KNOWLEDGE SHARING AMONG AGENTS VIA UNIFORM INTERPOLATION

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2023

David Toluhi

Department of Computer Science

Contents

Word Count: 55,374

List of Tables

List of Figures

Abstract

KNOWLEDGE SHARING AMONG AGENTS VIA UNIFORM INTERPOLATION

David Toluhi

A thesis submitted to The University of Manchester for the degree of Doctor of Philosophy, 2023

In our age of big data and the knowledge society, effective processing and sharing of knowledge is crucial. Agents provide a key abstraction within modern development of artificial intelligence systems and software. In current multi-agent platforms a simplifying assumption is that all agents use the same symbols to represent their knowledge or expertise. The assumption that agents use the same symbols to represent their knowledge imposes a constraint on system developers and agent systems: agents are forced to use the same ontological representation of the world at the cost of having diverse and unique viewpoints. Furthermore, agents typically exist in dynamic environments that require update and revision of their knowledge and beliefs.

We focus on agents that make use of description logic ontologies to represent their knowledge and expertise. Ontologies are knowledge bases consisting of logical statements called *axioms*. This comes with the advantage that a set of logical axioms can entail knowledge that is not explicitly stated by the axioms: lots of information can be implicit. As a consequence, when communicating, agents must take into account the implicit knowledge contained in their expertise. Assuming a common signature is established between two communicating agents, the agents still require methods to extract specific knowledge from their ontologies that go beyond sending a list of axioms. Agents require methods to extract both explicit and implicit knowledge from their ontologies in a way that can be communicated to another agent if the need arises. This thesis re-uses several existing techniques in the logic literature, mainly *uniform interpolation*, that can be used to extract knowledge from ontologies and can be adapted for agent communication.

In particular, our aim is to develop and evaluate novel algorithms which provide support for scenarios where multiple agents are responsible for different knowledge

bases (e.g., that capture the agent's different expertise) and have the ability to restrict (specifically on the basis of uniform interpolation) and adapt their knowledge with respect to the signature of their knowledge base that is shared with other agents. This ensures that knowledge shared is understood by the communication partners.

We have realised and developed a range of knowledge extraction methods and structures that enable agents to extract knowledge that can sometimes be implicit with respect to arbitrary subset signatures of their knowledge base. We have analysed and evaluated these knowledge extraction methods on $\mathcal{ALC}$ description logic ontologies. Our results suggest that the knowledge extraction methods realised are feasible in some practical settings; unlike the knowledge extraction methods realised, the knowledge extraction methods developed are rarely feasible and useful in practical settings, and will require further exploration and research to be useful in practical settings.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

First of all, I would like to thank my parents and my family for encouraging me to pursue a PhD who were able to recognise my affinity for research and spotted it before I did (specifically my mum), and specifically my father for paying my tuition fees, this thesis would not exist without you all. I would also like to thank my examiners Uli Sattler and Terry Payne for their valuable suggestions on how to improve my thesis.

I must thank my supervisors Bijan Parsia and Renate Schmidt for their continuous guidance and support throughout the project; you have both improved my mathematical skills, my critical thinking skills, my writing skills, and I am eternally grateful to the both of you.

I would also like to thank Uli and Bijan for their advice, encouragement and support throughout the PhD. They have truly been a primary source of support and encouragement through all my experiences at the University of Manchester (including during my masters degree); they are the unsung heroes of postgrad students in the department. I want to thank Simon Harper for giving me good guidance on how to approach my career in research and being sympathetic throughout my experience. I also have to thank all my colleagues who have supported me all throughout this journey, including Ruba, Warren, Sen, Mostafa, Hauruo, Ghadah and Martina, with special thanks to Ruba who helped me to understand uniform interpolation and was very supportive throughout the journey, she is a friend in the truest sense of the word. Dr Patrick Koopmann must also be acknowledged for his incredible kindness in making the time to meet with me and discuss uniform interpolation during a few periods of struggle.

I also have to thank my family members and friends in the UK that have provided me with support outside academic life and made the isolation of a PhD bearable, Grandma Rebecca, Hannah, Jemimah, Patience, Comfort, Seun, Kanyinsola, Gbenga, Khaldon, and Faith.

Chapter 1

Introduction

Generally speaking, agents are autonomous entities acting within an environment that are seeking to achieve a desired goal or state. To accomplish its desired goal or state, an agent requires some ways to perceive its environment as well as an internal representation of its environment. There may be several agents in an environment collaborating together in order to achieve their individual or shared goals or states. We call a set of multiple agents in an environment a multi-agent system MAS(s).

One of the appealing characteristics of agents and MAS(s) is that each agent can provide distinct services and likewise have distinct knowledge about the world. This avoids the need for massive upfront coordination but introduces fundamental communication challenges including the determination of common knowledge and potentially increasing amount of knowledge shared between any set of agents (i.e., online data integration).

We are interested in agents that have expressive and rich representations of their knowledge. There are several ways agents can represent their knowledge; however, this thesis is concerned with agents that represent their knowledge using a logical formalism. We call any agent that uses a logic formalism to represent its knowledge a logic-based agent. Intuitively speaking, a logical formalism facilitates a representation of a set of facts about a domain in such a way that allows for deductions to be made from those set of facts. A set of facts represented using a logical formalism is usually referred to as an *ontology* or a *knowledge base*. In this thesis, we are interested in agents that use the description logic $\mathcal{ALC}$ to represent their knowledge.

