problems, the creation and/or integration of appropriate solver software and the exact modeling and fine-tuning of instances over broad knowledge domains requires extensive expertise, experimentation and time. The use of ontologies to provide conceptual contexts for constraint solving and to ”shield” users from constraint programming details is a general idea that may be useful in various applications [Wallace et al., 2007]. Additionally, with the increasing popularity and maturity of the Semantic Web and the Internet of Things (IoT), ontology-related technologies are becoming expressive and practical formalisms for sharing models and describing interoperable representations of them. As discussed on chapter 3, fuzzy knowledge bases are of special interest as they are under active research and provide promising representation and reasoning features. In this chapter, we will discuss current work on the area of fuzzy knowledge engineering and especially fuzzy constraint satisfaction ontologies, interesting extensions and future possibilities.

A fuzzy ontology system may have additional useful applications which may be modeled as a fuzzy optimization problem (and thus an f-CSP), including:

- **Matchmaking**: By specifying fuzzy weighted concepts and implementing restrictions as fuzzy constrains, a knowledge base may provide a description of services, domain information and ranges of available options. User preferences may be expressed via fuzzy degrees, possibly in a linguistic way. The system may then execute fuzzy decision making on user queries to determine whether there are available options that would satisfy the user's preferences and, if not, suggest alternative partially consistent configurations [Ruta et al., 2010].

- **Fuzzy Control**: Fuzzy Control is an old and well-researched area of applications for fuzzy logic. Nevertheless, new research in ontologies and integration of sensor data and services on the IoT provides further possibilities for development. Such an example is the Ontology Web Language for Fuzzy Control (OWL-FC), a domain independent model formalism useful for re-usable description of such systems. Specifications and constraints can be modeled as concepts and rules are used to infer the appropriate instances (i.e. controllers) for the task at hand [Loia et al., 2010]. Such a formalism may also be applied in multi-agent or robotic control systems, where for

example CSPs may be used to solve sequential manipulation planning problems or mobile path tracking [Nacer and Jilani, 2014].

- **Classification**: By specifying a fuzzy knowledge base describing sensor data or multimedia signal data, fuzzy reasoning can be used in order to provide classification services. Fuzzy DL technologies, including spatial and temporal reasoning capabilities, may lead to interesting and unexpected applications such as the one discussed by Eich [2013], where a semantic description of a marine vessel's space (represented by a LIDAR point cloud) is used to execute spatial classification and infer metadata via fuzzy rules concerning the structure of the cargo hold. This application is especially useful when a researcher is simply interested in visualizing and classifying large amounts of vague data, where fuzzy rules may be used in order to categorize data into appropriate classes and then use this inferred knowledge to generate relevant visual representations. A fine example of this practice is presented by Danyaro et al. [2012], where the authors integrate meteorological data into a fuzzy KB and then reduce the inherent uncertainty and ambiguity by applying fuzzy rules and annotating them via fuzzy concepts. Another example is the research target of Simou et al. [2008] on the FiRE reasoner, which is the analysis and classification of multimedia content, inserting semantic knowledge into a fuzzy KB and then permitting the retrieval of content based on semantic metadata. A CSP-based approach to classification may model a fuzzy optimization problem where the goal is to maximize correct classifications and at the same time minimizing the information acquisition cost. Pendharkar [2006] demonstrated such an approach by formulating this problem as a set of binary variable knapsack optimization problems to be solved sequentially.

- **Web Services**: Integration of web services by fuzzy discovery is a relatively new practice. As a more specific instance of matchmaking, there may be multiple available and similar web services which may be (fuzzily) evaluated before choosing the appropriate candidate [Tsetsos et al., 2006]. A novel method of service discovery may use fuzzy ontologies to describe services, provide integration facilities and then match appropriate services based on specified criteria, which can be achieved with constraint-based matchmaking procedure. Based on the recent surge in popularity of IoT research (e.g. [Wang et al., 2013]), there are interesting possibilities in this area of research.

4.1 Managing constraint problems with ontologies

Descriptive Constraint Library (DCL) is a draft formalism we investigated for describing constraints in an ontology. This tool would facilitate the description of a constraint domain via concepts and object properties for describing variables, data properties for describing value domains, and DCL-introduced annotation properties to describe constraints. Such a formalism could facilitate the investigation problem modeling, constraint interactions or possible optimizations, providing the means for an ontological description of the CSP problem which can either facilitate experimentation with inconsistent modeling and constraint relaxation, or even be transformed to an initial model to be solved (for example, by producing Constraint Logic Programming (CLP) code). The generalized syntax for describing a CSP-related ontology will be called DCL-Full, where additional annotation can be inserted in order to communicate metadata about the model. A simplified version, called DCL-Lite, would be used to describe the computable model and possibly be translated into other forms (such as an intermediate specification or a valid CLP program) to be integrated in existing solvers and reasoning systems. When a specific model of the problem is inconsistent or unsolvable in an appropriate time frame, the investigator is typically required to manually relax some constraints in order to quickly find an initial solution. We have discussed how interactive constraint satisfaction helps in this regard and our system could facilitate experimentation on this area. For example, a locally consistent instantiation could be imported into the system (i.e. a possible partial solution can be asserted and included in the KB) and then the investigator may execute appropriate changes to the ABox and TBox description, checking for inconsistencies. Another way to think about this process is that we introduce a specific instantiation and try to pinpoint the inconsistencies. Before proceeding into such a venture, we would have to investigate the latest research in order to find about any efforts made in that direction, its focus and its limitations.

Kim et al. [2003] describe an OWL extension, called Semantic Web Constraint Language (SWCL), intended to provide constraint representation in existing Semantic web environments. They first describe currently used methods to express more information about existing rules and thus provide richer expressiveness to existing (OWL-based) knowledge systems. One of such common tools is SWRL which integrates RuleML functionality into OWL-DL. In other words, OWL KBs can provide further inference capabilities by integrating Horn rules, even though they trade decidability for expressiveness. Preece et al. [2000] introduced the $(CIF/SWRL)$ formalism, where he provided constraint semantic extensions

to SWRL via RDFS, initially demonstrated in the KRAFT platform [Hui et al., 2003]. These descriptions can then be enhanced by different knowledge bases or even transformed to other formalisms. A different approach could permit arithmetic constraints which could integrate various solution approaches and optimizations and thus focus on optimization techniques. The authors introduce said approach by implementing SWCL as the basis of a system for linear optimization. The main focus of a linear optimization problem is to express a mathematical relation between variables in a specific numerical domain. Thus, in an ontology environment, we could express a mathematical relation between concepts via data properties. The formalisms of OWL and even SWRL are not adequately expressive for such a formulation, but the SWCL extension provides a basis towards such an approach.

The authors then describe a practical shopping agent example to demonstrate the difficulty of finding optimal or event relevant shopping information by typical search engines, making a case for enhanced semantic information which may be provided via SWCL. Even though the initial OWL and SWRL implementation may describe concepts, individuals and data properties relevant to a specific shopping model, the formalism is lacking practical constraints that would efficiently describe hierarchical relations between those elements. For example, one could not describe a discount policy which would take into consideration the inferred discount amount when computing the total checkout cost. SWCL not only provides this facility, but also gives the agent the ability to optimize its shopping order based on the supplied constraint model in order to achieve the optimal purchase.