Some of the knowledge agents possess may be *implicit* due to the rich and expressive nature of the logic formalism used in representing their knowledge. Thus agents may require a means with which to extract *implicit* knowledge to reason about their

environment or convey and communicate effectively with and to other agents. As an illustrative example, consider that humans can be in some ways analogous to agents. For example, the staff of a restaurant: waiters, bartenders, chefs, and managers could be considered agents with the shared goal of preparing food and drink to serve paying customers. Each category of restaurant staff has their own specialised knowledge: waiters have people and presentation skills, bartenders have knowledge about drinks and cocktails, and chefs have knowledge on recipes and preparing food. In order to achieve their shared goal of preparing and serving food to paying customers, all staff members must communicate effectively and be able to convey their specialised knowledge using words and language that is understood by staff members if the situation requires it. The following exchange is an example of a situation that requires staff members to convey specialised knowledge:

- **chef:** *"Make sure you **acidulate** the chicken before serving it to the customer"*
- **waiter:** *"What does **acidulate** mean?"*

At this point, there are several ways the chef can respond to the waiter to *convey* what it means to acidulate. The chef can either:

1. Show the waiter an *example* of acidulation.
2. *Define* or *describe* what it means to acidulate.
3. Give some *context* about acidulation and its effects on foods so that the waiter has an intuition on what kind of foods may require acidulation.

This list is not exhaustive, but mainly serves to illustrate that in the last two approaches, to convey the meaning of 'acidulate', the chef must extract some of their *implicit* knowledge on the topic of acidulation. Furthermore, it highlights that there are forms (highlighted in italised font) of representing the knowledge conveyed to the waiter. We refer to such forms as *knowledge structures*. In this example, the task that requires extracting implicit knowledge arises when one of the human agents (the chef) is attempting to convey the meaning of a word that another human agent (the waiter) does not know. However, there are scenarios which require the extraction of implicit knowledge even if all the words are understood by the communicating participants. Again consider the following exchange:

- **chef:** *"Make sure you **acidulate** the chicken before serving it to the customer"*

- **waiter:** *"Why would we pour acid on the chicken?"*
- **chef:** *"I'm not asking you to do that. In the kithcen,* ***acidulate*** *refers to making something sour by using lemon or lime juice."*

In this exchange, both human agents have internal but different meanings of the word *acidulate*, but are required to extract what their individual meanings to resolve a misunderstanding in communication. Observe that the waiter knows the meaning of the word *acidulate* in a general sense; however, the chef has a specialised interpretation of the word. The knowledge structure (underlined in the exchange) used by the chef to convey their interpretation of *acidulation* is a definition.

In logic, there are several knowledge structures and extraction methods exist. **This thesis is concerned with exploring how knowledge structures and extraction methods in $\mathcal{ALC}$ can be used by logic-based agents to extract implicit knowledge that facilitates communication**. We study, realise, and evaluate knowledge structures and extraction techniques for $\mathcal{ALC}$.

1.1 Scope of Research

There are several topics of interest in this thesis, namely, agent systems, agent communication, and logic-based knowledge extraction. However, in this thesis, we are concerned with specific subsets of the main topics of interest which we highlight in this section.

Agent Systems

This research focuses on agents that may represent their knowledge and communicate using logic-based formalisms, specifically description logics. We call such agent systems *logic-based* agent systems and agents that exist within such systems *logic-based* agents.

A common assumption in logic-based agent systems that adopt description logics [LBBH06, KHA06] to represent the knowledge of the agents is that all agents share the vocabulary used to construct each individual agent's knowledge. There are a few agent systems such as ANEMONE [vDBD$^+$06] and Cool-AgentSpeak [VDRA11] that do not hold this assumption. We investigate how knowledge sharing can be facilitated with respect to a shared vocabulary when this assumption is relaxed.

Knowledge extraction

There are several notable knowledge extraction methods in the context for $\mathcal{ALC}$. In Chapter 2, we discuss several knowledge extraction methods that have an emphasis on restricting the knowledge in an ontology with respect to a subset vocabulary of the ontology. The idea is to highlight why some of these tools may or may not be appropriate for adoption to knowledge sharing among agents.

Ontology alignment

An ontology alignment is a set of correspondences describing the relationships among symbols that exists in two distinct ontologies. Ontology alignments are often proposed as a method for establishing a common vocabulary among communicating agents. In this thesis, we assume that a common vocabulary for communication between agents is established prior to communication; however, we also investigate how the knowledge extraction methods investigated and developed in this thesis may be applicable to evaluating ontology alignments (specifically in Chapter 8).

1.2 Research Questions

The primary research question this thesis explores is:

> **How can knowledge extraction techniques be applied to facilitate communication among agents that only have a subset of their respective vocabularies in common?**

For a given knowledge structure ε exchanged between communicating agents, there are three main possibilities of ε:

1. ε only includes the words from the vocabulary that are common to the agents, thus minimal-to-zero misunderstanding occurs.

2. ε partially includes words from the vocabulary that are common to the agents and thus a misunderstanding is likely to occur.

3. ε only includes words or expressions that one of the agents knows nothing about and thus a misunderstanding occurs.

In the second and third possibilities, the agent conveying ε must find a way to convey any uncommon word or expression that occurs in ε with respect to the common words if it wants the other agent to understand ε.

Formally speaking, given two agents, AG_1 with domain expertise encoded in ontology O_1, and AG_2 with domain expertise encoded in O_2 where Σ denotes the overlap of the vocabularies of O_1 and O_2, we are interested in how agents can describe (using only words in Σ) words or expressions whose vocabularies fall outside Σ to each other.

We refer to Σ as the *common vocabulary* Σ between the agents. Let X denote a

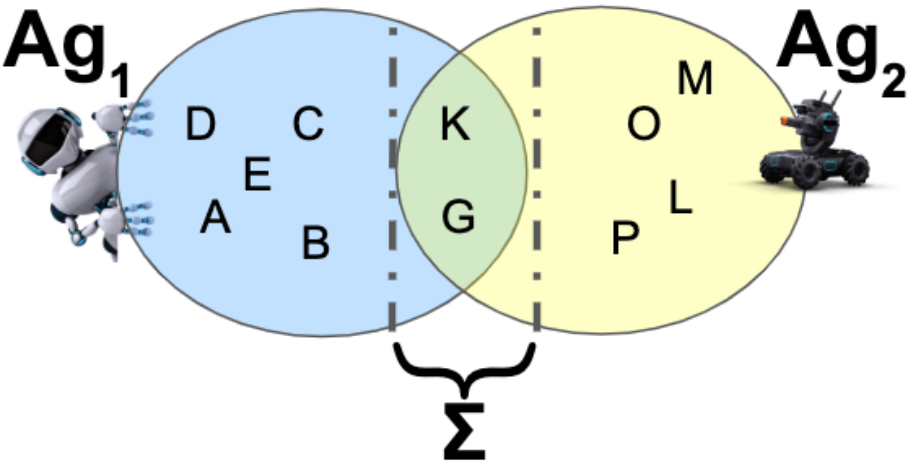

Figure 1.1: An overlap between agent ontology vocabularies. Σ denotes the overlap.

symbol not in Σ that an agent wishes to convey, it must find some knowledge structure Y that describes X such that the vocabulary of Y is a subset of Σ. Desirable properties that the knowledge structure Y should have include:

1. The vocabulary of Y should be a subset of Σ to ensure no new misunderstandings are introduced during communication.

2. Y should be a maximally precise description of X with respect to the common vocabulary. This is to minimise any information loss that may result from conveying a description of X as opposed to X itself.

3. Y should be as concise as possible to minimise communication latency among the agents.

4. Y should be efficient to compute to minimise communication latency among the agents.

There are complexities regarding aspects of the primary research questions which are research questions in their own right. For example, regarding "*knowledge extraction techniques*", it must be noted there are several knowledge extraction techniques that exist in DL which begs the question:

> **RQ1:** What are the various knowledge extraction techniques in $\mathcal{ALC}$ and which of them are relevant to the problem of conveying words or expressions with respect to a restricted vocabulary?

Several research questions follow from **RQ1**:

> **RQ2:** How do the relevant knowledge extraction techniques perform in practice on real-world ontologies?

> **RQ3:** What are the factors that affect the performance of the relevant knowledge extraction techniques?

Research questions **RQ2** and **RQ3** are explored by accounting for the computational complexities of the various knowledge extraction techniques explored and through empirical evaluation.

The primary research question also talks about "*facilitating communication among agents*". This invites the next research question, which is:

> **RQ4:** How can the relevant knowledge extraction techniques be integrated into existing MAS frameworks or agent communication protocols?

Without the proper integration of relevant knowledge extraction techniques, communication can not be facilitated. The last part of the primary research question is concerned with facilitating communication with respect to the vocabulary of the communicating agents that is common, specifically conveying uncommon words with respect to the common words. In order to minimise information loss during such communication (conveying uncommon words with respect to the common words), a further research question needs to be answered:

> **RQ5:** What is the most precise way to convey a word or expression with respect to a subset vocabulary of an ontology?

Furthermore:

> **RQ6:** What are the considerations on the common vocabulary that should be accounted for when conveying words or expressions?

Establishing a common vocabulary also facilitates communication among agents, as such, an additional research question is:

> **RQ7:** Can knowledge extraction techniques be applied to establishing a common vocabulary?

1.3 Challenges and Contributions

This thesis fills in several gaps in the research of knowledge sharing among agents, knowledge extraction in description logic ontologies, ontology alignment, and definition extraction in the context of description logics. We study existing logic-based agent systems, specifically agent systems that incorporate $\mathcal{ALC}$ ontologies in some manner, and highlight the gaps of these agent systems that our research aims to address, specifically, communication under the assumption that agents share their entire, respective ontology vocabularies, in other words, communication with respect to limited or subset vocabularies. We study and discuss existing knowledge extraction techniques in logic that can be potentially applied to facilitate communication among agents in the context of the assumption that they do not share their entire vocabularies. Our study demonstrates that while there are several existing knowledge extraction tools such as uniform interpolation and module extraction that have the potential to facilitate communication with respect to a limited vocabulary, there still lies several gaps in the way of successfully adopting existing knowledge extraction tools to facilitate communication with respect to a limited vocabulary. Specifically, existing knowledge extraction tools such as uniform interpolation and module extraction are not designed for extracting descriptions about specific concepts or symbols from ontologies with respect to limited vocabularies. Our research fills this gap by investigating and developing tools (specifically strongest necessary conditions and weakest sufficient conditions) for extracting concept descriptions about symbols in ontologies that are based on uniform interpolation. We empirically evaluate the developed tools using existing ontologies to test their feasibility in the real world for agent communication applications: even in the context of other logic formalisms (such as propositional logic and first order logic), some of the knowledge extraction tools discussed lack empirical evaluations for agent communication applications. Furthermore, we investigate applications of the developed tools in the context of ontology alignments, as ontology alignments are frequently proposed in the literature as the interface for establishing a common vocabulary among communicating agents.