More importantly, SWCL provides new axioms to describe linear optimization concepts, extending the OWL semantics [Patel-Schneider et al., 2004]. The core of the SWCL formalism lies in specifying the model of an optimization problem, which is a collection of objectives and their relevant applied constraints. The authors introduce ”Objective” and ”subject˙to” terms (used to describe optimization goals (specifically min or max) and constraint relations between ”factors”, i.e. variables and individuals), which provide a mapping between SWCL terms and OWL resources and introduce basic constraint and optimization features in the SWCL ontology. A constraint is simply the expression of an arithmetic constraint between two aggregate terms, which is a sum or a product of the values of specific data properties (applied on classes or specific individuals via ”class factors” and ”individual factors”). The authors provide a framework which integrates various reasoning and interface systems together in order to identify possible goal, constraint and model sources, furnishing an easy way for the user to specify the optimization problem and handle multiple backends. The ontology and the relevant optimization model are transformed to a CLP de-

scription which can be solved with typical methods. A problem solver interface is provided which can be utilized by multiple problem solvers, such as GNU Prolog and ILOG Solver, and was empirically validated by implementing various optimization problems and checking their execution results in Optimization Programming Language (OPL) via the ILOG POL Studio solving software.

A possible extension to the system would be to propagate or enable aggregate operations on data properties. Let us think of an example of a geographical ontology, containing population, perimeter and area data for various communities and provinces of a country. Similarly, a data property describing age or time may be aggregatable is some situations (for example, the average of all ages in a group) but in others not (in the case of summation of running times for subtasks in a specific project). Such an ontology would contain an hierarchy of concepts where a country would contain provinces and a province would contain communities. It follows naturally that the area of different communities equals the area of a province and, in turn, areas of member provinces equal the area of the country. In multi-depth hierarchies, this would linearly expand the number of constraints that the user is required to manually specify. An extension to the system is possible where we can specify that the area and population data properties are aggregatable on the membership property and thus the system can automatically infer the relevant constraints. That way, it is also possible to specify non-aggegatable data properties such as perimeter, as the perimeter of a country does not equal the perimeter of its provinces.

Croitoru and Compatangelo [2007] propose a visual representation for constraints in knowledge-rich domains. Their visual mechanism was able to build upon KBs regardless of the chosen representation language by building on the semantic properties of Conceptual Graphs (CGs), which are visual formalisms for intensive domain knowledge representation initially used for representing conceptual schemas in database systems. The authors formalize the combination of CSPs and CGs by mathematically defining the preliminary notions of ordered bipartite graphs, support, conceptual graphs and the projection (or subsumption) relation. In summary, they define operations on two corresponding graph sets in order to indicate mapping between concepts and relations. An “ordered bipartite graph” is a tuple consisting of a set of concept nodes, a set of relation nodes and a mapping between the relation nodes and non-empty, finite sequences over concept nodes. “Support” is simply the domain knowledge defined by the known facts and the taxonomy of the problem (that is concept and role hierarchy, known individuals and unknown, inferred concepts). The “projection” operation is a labeled graph homomorphism which defines a

generalization-specialization relation over the CGs, indicating that a specific graph structure is more general than another.

Their motivation for a coherent visual representation is based on the thought that Semantic web applications dealing with CSPs should be addressed in a domain oriented manner in order to exploit the ontological information encoded in the knowledge representation formalism. They argue that in order to achieve this, a clear visual representation is required. The central element of their approach is that using CGs provides semantic metadata for:

- easy knowledge acquisition (instead of traditional CSP formulations, visual representation of constraints makes the identification of problem-specific constraints easier),
- enhancing the reasoning process (traditional CSP strategies can avoid backtracking steps by checking CG node labeling because existing partial solutions for sub-graphs can be saved and reused), and
- permitting the transformation of the intermediate CG representation into various reusable formalisms such as RIF, OWL or even CLP.

In conclusion, the visual representation of a CG-derived model expresses, via graphs, all the required information to model a CSP's variables, their corresponding domains and the constraints. This representation gives, as such, a static view over the CSP instance to be addressed. The advantage of this framework is that arbitrary solution and reasoning methods can be employed, as well as hyperheuristics and local search methods. On the other side, a static representation of a large domain complicates the constraints in some complex scenarios with many alternative configurations, such as scheduling and planning. In those cases, the CG model needs to be extended by new syntactic operations. One way of solving this problem is to design a dynamic representation; Conceptual Graph Assembliess (CGAs) is a structure which contains a CG along with a combinatorial structure on its relation nodes and can capture alternative knowledge views on data instead of traditional, static facts. Bringing together the CSP formalism and visual knowledge representations has research potential both from a theoretical and an application point of view. The authors' goal was neither to provide competitive results, nor to propose specific problem driven combinations which compete with state of art solvers, but to illustrate the use of this framework in designing hybrid strategies to addressing CSPs.

Preece et al. [2008] presented a proposal for representing FCSP with soft constraints using Semantic Web technologies. Their work was motivated by the need for a service-providing agent in a Virtual Organization (VO) which could reason about its tasks in a dynamic fashion. Semantic Web services, such as e-commerce, e-science and e-response, constitute a service-provider managing particular resources for satisfying specific goals and queries. A interesting feature of such organizations is that the commitment of resources to goals is typically governed by service-level agreements, which can be modeled as soft constraints on resources managed by a FCSP model. When a service provider is presented with a new potential commitment, it must perform reasoning to determine if it can take on this commitment, possibly by dropping (breaking) existing lower-utility commitments. Their system is built on CIF and proposes a new ontology for representing FCSPs, making it potentially usable with other constraint and rule representations. This ontological description is used as an intermediate form, the goal being to facilitate interchange of information between a CSP problem constructor and an appropriate solver, where the solver would execute the reasoning process, determine any possible solutions, and then use this information to check for commitment violations.

The three essential requirements addressed by the authors are:

1. expressing commitments in terms of Semantic Web Services,
2. associating utility values with constraints in order to reflect their relative importance, and
3. determining which are satisfied or violated by any given solution and how it affects other serviced agents.

It is apparent that the increasing number of agents and commitments in live systems leads to combinatorial explosion and also introduces many antagonizing service requests. Additionally, the number of trivial solutions also increases, wasting memory and time without introducing significant gain. Thus, in order to support the main characteristic of their system (i.e. finding adequately different solutions that break commitments), the authors introduce methods for prioritizing commitments and evaluating the “differentiation” between solutions, in order to quickly determine when an important commitment is being violated. This “commitment management” system is implemented as combination of reification (i.e. the creation of a data model) and constraint value labeling in order to provide the required prioritization and commitment metadata to the solver software. Some additional

features implemented are: enhancing SWRL with nested quantified implications, a new Quantifier class, providing an RDF syntax specification which includes definitions for class constraints, properties such as “hasQuantifiers” and “hasImplication”, list ranges, “OrExpression” and “AndExpression” axioms which are defined as sub-classes of rdf:List, and a Negation class which uses an SWRL argument property to describe the negated atom. These axioms have intuitive and apparent mappings to traditional CSP definitions and constructs and would ease the transition from an ontological description to a valid program or specification for reasoning by a CSP solver, as there would be less need for transformation between those forms.