Overall, the primary contributions of the thesis can be summarised as follows:

- A formal description of strongest necessary conditions and weakest sufficient conditions in description logics, alongside appropriate algorithms and proofs for extraction of the conditions (for $\mathcal{ALC}$).
- The introduction of compiled superconcepts and compiled subconcepts which account for potential shortcomings of strongest necessary conditions and weakest sufficient conditions.
- A novel algorithm for extracting approximate domains and ranges of roles in $\mathcal{ALC}$ ontologies.
- An empirical analysis of the developed algorithms.
- A theoretical and empirical investigation of *definitions* and *definability* in the context of $\mathcal{ALC}$ ontologies.
- An empirical analysis and investigation of the impact of the developed algorithms for extracting strongest necessary conditions and weakest sufficient conditions to agent communication in an existing multi-agent system framework.
- An analysis of the role of uniform interpolation and the developed algorithms to improving the quality of ontology alignments.

1.3.1 Published work

1. Some of the background exploration on Craig interpolation and definitions discussed in Chapter 2 were first presented at the Automated Reasoning workshop in 2019 [TS19].
2. Some of the background exploration on ontology based agent systems discussed in Chapter 3 were first presented at the Logical Aspects in Multi-Agent Systems and Strategic Reasoning workshop in 2021 [TS21].
3. Some of the theoretical results on strongest necessary and weakest sufficient conditions in Chapter 5 were first presented at the International Workshop on Engineering Multi-Agent Systems in 2021 [TSP22].

1.4 Overview of the Thesis

Chapter 2 studies and discusses existing knowledge extraction tools with existing implementations including uniform interpolation, module extraction, strongest necessary conditions, and weakest sufficient conditions. In Chapter 3 we introduce and discuss existing logic-based multi-agent systems and existing related work about agents communicating with respect to overlapping but distinct vocabularies. Chapter 4 provides additional preliminaries and overview on the thesis that takes into account the discussions in Chapters 3 and 2.

The primary theoretical contributions of this thesis are outlined in Chapter 5 where we provide adaptations of strongest necessary conditions and weakest sufficient conditions to the description logic $\mathcal{ALC}$. Furthermore, we develop other methods of knowledge extraction based on uniform interpolation. Chapter 6 empirically evaluates most of the algorithms developed in Chapter 5. In Chapter 7 we discuss how the tools developed in Chapter 5 may be adopted into an existing multi-agent system. We also empirically analyse the impact of adopting the knowledge extraction tools developed in Chapter 5 into an existing multi-agent system. Chapter 8 investigates applications of the tools developed in Chapter 5 to ontology alignment. Chapter 9 concludes the thesis and discusses the limitations and further applications of the thesis.

Chapter 2

Background on Logic and Knowledge Extraction

This chapter introduces relevant definitions on logic and descriptions of existing knowledge extraction and projection techniques in logic, thus exploring **RQ1** ("*What are the various knowledge extraction techniques in DL and which of them are relevant to the problem of conveying terms with respect to a restricted vocabulary?*"). However, we assume that the reader is familiar with the basic definitions and semantics of propositional logic, first-order logic (FOL), and second-order logic (SOL).

In Section 2.1, we introduce the basic semantics of Description Logic (DL) languages used to formalise the standardized web ontology language (OWL). Section 2.2 introduces several well-established knowledge extraction and projection techniques for DL, first order logic, second order logic and propositional logic.

Logical formalisms may be used by agents in order to reason and represent expressive and rich knowledge. In fact, the development of OWL has some history of being driven by agent based applications. The predecessor to OWL was the combination of two frameworks: (1) DARPA Agent Markup Language (DAML) and (2) Ontology Inference Layer (OIL) commonly referred to as DAML+OIL [MFHS02]. DAML was intended to enable software agents to consume content on the web [HM+00]. DAML+OIL were part of several projects driven by Defense Advanced Research Projects Agency (DARPA) Knowledge Sharing Effort. The knowledge sharing effort included research into how agent's knowledge could be formulated in a fashion that agents could parse (these included efforts such as DAML, OIL, and the Knowledge Interchange Format (KIF) [GF+92]), and how agents could communicate (these included efforts such

as the Knowledge Query Manipulation Language (KQML) [FFMM94]). KIF is a language designed for the exchange of knowledge among disparate computer systems including agents. It provides constructs for the expression of statements or axioms in first order logic as well as some extensions to support nonmonotonic reasoning and the representation of entities lists and functions. KIF was proposed as the message format for the agent communication language KQML [FFMM94]. Given that KIF allows for the expressions of statements in FOL, there is no guarantee on how messages sent using KIF can be interpreted, validated, or integrated into an existing agent's knowledge as FOL is undecidable. However, note that since DL(s) are decidable fragments of FOL, they form a less complex subset of KIF that may be used to facilitate communication among agents.

The high computational complexity of KIF hindered its progress and adoption [Ram92], and the DARPA knowledge sharing effort eventually converged its efforts to focus on DAML+OIL which later evolved into the web ontology language OWL which is based on description logics.

2.1 Description Logics

Description logics are languages that are decidable fragments of first order logic used to describe knowledge about a domain of interest. OWL [MGH+09] is a World Wide Web Consortium (W3C) standard that standardises language constructs of description logics for use in academic and industrial environments and has a programmable implementation called the OWL API [HB11] (the version of OWL API used throughout the thesis is 5.1.7).

A description logic is characterised by a set of *language constructs* that determine the expressivity of description logic. The constructs of a description logic define the semantics over any arbitrary set of symbols that are split into three main categories:

- N_C**:** a set of atomic concepts, also called concept names, which correspond to unary predicates. We usually denote atomic concepts using the symbols A and B unless otherwise stated.

- N_R**:** a set of atomic roles, also called role names, which correspond to binary predicates. We usually denote atomic roles using the symbols r and s unless otherwise stated.

- N_I**:** a set of individuals, also called instances, which correspond to constants in first order logic. In this thesis, we usually denote individuals using the symbols a and b unless otherwise stated.

For a description logic $\mathcal{L}$, the semantics of $\mathcal{L}$ are defined using an interpretation I of N_C, N_R, and N_I. I is a pair $\langle \Delta^I, \cdot^I \rangle$, where Δ^I is a non-empty set representing an interpretation domain, and $\cdot^I$ is an interpretation function that maps every atomic concept $A \in N_C$ to a subset A^I of Δ^I; every $r \in N_R$ to a binary relation r^I over Δ^I; every individual a in N_I to an element $a^I \in \Delta^I$. Given an interpretation I and a signature Σ, $I|_\Sigma$ denotes the restriction of I to the symbols in Σ.

An ontology is a knowledge base expressed using a description logic. An ontology consists of at least one of two components:

1. **The TBox:** 'TBox' is short for "terminological box". The TBox is a set of axioms that describe the terminology that defines the vocabulary of the domain of interests by specifying how concepts are related to each other.

2. **The ABox:** 'ABox' is short for "assertional box". The ABox is a set of assertions about named individuals in terms of $N_C \cup N_R$. The ABox may be viewed as an instantiation of the TBox and is composed of concrete data.

Table 2.1 highlights the constructs present in most description logics and the corresponding name of description logic (in the column titled 'Language') associated with specific subsets of the constructs.

Language	Construct	Syntax	Semantics
$\mathcal{EL}$	Top	$\top$	Δ^I
	Intersection	$C \sqcap D$	$C^I \cap D^I$
	Existential quantification	$\exists r.C$	$\{a \in \Delta^I \mid \exists b.(a,b) \in r^I \wedge b \in C^I\}$
$\mathcal{AL}$	Top	$\top$	Δ^I
	Bottom	$\bot$	$\emptyset$
	Intersection	$C \sqcap D$	$C^I \cap D^I$
	Atomic negation	$\neg A$	$I \setminus A^I$
	Limited existential quantification	$\exists r$	$\{a \in \Delta^I \mid \exists b.(a,b) \in r^I$
	Universal quantification	$\forall r$	$\{a \in \Delta^I \mid \forall b.(a,b) \in r^I \rightarrow b \in C^I\}$

Table 2.1: Basic description logic language constructs

Most description logics consist of the constructs in Table 2.1. More expressive description logics may have additional constructs to those described in Table 2.1, most of which are depicted in Table 2.2.

Symbol	Construct	Syntax	Semantics
$\mathcal{U}$	Union	$C \sqcup D$	$C^I \cup D^I$
$\mathcal{C}$	Negation	$\neg C$	$I \setminus C^I$
$\mathcal{E}$	Existential quantification	$\exists r.C$	$\{a \in \Delta^I \mid \exists b.(a,b) \in r^I \wedge b \in C^I\}$
$\mathcal{N}$	Unqualified number restriction	$\geq nr$	$\{a \in \Delta^I \mid \#\{b \in \Delta^I \mid (a,b) \in r^I\} \geq n\}$
		$\leq nr$	$\{a \in \Delta^I \mid \#\{b \in \Delta^I \mid (a,b) \in r^I\} \leq n\}$
		$= nr$	$\{a \in \Delta^I \mid \#\{b \in \Delta^I \mid (a,b) \in r^I\} = n\}$
$\mathcal{Q}$	Qualified number restriction	$\geq nr.C$	$\{a \in \Delta^I \mid \#\{b \in C^I \mid (a,b) \in r^I\} \geq n\}$
		$\leq nr.C$	$\{a \in \Delta^I \mid \#\{b \in C^I \mid (a,b) \in r^I\} \leq n\}$
		$= nr.C$	$\{a \in \Delta^I \mid \#\{b \in C^I \mid (a,b) \in r^I\} = n\}$
$\mathcal{O}$	Nominals	$\{a_1, ..., a_N\}$	$\{a_1^I, ..., a_N^I\}$
$\mathcal{H}$	Role Hierarchy	$r \sqsubseteq s$	$r^I \subseteq s^I$
$(\mathcal{D})$	Datatypes		
$\mathcal{I}$	Inverse role	r^-	$\{(b,a) \in \Delta^I \times \Delta^I \mid (a,b) \in r^I\} \geq n\}$
$\circ$	Composition	$r \circ s$	$r^I \circ s^I$
$+$	Transitive closure	r^+	$\bigcup_{n \geq 1} (r^I)^n$

Table 2.2: Expressive description logic constructs

The name of a description more expressive than those depicted in Table 2.1 is usually denoted by the name of its corresponding basic DL, followed by a list of all other constructs allowed in the DL. For example, the description logic $\mathcal{ALC}$ corresponds to $\mathcal{AL}$ extended by allowing complex concept negation. Some DLs do not follow the naming scheme described above. For example, the DL $\mathcal{ALC}+$ is often abbreviated as $\mathcal{S}$.