The authors note that a utility value is not an intrinsic part of a constraint itself but simply a metadata annotation, which is also the case for constraint satisfaction. Therefore, a separate ontology was created by the authors to represent a CSP, independent of the CIF/SWRL representation of the individual constraints themselves. The authors created two variant representations of their CSP ontology, a DL-Safe one and another enhanced with SWRL rules. In either case (the latter being easier to express), the modeled information is intended only as an interchange format, as the transformation and resolution of the modeled CSP in a traditional format (such as Prolog or a Java implementation) lies to the investigator and involves non-trivial conversion and mapping of ontological descriptions and annotations to traditional CSP concepts. The authors then demonstrated a prototype interface for supporting a basic RDFS-based representation, which implements services for bidding agents and provision of video content in response, and expressed their future plans of creating an appropriate toolchain for linking the generated representation to a traditional solver and representing multiple solutions with various overall utilities so that the agent or user can choose a preferred set of commitments.

Other existing work has extended OWL and SWRL in order to specify additional properties which are then translated either in CIF as XML files [Badra et al., 2011] or straight to CLP (such as SweetProlog [Laera and Tamma, 2004] and SWCL [Kim et al., 2003]). These projects, although creating a great initial prototype, provide implementations with very specific focus and are not aimed as generalized frameworks (possibly with optimization capabilities), but only focus on specific problems and systems. Additionally, there has not been any follow-up work in order to provide an easy to use framework or library and thus lack integration capabilities into new research. As previously discussed, there is no standard semantic web technology, let alone a formalism to describe constraints. Even though there are de-facto standards being implemented and used in the semantic web

community, there is still no specific focus on the body of work developed. Extensions are developed and described but few are adequately fine-tuned and standardized (de-facto standardization is not a quick or even possible process and the same applies to official standardization by committees, where multiple requirements are proposed). A very simple testament to this is the inability of using a default value in an OWL description, even though such a feature is available in Common Information Model (CIM) and Silk Link Specification Language (SILK) [Grosof et al., 2009]. These facts are an indication that the creation of a generic and popular framework is a daunting task, and the fact that research focuses on specific platforms or prototype research solutions indicates that further fragmentation and lack of coordinated effort is expected.

4.2 Ontology engineering patterns

Investigators working with CSPs are typically using CLP systems to describe and solve their models. This requires experience and intuition in using FOL and Horn Clauses, which have important differences from DLs (as discussed in section 3.3). Knowledge engineering and its prominent technologies require a paradigm shift in order to be used successfully, especially when enhanced with fuzzy features with are still under extensive research.

Alas, being knowledgeable on the Semantic Web stack technologies is not enough, because practical ontology design experience is required in order to satisfy the design requirements and fully represent the required system. Similar to software engineering, where software development methodologies and knowledge of engineering patterns and anti-patterns are useful tools for increasing software quality, knowledge engineering practices are required for creating a generic, usable ontology that can adequately support the problem model. An efficient problem design practices and re-use of tested solutions are helpful for detecting uncovered requirements, expressiveness or performance inefficiencies, facilitate the addition of new axioms and rules for extending the knowledge base, and make ontology re-use and integration easier for other interested parties (in contrast to providing an proof-of-concept prototype which cannot be reused in different projects). Simple solutions such as “FuzzyOwl2Ontology” by Bobillo and Straccia [2009] are trying to ease the task of generating fuzzy ontologies and introducing new axioms by creating an ontological description of a fuzzy ontology that the user populates with instances representing the axioms and the elements of his design. This description can then be used by custom parsers to generate ontological descriptions for a reasoner of choice. Alas, this procedure

involves translating axioms of different semantics and satisfying possibly divergent specifications, indicating that a common specification and extensible, reusable practices are of significant importance.

Ontology Design Patterns (ODPs) describe a reusable successful solution to recurrent modeling problems, such as describing spatio-temporal data and relations, N-ary relationships, Composite Property Chains etc. There are both online[1] and bibliographic (e.g. Gangemi [2005]) resources describing such practices, some of which are de-facto standards while others being tested proposals under current research or application. Since the early days of the Semantic Web, the need for a formal practice or standard framework for the identification of requirements, their translation to usable components and the final generation of ontologies incorporating them was apparent. These may either provide some standard features to be utilized by investigators in their manual creation of the ontology, or provide tools for (semi-)supervised automatic generation of ontologies from specifications and/or data. An unfortunate and under-researched consequence of using ODPs is that they effectively affect the computational profile of ontologies, possibly compromising language expressibility constraints, degrading performance or making large ontologies practically intractable [Horridge et al., 2012].

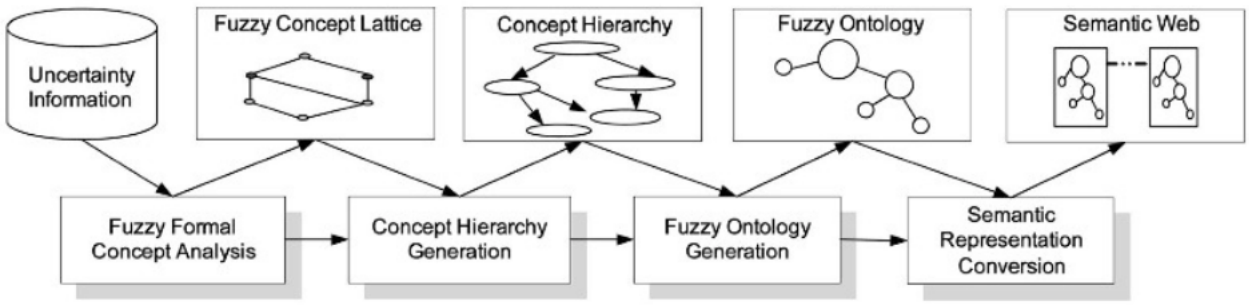

Figure 4.1: From vague data to the Semantic Web via Fuzzy Ontologies. (source: FOGA, Quan et al. [2004])

An initial effort of that kind supporting fuzzy knowledge was the Fuzzy Ontology Generation Framework (FOGA) system. Quan et al. [2004] introduced Fuzzy Formal Concept Analysis (FFCA), a technique of deriving concept hierarchies (in the form of a lattice) from specified objects and their properties [Ganter and Wille, 1999], extended to support fuzzy relations. Input data are linguistically analyzed and ontological information is extracted