We shall use the terms *complex concept*, *compound concept*, *concept description* to refer to concepts that are not atomic. Table 2.3 depicts the logical axioms that characterise an ontology. We say an interpretation I satisfies:

- A concept assertion axiom $C(a)$ if $a^I \in C^I$;
- A role assertion axiom $r(a,b)$ if $(a^I, b^I) \in r^I$;
- A individual equality axiom $a \approx b$ if $a^I = b^I$;

- A individual equality axiom $a \not\approx b$ if $a^I \neq b^I$;
- A concept inclusion axiom $C \sqsubseteq D$ if $C^I \subseteq D^I$;
- A concept equivalence axiom $C \equiv D$ if $C^I = D^I$;
- A role inclusion axiom $r \sqsubseteq s$ if $r^I \subseteq s^I$;
- A role inclusion axiom $r \equiv s$ if $r^I = s^I$;
- A complex role inclusion axiom $r \circ s \sqsubseteq p$ if $r^I \circ s^I \subseteq p^I$;
- An ontology O if it satisfies each axiom in O.

An interpretation I is said to *be a model* of O if I satisfies O. An ontology O is said to *entail* an axiom α if α is satisfied in all models of O. We write $O \models \alpha$ to indicate that O entails α.

We write $O \models O'$ to indicate that for every axiom $\alpha \in O'$, we have that $O \models \alpha$. We write $\models \alpha$ to indicate that α follows from the empty ontology.

If a complex concept C occurs on the left-hand side of TBox axiom $C \sqsubseteq D$ or $C \equiv D$, we refer to this TBox axiom as a *general concept inclusion* (GCI). A TBox is said to be *general* if it contains GCI axioms. Furthermore, given a concept inclusion axiom of the form $C \sqsubseteq D$, such that $C \sqsubseteq D$ is entailed by some ontology O, we shall refer to C as a *subconcept* or *subsumee* of D and D as a *superconcept* or *subsumer* of C. Given a TBox O, we say a subsumer D of a concept C is *maximally precise* if, for any other subsumer D' of C, we have that: $O \models D \sqsubseteq D'$; D may also be referred to as a *maximally precise subsumer* of C. Similarly, we say a subsumee C of a concept D is *maximally precise* if, for any other subsumee C' of D, we have that: $O \models C' \sqsubseteq C$; C may also be referred to as a *maximally precise subsumee* of D. The function $sig()$ returns the set of concept names, roles, and individuals that occur in a given ontology, concept, or axiom. The function $abox()$ returns the set of ABox axioms in a given ontology, and similarly, the function $tbox()$ returns the set of TBox axioms in a given ontology.

We now introduce querying over ontologies following Konev et al. [KLWW09]. An atom is of the form $C(x)$ or $r(x,y)$, where x,y are from a set of individual variables N_V, C is a concept and r is a role name. A conjunctive query is an expression of the form $q(x) = \exists y.\varphi(x,y)$, where x and y are disjoint sequences of variables and φ is a conjunction of atoms using only variables from $x \cup y$. The variables in x are the answer variables of q, and the ones in x are the (existentially) quantified variables.

Ontology Part	Axiom	Syntax	Semantics
TBox	Concept inclusion	$C \sqsubseteq D$	$C^I \subseteq D^I$
	Concept equivalence	$C \equiv D$	$C^I = D^I$
RBox	Role inclusion	$r \sqsubseteq s$	$r^I \subseteq s^I$
	Role equivalence	$r \equiv s$	$r^I = s^I$
	Complex role inclusion	$r \circ s \sqsubseteq p$	$r^I \circ s^I \subseteq p^I$
ABox	Concept assertion	$C(a)$	$a^I \in C^I$
	Role assertion	$r(a,b)$	$(a^I, b^I) \in r^I$
	Individual equality	$a \approx b$	$a^I = b^I$
	Individual inequality	$a \not\approx b$	$a^I \neq b^I$

Table 2.3: Basic logical axioms.

Let O be an ontology and $q(x) = \exists y.\varphi(x,y)$ a conjunctive query. Then a sequence of individual names is a certain answer to O and q if for every model I of O, there is a mapping $\pi : x \cup y \to \Delta^I$ such that $C(x) \in \varphi$ implies $\pi(x) \in C^I$, and $r(x,y) \in \varphi$ implies $(\pi(x), \pi(y)) \in r^I$.

2.2 Knowledge Structures and Knowledge Extraction Tools in Logic

Theoretical approaches for extracting and projecting knowledge in the context of communicating agents with logic-based knowledge bases have been proposed by Lin and Doherty and Lukaszewicz [Lin01, DLS01] for *projecting* knowledge. Here, by *projecting* (or *projection*), we mean conveying terms, concepts, or propositions with respect to a subset signature of a given knowledge base. Other well established logic based approaches include Uniform Interpolation (discussed in Section 2.2.3) module extraction [SSZ09, GPSK06, GHKS08, LWW07a] and Craig interpolation [TCFS13] are discussed in this chapter as they are also relevant to projecting knowledge represented using logic-based formalisms. We also discuss the relevance of these knowledge projection techniques to agent communication, the shortcomings encountered when adopting these methods to description logic, and approaches to mitigate such shortcomings (which forms a portion of our contribution).

As highlighted in Chapter 1, when humans communicate there are several knowledge structures available to convey knowledge to each other. There are a few analogous structures in logic to the knowledge structures available to humans. For example, definitions in human language communication are somewhat analogous to equivalence

($\equiv$) axioms with an atomic left hand side in description logics. For example, in natural language, a definition for *student* may be "*people who are enrolled in a university*"; an analogous axiom in description logic may be:

$$Student \equiv Person \sqcap \exists isEnrolledIn.University$$

Similarly, examples in human language communication are somewhat analogous to instances, individuals, or assertion axioms in description logics.

We are especially interested in definitions as they are knowledge structures that convey the *exact* meaning of a word, expression, or symbol. We are interested in how knowledge extraction techniques in DL may be used by logic-based agents to extract and exchange definitions to facilitate communication.

Motivating Example with Logic-based Agents Consider a scenario where there is a MAS that helps facilitate a food ordering website. The MAS consists of an agent that represents the website itself and an agent that represents each restaurant whose dishes can be ordered on the website as depicted in Figure 2.1.

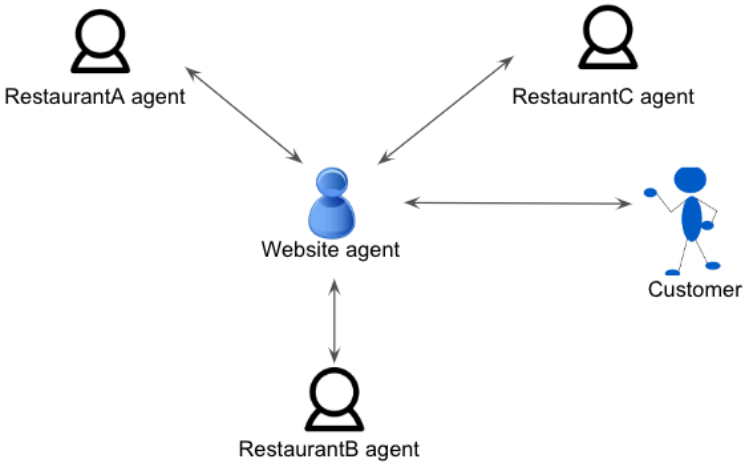

Figure 2.1: MAS of food ordering website.

Each agent's ontology is a representation of the food menu of its respective restaurant as illustrated in Figure 2.2. Each restaurant has its own unique set of dishes; however, in order to achieve some uniformity in the ordering process, all agents in the MAS refer to an *ingredients* ontology managed by the website agent. The website agent also has its own food menu ontology that also references the ingredients ontology

and aims at roughly capturing most of the unique dishes across all restaurants.

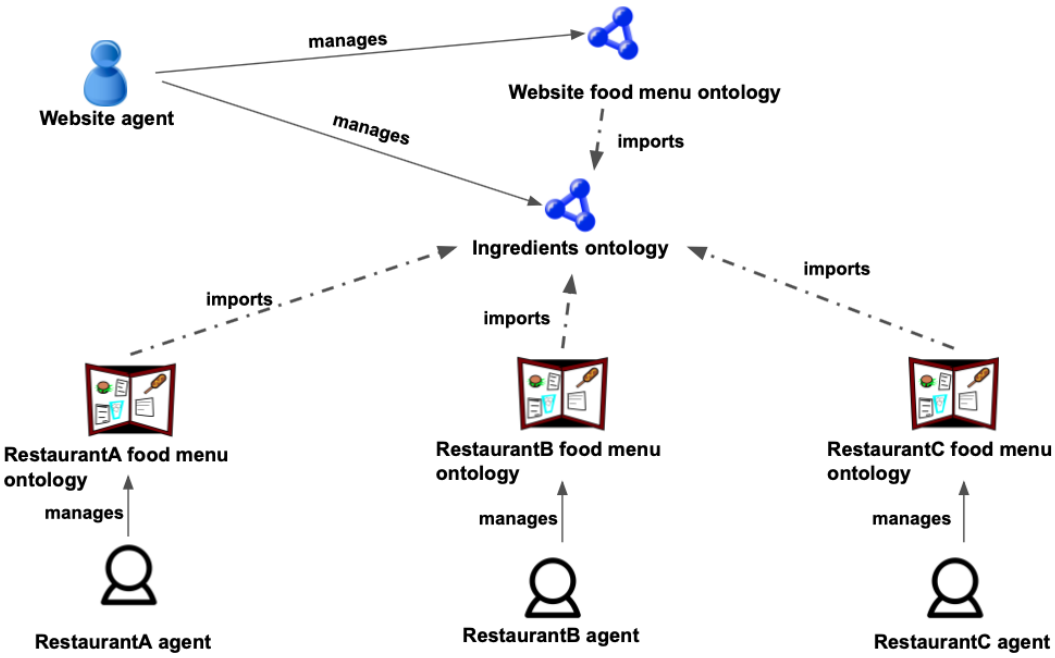

Figure 2.2: Ontologies of the agents.

All agents reference the *ingredients* ontology and thus share the vocabulary from that ontology. Below is a list of axioms from the ingredient ontology (the vocabulary of the ontology is the boldened text):

1. $Anchovy \sqsubseteq Fish$
2. $Prawn \sqsubseteq ShellFish$
3. $Shellfish \sqsubseteq Fish$
4. $Beef \sqsubseteq Meat$
5. $Jalepeno \sqsubseteq SpicyVegetable$
6. $Cheese \sqsubseteq DairyProduct$

Below is a list of axioms from the website food menu ontology:

1. $SeafoodPizza \equiv Pizza \sqcap \exists hasTopping.Fish$
2. $ShrimpPizza \equiv Pizza \sqcap \exists hasTopping.Shrimp$

3. $Shrimp \equiv Prawn$

Below is a list of axioms from the restaurantA's food menu ontology:

1. $PlainCheesePizza \equiv Pizza \sqcap \forall hasTopping.Cheese$
2. $AnchovyShrimpSpecial \equiv Pizza \sqcap \exists hasTopping.Shrimp \sqcap \exists hasTopping.Anchovy$
3. $AnchovyShrimpSpecial \sqsubseteq SeafoodPizza$