[1] Two of the more useful banks of ODPs are `http://ontologydesignpatterns.org/` and `http://odps.sourceforge.net/`

via conceptual clustering, leading to the construction of a concept lattice and the hierarchical relations between concepts. During the application of FFCA, metadata are kept such as membership values of objects in each fuzzy formal concept and similarities of fuzzy formal concepts required for the construction of concept hierarchy. This automated procedure provides the initial hierarchy of the ontology with whichever information can be extracted, and then the investigator may provide more specific names for relations and top concepts; similar work was presented by Zhai [2007] but focused on linguistic values to achieve better expression of generated fuzzy degrees and properties. It was demonstrated that the initial clustering (based on metrics such as *Relaxation Error* and *Average Uninterpolated Precision*) provides good clustering of concepts and is a good first step for the generation of an ontology and the bootstrapping of fuzzy degrees. Wallace et al. [2012] investigated the process of element identification and what tasks are required for ontology development, identifying fundamentally different tasks that may be singled out into two distinct layers, namely conceptualization and formalization. By separating tasks associated with each layer, any conventional ontology engineering methodology may be modified to facilitate efficient collaboration and communication of requirements between domain experts and ontology engineers, thus optimizing the overall process with respect to both effort and quality of results. That knowledge was collected in a comprehensive methodology by Alexopoulos and Wallace [2012] for the development of IKARUS-Onto, a methodology for manually creating fuzzy ontologies based on existing, crisp ones. The authors provide an initial overview of existing methodologies for crisp ontology development, noting that although automatic ontology generation is a helpful first step in the creation of the basic hierarchy, the real difficulty lies in the early and accurate description of relations, fuzzy degrees, the quality of expressing the subjective vagueness of the problem and the ability to create shareable, reusable and intuitive representations for use by other investigators. They identify the concepts of “degree-vagueness” (the difficulty of drawing precise boundaries for the applicability of a property) and “combinatory vagueness” (the difficulty of determining and comparing the partial importance of multiple contributing properties for the definition of a concept) which must be evaluated during the investigation of crisp relations in the initial ontology. The interpretation of those results is the first step in transforming the relations to fuzzy ones with appropriate fuzzy degrees. Afterwards, attributes are evaluated in order to determine the required fuzzy datatypes and their membership functions (effectively determining the fuzzy sets). Using all information until now, and taking into consideration the area of application as well as the datatype and reasoning features of

the reasoning system in use, the fuzzy ontology elements are formalized in the appropriate language. Last, the result is validated for correctness, completeness, accuracy and consistency, in order to determined unfulfilled requirements, controversial information etc. Their findings are summarized in the following figure they provide in their work, and were also demonstrated by a custom toolchain in Wallace et al. [2012].

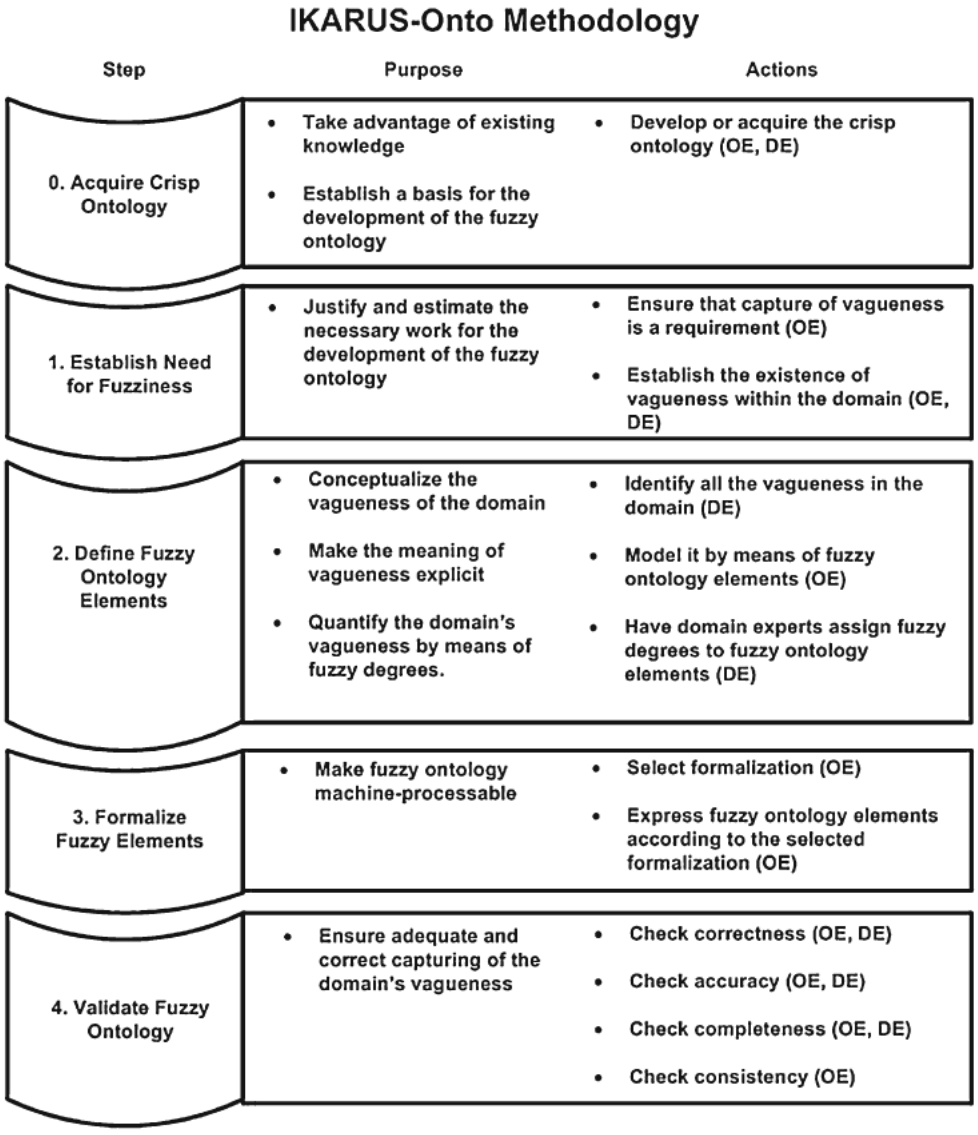

Figure 4.2: A complete methodology for manual fuzzy ontology acquisition, representation and re-use (source: Alexopoulos and Wallace [2012])

5. Conclusions

The semantics of a fuzzy constraint set requires a-priori knowledge in order to determine an appropriate membership function and to define its elements. Even though this limits the researcher's capability for more complete automatic modeling of problems, epistemic knowledge is usually enough in order to determine an adequately satisfying model [Lodwick, 2010]. A lot of research on optimization and constraint satisfaction problems, as well as adequate research on incomplete knowledge formalisms such as fuzzy logic prove to be adequate tools for describing and solving such practical applications.

Concerning the state of reasoning, there have been some methods which exploit specific arithmetic systems, especially real domains where the ranges of the domains may be combined or simplified or simply cleverly exploited for better constraint propagation. Nevertheless, the problem of automatically inferring new constraints and knowledge from a specific problem description is a difficult problem and seems to be a worse "value-for-money" compared to empirical optimization strategies and specific purpose systems and tools. This is especially true for fuzzy problems, where both the semantics, the modeling and the reasoning requirements become even more difficult.

Little work has been done on using ontologies to model constraint satisfaction problems, mainly due to the fragmentation of tools and ontologies for describing such formalisms and the focus of the research community on knowledge integration in the Internet of Things [Leuf, 2005; Gyrard et al., 2014]. Nevertheless, based on its constraint satisfaction roots and the traditional modeling process described in the relevant sections, the need for empirical prototyping and interactive refinement of such models remains important. The interoperability and expressiveness provided by ontologies, combined with interactive methods based on CSP and CLP formalisms, may lead to a tool that promotes experimentation and provides better insight into the specifics of each problem's domain, aiding in the incremental modeling of a fuzzy problem.

Incremental steps may be taken in order to transform basic CSP principles into an initial specific-purpose ontology, which can later be enhanced by domain knowledge in DL semantics and ontology engineering, as well as integration with existing solvers. Setbacks for such an endeavor will be the fragmented nature of the Semantic Web ecosystem and its associated standards and ontologies, the immature/prototype systems currently in use, as well as the lack in the area of visual design and representation of ontologies and relevant tools. Now that the Semantic Web and the Internet of Things are at the top of the hype cycle

[1], expectations and standards are expected to settle while at the same time technology and tools will adequately mature to be production-ready, paving the way for further development and widespread practical applications.

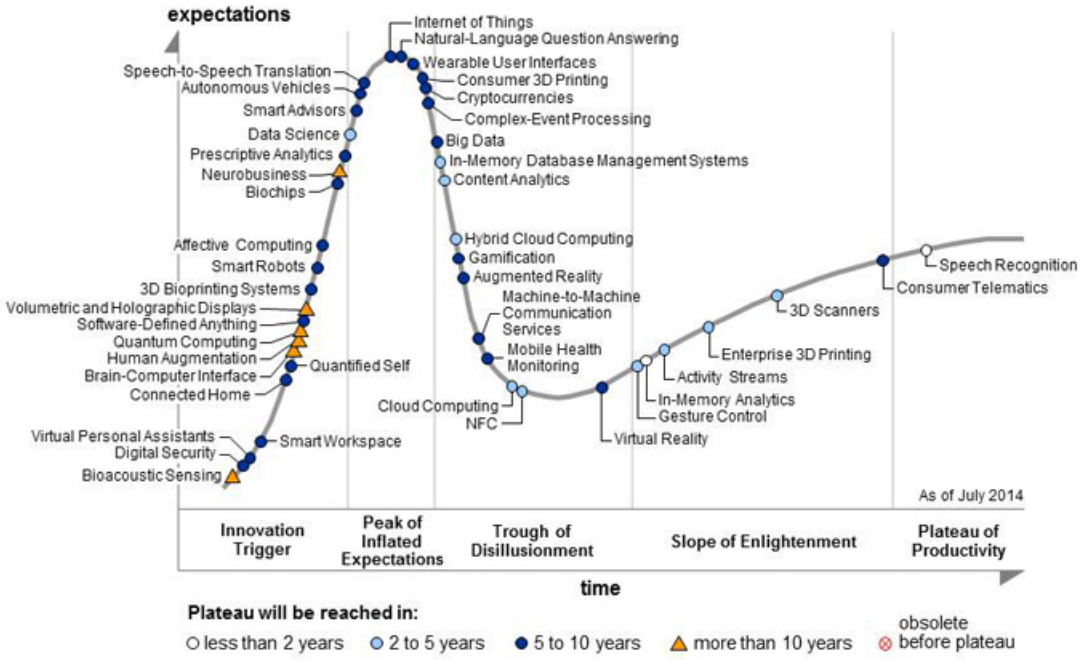

Figure 5.1: The Internet of Things (and thus the Semantic Web) is at the peak of the hype circle. Maturity and focus on practical applications will follow. (source: Gartner)

[1] https://www.gartner.com/newsroom/id/2359715

Glossary

ABox Assertional Box

BL Basic fuzzy propositional logic

BTVB Best Truth-Value Bound

CG Conceptual Graph

CGA Conceptual Graph Assemblies

CBI Constraint-Based Inference

CIF Constraint Interchange Formalism

CIM Common Information Model

CLP Constraint Logic Programming

CSP Constraint Satisfaction Problem

DAML DARPA Agent Markup Language

DCL Descriptive Constraint Library

DCSP Dynamic CSP

DL Description Logic

DLP Description Logic Program

FC Forward Checking

FCSP Flexible CSP

f-CSP Fuzzy CSP

f-DL Fuzzy Description Logic

FFCA Fuzzy Formal Concept Analysis

FL Fuzzy Logic

FOGA Fuzzy Ontology Generation Framework

FOL First-Order Logic

FOP Fuzzy Optimization

GCI General Concept Inclusion axiom

GLB Greatest Lower Bound

ILP Inductive logic programming

IMTL Involutive Monoidal T-Norm based Logic

IoT Internet of Things

KB Knowledge Base

LUB Least Upper Bound

MAC Maintaining Arc-Consistency

MILP Mixed-integer linear programming

MTL Monoidal T-Norm-based propositional fuzzy logic

ODP Ontology Design Pattern

OIL Ontology Inference Layer

OPL Optimization Programming Language

OWL Web Ontology Language

OWL-FC Ontology Web Language for Fuzzy Control

PFCSP Prioritized Fuzzy CSP

R-implication Implication associated with a T-Norm

S-implication Implication associated with a T-Conorm and a strong negation

SILK Silk Link Specification Language

RBox Role Hierarchy

RDF Resource Description Framework

RDFS RDF Schema

RIF Rule Interchange Format

SLD Resolution Selective Linear Definite clause resolution

SLG Resolution SL-resolution extended with tabling

SPARQL SPARQL Protocol and RDF Query Language

SWCL Semantic Web Constraint Language

SWRL Semantic Web Rule Language

T-Norm Triangular norm

T-Conorm Triangular conorm

TBox Terminological Box

URI Uniform Resource Identifier

VO Virtual Organization

WCSP Weighted CSP

W3C World Wide Web Consortium

XML Extensible Markup Language

Bibliography

Akplogan, M. and Dury, J. (2011). A Weighted CSP approach for solving spatio-temporal farm planning problems. *11th Workshop on Preferences and Soft Constraints*.

Alecha, M., Alvez, J., Hermo, M., and Laparra, E. (2009). A New Proposal for Using First-Order Theorem Provers to Reason with OWL DL Ontologies. *Citeseer*, pages 1–10.

Alexopoulos, P. and Wallace, M. (2012). IKARUS-Onto: a methodology to develop fuzzy ontologies from crisp ones. *. . . and information systems*, 32(3):667–695.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Apt, K. (2003). *Principles of Constraint Programming*. Cambridge University Press, Cambridge.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. (2007). *The Description Logic Handbook*. Cambridge University Press, Cambridge.

Baader, F., Horrocks, I., and Sattler, U. (2005). Description Logics as Ontology Languages for the Semantic Web. pages 228–248.

Baader, F. and Penaloza, R. (2011). Are fuzzy description logics with general concept inclusion axioms decidable? *Fuzzy Systems (FUZZ), 2011 IEEE . . .* , pages 1735–1742.

Bacchus, F. and Run, P. V. (1995). Dynamic variable ordering in CSPs. *CP '95 Proceedings of the First International Conference on Principles and Practice of Constraint Programming*.

BaczyE•ski, M. and Jayaram, B. (2008). (S,N)- and R-implications: A state-of-the-art survey. *Fuzzy Sets and Systems*, 159(14):1836–1859.

Badra, F., Servant, F., and Passant, A. (2011). A semantic web representation of a product range specification based on constraint satisfaction problem in the automotive industry. *. . . of the 1st international workshop on . . .* , pages 37–50.

Bartak, R. (1997). A Generalized Framework for Constraint Planning.

Behounek, L. (2010). Feasibility as a gradual notion. *LPAR short papers (Yogyakarta)*, 13(Icc):15–19.

Bellman, R. and Zadeh, L. (1970). Decision-making in a fuzzy environment. *Management science*, 23.

Bessiere, C., Freuder, E., and Regin, J. (1995). Using inference to reduce arc consistency computation. *IJCAI (1)*, (33).

Bobillo, F., Delgado, M., and Gomez-Romero, J. (2008a). DeLorean: A Reasoner for Fuzzy OWL 1.1. *URSW*.

Bobillo, F., Delgado, M., and Gomez-Romero, J. (2008b). *Optimizing the Crisp Representation of the Fuzzy Description Logic SROIQ*.

Bobillo, F. and Straccia, U. (2008). fuzzyDL: An expressive fuzzy description logic reasoner. In *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, number 1, pages 923–930. IEEE.

Bobillo, F. and Straccia, U. (2009). Fuzzy description logics with general t-norms and datatypes. *Fuzzy Sets and Systems*, 160(23):3382–3402.

Bobillo, F. and Straccia, U. (2010). Fuzzy Ontology Representation using OWL 2.

Bobillo, F. and Straccia, U. (2013). General Concept Inclusion Absorptions for Fuzzy Description logics: A First Step. *Description Logics*.

Bock, J. (2008). Parallel computation techniques for ontology reasoning.

Boley, H., Paschke, A., and Shafiq, O. (2010). RuleML 1.0: the overarching specification of web rules. *Lecture Notes in Computer Science*, 6403:162–178.

Bordeaux, L., Hamadi, Y., and Vardi, M. (2007). An analysis of slow convergence in interval propagation. *Principles and Practice of Constraint Programming*.

Borgwardt, S. and Distel, F. (2014). The Limits of Decidability in Fuzzy Description Logics with General Concept Inclusions.

Brown, K. (2003). Soft consistencies for weighted csps. *Proceedings of Soft*.

Carlson, B. and Gupta, V. (1997). Hybrid CC with interval constraints. pages 1–15.

Chang, L. and Mackworth, A. (2005). A generalization of generalized arc consistency: From constraint satisfaction to constraint-based inference.

Croitoru, M. and Compatangelo, E. (2007). Ontology Constraint Satisfaction Problem using Conceptual Graphs. *Specialist Group on Artificial Intelligence (SGAI2006)*.

Danyaro, K. U., Jaafar, J., and Liew, M. S. (2012). Fuzzy OWL-2 Annotation for MetOcean Ontology. In *International Symposium on Agricultural Ontology Service 2012 (AOS-2012)*, pages 148–154. Artificial Intelligence Workshop 2012.

Davis, E. (1987). Constraint Propagation with Interval Labels. *Artificial Intelligence*, 32(3):281–331.

Dechter, R. (1992). From local to global consistency. *Artificial Intelligence*.

Dechter, R. and Dechter, A. (1988). *Belief maintenance in dynamic constraint networks*.

Dentler, K., Cornet, R., Teije, A. T., and Keizer, N. D. (2011). Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web*, 1:1–5.

Dib, M., Abdallah, R., and Caminada, A. (2010). Arc-Consistency in Constraint Satisfaction Problems: A Survey. *2010 Second International Conference on Computational Intelligence, Modelling and Simulation*, pages 291–296.

Ding, Y. (2008). *Tableau-based reasoning for description logics with inverse roles and number restrictions*. PhD thesis.

Dubois, D., Fargier, H., and Prade, H. (1996). Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. *Applied Intelligence*, 6(4):287–309.

Eich, M. (2013). *Marine Vessel Inspection as a Novel Field for Service Robotics: A Contribution to Systems, Control Methods and Semantic Perception Algorithms.* PhD thesis.

Eiter, T., Ianni, G., Krennwallner, T., and Polleres, A. (2008). Rules and ontologies for the semantic web. *Reasoning Web*.

Eiter, T., Lukasiewicz, T., Schindlauer, R., and Tompits, H. (2004). Well-founded semantics for description logic programs in the semantic web.

Epstein, S. and Yun, X. (2010). From Unsolvable to Solvable: An Exploration of Simple Changes. *Abstraction, Reformulation, and Approximation*, pages 20–25.

Faltings, B. (1994). Arc-consistency for continuous variables. *Artificial Intelligence*, 65(2):363–376.

Floudas, C. A. and Pardalos, P. M. (2008). *Encyclopedia of Optimization*. Number τ. 1 in Encyclopedia of Optimization. Springer.

Fodor, J. (2004). Left-continuous t-norms in fuzzy logic: an overview. *Journal of applied sciences at Budapest Tech Hungary*.

Fuller, R. (1998). Fuzzy reasoning and fuzzy optimization.

Gangemi, A. (2005). Ontology design patterns for semantic web content. *The Semantic Webb•"ISWC 2005*, pages 262–276.

Ganter, B. and Wille, R. (1999). *Formal Concept Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Garrido, A. (2010). Fuzzy boolean algebras and Lukasiewicz logic. *Acta Universitatis Apulensis*, (22):101–111.

Gassert, H. (2004). Operators on Fuzzy Sets: Zadeh and Einstein. *diuf.unifr.ch*, (May):1–10.

Glimm, B., Horrocks, I., and Motik, B. (2010). Optimized description logic reasoning via core blocking. *Automated Reasoning*.

Gottwald, S. (2000). Axiomatizations of t-norm based logics: A survey. *Soft Computing*, 4(2):63–67.

Grosof, B., Dean, M., and Kifer, M. (2009). The Silk System: Scalable higher-order defeasible rules. *The International RuleML Symposium on Rule Interchange and Applications 2009*.

Grosof, B., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: Combining logic programs with description logic. *Proceedings of the 12th . . . .*

Guesgen, H. and Philpott, A. (1995). Heuristics for solving fuzzy constraint satisfaction problems. *Artificial Neural Networks and Expert . . .* , pages 132–135.

Guo, H., Liu, M., and Jayaraman, B. (2006). Relaxation on optimization predicates. In *Logic Programming*.

Gyrard, A., Bonnet, C., and Boudaoud, K. (2014). Domain knowledge Interoperability to build the Semantic Web of Things. *w3.org*, (June):25–26.

Haralick, R. and Elliott, G. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artificial intelligence*, 14(3):263–313.

Horridge, M., Aranguren, M., and Mortensen, J. (2012). Ontology Design Pattern Language Expressivity Requirements. *WOP*.

Hosobe, H., Matsuoka, S., and Yonezawa, A. (1996). *Generalized local propagation: A framework for solving constraint hierarchies*, volume 1118 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hui, K., Gray, P., Kemp, G., and Preece, A. (2003). Constraints as mobile specifications in e-commerce applications. *Semantic Issues in E-Commerce Systems*.

Hyvonen, E. (1989). Constraint reasoning based on interval arithmetic. *International Joint Conference on Artificial Intelligence (. . .* , pages 1193–1198.

Jang, M. and Sohn, J. (2004). Bossam: An extended rule engine for OWL inferencing. In Antoniou, G. and Boley, H., editors, *Rules and Rule Markup Languages for the Semantic Web 2004*, volume 3323 of *Lecture Notes in Computer Science*, pages 128–138. Springer.

Jussien, N. (2001). e-Constraints: Explanation-based constraint programming. *CP01 Workshop on User-Interaction in Constraint . . . .*

Jussien, N. and Boizumault, P. (2002). Dynamic backtracking with constraint propagation: application to static and dynamic csps. *Principles and Practice of Constraint Programming (CP2002)*.

Kask, K. (2000). New search heuristics for MAX-CSP. *Principles and Practice of Constraint Programming - CP2000*.

Kearfott, R. B. (1996). Interval computations: Introduction, uses, and resources. *Euromath Bulletin*, pages 1–23.

Kim, W., Lee, M., Hong, J., Wang, T., and Kim, H. (2003). Merging Mathematical Constraint Knowledge with the Semantic Web using a Semantic Web Constraint Language. *iconceptpress.com*, (October 2009).

Kleemann, T. (2006). Towards mobile reasoning. *International Workshop on Description Logics DL06*.

Klement, E. and Navara, M. (1999). A survey on different triangular norm-based fuzzy logics. *Fuzzy Sets and Systems*, (16):1–19.

Konstantopoulos, S. and Apostolikas, G. (2007). Fuzzy-DL reasoning over unknown fuzzy degrees. *On the Move to Meaningful Internet . . .* , pages 1312–1318.

Kumar, V. (1992). Algorithms for constraint-satisfaction problems: A survey. *AI magazine*.

Laera, L. and Tamma, V. (2004). SweetProlog: A system to integrate ontologies and rules. *Rules and Rule Markup . . .* , pages 2–7.

Lecoutre, C., Sais, L., Tabary, S., and Vidal, V. (2006). Last conflict based reasoning. *European Conference on Artificial Intelligence*.

Leuf, B. (2005). *The Semantic Web: Crafting Infrastructure for Agency*. John Wiley & Sons, Ltd, Chichester, UK.

Levy, J., Ansótegui, C., and Bonet, M. (2007). The logic behind weighted CSP. *Trends in Constraint . . .* , pages 32–37.

Lhomme, O. (1993). Consistency techniques for numeric CSPs. *IJCAI*, pages 232–238.

Liu, C., Qi, G., Wang, H., and Yu, Y. (2011). Fuzzy Reasoning over RDF Data Using OWL Vocabulary. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pages 162–169. IEEE.

Liu, C., Qi, G., Wang, H., and Yu, Y. (2012). Reasoning with Large Scale Ontologies in Fuzzy EL+ Using MapReduce. *IEEE Computational Intelligence Magazine*, 7(2):54–66.

Lodwick, W. (2010). Interval Analysis, Fuzzy Set Theory and Possibility Theory in Optimization. pages 1–68.

Lodwick, W. A. and Kacprzyk, J. (2010). *Fuzzy Optimization*, volume 254 of *Studies in Fuzziness and Soft Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Loia, V., De Maio, C., Fenza, G., and Senatore, S. (2010). OWL-FC Ontology Web Language for fuzzy control. In *International Conference on Fuzzy Systems*, pages 1–8. IEEE.

Lukasiewicz, J. (1930). Philosophical remarks on many-valued systems of propositional logic.

Lukasiewicz, T. and Straccia, U. (2008). Managing uncertainty and vagueness in description logics for the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):291–308.

Luo, X., Lee, H., Leung, H.-f., and Jennings, N. (2003). Prioritised fuzzy constraint satisfaction problems: axioms, instantiation and validation. *Fuzzy sets and systems*, 136(2):151–188.

Madsen, J. (2003). Methods for interactive constraint satisfaction. *Master's thesis, Department of Computer Science, ...*.

Mas, S., Wang, F., and Reinhardt, W. (2005). Using ontologies for integrity constraint definition. *Proceedings of the 4th international ...*.

Mazeika, A., Jaulin, L., and Osswald, C. (2007). A new approach for computing with fuzzy sets using interval analysis. *Information Fusion, 2007 10th ...*, pages 1–8.

Meech, A. (2010). Business Rules Using OWL and SWRL. *Advanced in Semantic Computing*, 2:23–31.

Miguel, I. (2001). The case for dynamic flexible constraint satisfaction.

Miguel, I. (2003). Fuzzy rrDFCSP and planning. *Artificial Intelligence*, 148(1-2):11–52.

Moore, R. E. (1966). *Interval analysis*. Prentice-Hall series in automatic computation. Prentice-Hall.

Motik, B., Shearer, R., and Horrocks, I. (2007). Optimized reasoning in description logics using hypertableaux. *Automated Deduction - CADE-21*.

Mouhoub, M. and Sukpan, A. (2012). Managing dynamic CSPs with preferences. *Applied Intelligence*, 37(3):446–462.

Nacer, K. and Jilani, K. (2014). New Fuzzy CSP for an Optimized Mobile Robot's Path Tracking using Genetic Algorithms. *ijcit.com*, 03(03):523–531.

Noguera, C., Esteva, F., and Godo, L. (2010). Generalized continuous and left-continuous t-norms arising from algebraic semantics for fuzzy logics. *Information Sciences*, pages 1–33.

Novak, V., Perfilieva, I., and Mockor, J. (1999). *Mathematical principles of fuzzy logic*.

Parsia, B., Sirin, E., Grau, B. C., Ruckhaus, E., and Hewlett, D. (2005). Cautiously approaching SWRL. Technical Report February 2005, University of Maryland.

Patel-Schneider, P., Hayes, P., and Horrocks, I. (2004). OWL web ontology language semantics and abstract syntax. –W3C˝ recommendation, W3C.

Pendharkar, P. C. (2006). A data mining-constraint satisfaction optimization problem for cost effective classification. *Computers & Operations Research*, 33(11):3124–3135.

Preece, A., Chalmers, S., and McKenzie, C. (2008). A semantic web approach to handling soft constraints in virtual organisations. *. . . Commerce Research and . . .*, page 151.

Preece, A., Hui, K., Gray, A., Marti, P., Bench-Capon, T., Jones, D., and Cui, Z. (2000). The KRAFT architecture for knowledge fusion and transformation. *Knowledge-Based Systems*, 13(2-3):113–120.

Qi, G., Liu, W., and Glass, D. (2004). Combining individually inconsistent prioritized knowledge bases. *NMR*.

Quan, T., Hui, S., and Cao, T. (2004). FOGA: a fuzzy ontology generation framework for scholarly semantic web. *Proceedings of the Knowledge Discovery and . . .*.

Ramik, J. (2001). Soft computing: overview and recent developments in fuzzy optimization.

Ren, Y., Pan, J., and Lee, K. (2012). Optimising Parallel ABox Reasoning of EL Ontologies. *25th International Workshop on Description Logics*.

Rudolph, S. and Hitzler, P. (2008). Description Logic Rules. (Dl):1–23.

Ruta, M., Scioscia, F., and Sciascio, E. D. (2010). Mobile Semantic-based Matchmaking: a fuzzy DL approach. *Lecture Notes in Computer Science*, 6088:16–30.

Schiex, T. (1992). Possibilistic constraint satisfaction problems or how to handle soft constraints? *Proceedings of the Eighth international conference on . . .* , pages 269–275.

Simou, N., Athanasiadis, T., and Kollias, S. (2008). An architecture for multimedia analysis and retrieval based on fuzzy description logics. *2nd K-Space PhD Students Workshop*.

Simou, N. and Kollias, S. (2007). FiRE: A fuzzy reasoning engine for impecise knowledge. *K-Space PhD Students Workshop, Berlin, Germany*, pages 1–2.

Simou, N. and Mailis, T. (2010). Optimization techniques for fuzzy description logics. *. . . on description logics . . . .*

Sirin, E., Grau, B., and Parsia, B. (2006). From Wine to Water: Optimizing Description Logic Reasoning for Nominals. *KR*, pages 90–99.

Sirin, E., Parsia, B., and Grau, B. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: science, . . .* , 5(2):51–53.

Staab, S., Horrocks, I., Angele, J., Decker, S., Kifer, M., Grosof, B., and Wagner, G. (2003). Trends & controversies - Where are the rules? *IEEE Intelligent Systems*, 18(5):76–83.

Staab, S. and Maedche, A. (2000). Axioms are objects, too: Ontology engineering beyond the modeling of concepts and relations. pages 1–16.

Steigmiller, A., Glimm, B., and Liebig, T. (2014). Optimised absorption for expressive description logics. *Workshop on Description Logics (DL'14)*.

Stoilos, G., Stamou, G., and Pan, J. (2006a). Handling imprecise knowledge with fuzzy description logic. *. . . Internat. Workshop on Description Logics (. . . .*

Stoilos, G., Stamou, G., Pan, J. Z., Tzouvaras, V., and Horrocks, I. (2007). Reasoning with Very Expressive Fuzzy Description Logics. 30:273–320.

Stoilos, G., Straccia, U., Stamou, G., and Pan, J. (2006b). General Concept Inclusions in Fuzzy Description Logics. In *Proceedings of the 17th Eureopean Conference on Artificial Intelligence (ECAI-06)*, pages 457—-461. IOS Press.

Straccia, U. (1998). A fuzzy description logic. *AAAI/IAAI*.

Straccia, U. (2004). Transforming fuzzy description logics into classical description logics. *Logics in Artificial Intelligence*, pages 1–13.

Straccia, U. (2011). Reasoning within Fuzzy Description Logics. *Journal of Artificial Intelligence Research*, 14:137–166.

TANG, J. F., WANG, D. W., FUNG, R. Y. K., and Yung, K.-L. (2004). Understanding of fuzzy optimization: theories and methods. *Journal of Systems Science and Complexity*, 17(1).

Tick, J. and Fodor, J. (2005). Fuzzy implications and inference processes. *ICCC 2005 - IEEE 3rd International Conference on Computational Cybernetics*, 24:591–602.

Tsarkov, D. and Horrocks, I. (2006). FaCT++ description logic reasoner: System description. *Automated reasoning*.

Tsatsou, D., Dasiopoulou, S., Kompatsiaris, I., and Mezaris, V. (2014). LiFR: A Lightweight Fuzzy DL Reasoner. *2014.eswc-conferences.org*, pages 5–8.

Tsetsos, V., Anagnostopoulos, C., and Hadjiefthymiades, S. (2006). On the Evaluation of Semantic Web Service Matchmaking Systems. *2006 European Conference on Web Services (ECOWS'06)*, pages 255–264.

Urbani, J. (2010). Scalable and parallel reasoning in the Semantic Web. *The Semantic Web: Research and Applications*.

Vanekova, V., Bella, J., Gursky, P., and Horvath, T. (2005). Fuzzy RDF in the semantic web: Deduction and induction. *Proceedings of Workshop on Data Analysis (WDA 2005)*.

Verfaillie, G. and Schiex, T. (1994). Solution reuse in dynamic constraint satisfaction problems. *AAAI*.

Vojtas, P. (2005). Fuzzy logic as an optimization task.

Wallace, M., Alexopoulos, P., and Mylonas, P. (2012). Identifying conceptual layers in the ontology development process. *Lecture Notes in Artificial Intelligence*, 7297:375–382.

Wallace, R., Grimes, D., and Freuder, E. (2009). Dynamic Constraint Satisfaction Problems: Relations between Search Strategies and Variability in Performance. *4c.ucc.ie*.

Wallace, R., Nordlander, T., and Dokas, I. (2007). Ontologies as Contexts for Constraint-Based Reasoning. *Contexts and Ontologies Representation and Reasoning*, 81:1–2.

Wang, W., De, S., Cassar, G., and Moessner, K. (2013). Knowledge Representation in the Internet of Things: Semantic Modelling and its Applications. *Automatika Journal for Control, Measurement, Electronics, Computing and Communications*, 54(4):388–400.

Wlodarczyk, T. and Rong, C. (2011). SWRL-F: a fuzzy logic extension of the semantic web rule language. *Proceedings of the International Conference on Web Intelligence, Mining and Semantics (WIMS11)*, pages 1–4.

Xu, B., Li, Y., Lu, J., and Kang, D. (2006). Secure OWL Query. In *Proceedings of the 6th International Conference on Computational Science - Volume Part IV*, ICCS'06, pages 95–103, Berlin, Heidelberg. Springer-Verlag.

Yaguinuma, C. A., Santos, M. T. P., Camargo, H. A., Nicoletti, M. C., Nogueira, T. M., and Paulista, C. L. (2014). A meta-ontology for modeling fuzzy ontologies and its use in classification tasks based on fuzzy rules. *International Journal of Computer Information Systems and Industrial Management Applications*, 6:89–101.

Zadeh, L. (1965). Fuzzy sets. *Information and control*.

Zadeh, L. (1968). Fuzzy algorithms. *Information and control*, 102:94–102.

Zadeh, L. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1):3–28.

Zhai, J. (2007). Fuzzy Ontology Model for Knowledge Management. In *Proceedings on Intelligent Systems and Knowledge Engineering (ISKE2007)*, Paris, France. Atlantis Press.

Zimmermann, H. J. (2001). *Fuzzy Set Theory and Its Applications*. Springer Netherlands.