Because restaurantA's agent is not familiar with the concept name *Shrimp*, the website agent may run into communication issues with restaurantA's agent when conveying any statement that includes *Shrimp*. Similarly, because the website agent is not familiar with the concept name *AnchovyShrimpSpecial* restaurantA's agent may run into communication issues with the website agent when conveying any statement that includes *AnchovyShrimpSpecial*. However, both agents would not run into any issues when conveying any statements that only involves the vocabulary *hasTopping*, *SeafoodPizza*, *Anchovy*, *Prawn*, *Cheese*, and *Pizza* because both the website agent and restaurantA's agent are familiar with those terms. Similarly, any statement involving *Shrimp*, *AnchovyShrimpSpecial*, *PlainCheesePizza*, and *ShrimpPizza* is likely to lead to a misunderstanding between both agents because that vocabulary is not common to both agents. In fact, to communicate without misunderstandings, both agents may require reasoning or knowledge extraction techniques that help encapsulate the knowledge in their ontologies with respect to a subset vocabulary, in this case, the subset vocabularies of the ontologies that are common to both the website agent and restaurantA's agent. For example, if the website agent is attempting to convey *ShrimpPizza* to restaurantA's agent, it may attempt to use the concept "*Pizza* and *hasTopping* some *Shrimp*" since it is a definition of *ShrimpPizza*; however, this may still lead to communication issues as restaurantA's agent is not familiar with *Shrimp*. However since *Shrimp* is a definition of *Prawn*, the website agent may convey *ShrimpPizza* as "*Pizza* and *hasTopping* some *Prawn*" in place of *ShrimpPizza*.

2.2.1 Definitions and Definability in DL

The DL handbook [BCM$^+$03] describes definitions as equivalence axioms of the form $A \equiv C$ where A is a defined atomic concept. Baader [BCM$^+$03] makes a distinction between *primitive* and *non-primitive* definitions for concepts: *primitive* definitions are of the form of $A \sqsubseteq C$ where A is atomic while *non-primitive* definitions are of the form

of $A \equiv C$ where A is atomic. The focus is usually on *non-primitive* definitions as these are the only types of definitions considered when discussing *Beth definability*.

Beth Definability

The Beth definability property may be held by a logic and is predicated on two notions of *definability*: *explicit definability* and *implicit definability*.

Definition 1 (Implicit Definability)**.** *Let A be a concept, O a TBox, and $\Sigma \subseteq sig(A) \cup sig(O)$. A is* implicitly definable *in O with respect to Σ if for every two models $\mathcal{I}$ and $\mathcal{J}$ of O satisfying $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$, and for all $C \in \Sigma$, $C^{\mathcal{I}} = C^{\mathcal{J}}$, it holds that $A^{\mathcal{I}} = A^{\mathcal{J}}$.*

Definition 2 (Explicit Definability)**.** *Let A be a concept, O a TBox, and $\Sigma \subseteq sig(A) \cup sig(O)$. A is explicitly definable in O and a description logic $\mathcal{L}$ using only symbols from Σ if there is some concept C in $\mathcal{L}$ such that $O \models A \equiv C$ and $sig(C) \subseteq \Sigma$. This concept C is called an explicit definition of A in O with respect to Σ.*

A logic is said to have the Beth definability property if every concept defined implicitly can also be defined explicitly.

Definition 3 (Beth Definability)**.** *Let $\mathcal{L}$ be a description logic. $\mathcal{L}$ has the Beth definability property (BP) if for all C in $\mathcal{L}$, all TBoxes T in $\mathcal{L}$, and all signatures $\Sigma \subseteq sig(C,T)$, if C is implicitly definable with respect to Σ under T , then C is explicitly definable with respect to Σ under T.*

The DL $\mathcal{ALC}$ is shown to have the Beth definability property in [CCMV06].

A useful result of [TCFS13] is a test to determine whether a concept is implicitly definable over a given signature and ontology.

Definition 4 (Implicit Definability Test)**.** *Let C be a concept, O an ontology, O' a copy of O, and Σ a subset signature of O such that $sig(C) \subseteq sig(O)$, $sig(C) \not\subseteq \Sigma$, $\Sigma \subseteq sig(O)$, and O' is O with every concept name and role symbol $A \notin \Sigma$ replaced by a copy A', C is implicitly definable from Σ under O iff $O \cup O' \models C \equiv C'$.*

Ten Cate et al. [TCFS13] present an algorithm for extracting explicit definitions for implicitly defined concepts using Craig interpolation.

The implicit definability test has uses in other application domains. For example, Geleta et al. [GPT16] apply the results of [TCFS13] to ontology alignment by presenting practical methods for computing *minimal definition signatures* which can be used to identify seemingly unrelated concepts in ontologies to be aligned.

Definitorial TBoxes

At certain points in this thesis, the notion of *definitorial* ontologies is used. We adapt the notion *definitorial* from the description logic handbook [BCM+03]; however, to define and understand definitorial ontologies, we must first define the notions of *terminologies*, *name symbols*, *base symbols*, *base interpretations*, and *extensions*.

Definition 5 (Terminology [BCM+03])**.** *Let O be an ontology, O is said to be a* terminology *if O only consists of equivalence axioms of the form $A \equiv C$ where A is atomic, and for every such A, there is at most one axiom in O whose left hand side is A.*

Please note that a terminology may contain cycles, for example A is *cyclic* in the terminology $\{A \equiv \exists r.A\}$. We expand on cycles later on in this section.

Definition 6 (Base and Name Symbol [BCM+03])**.** *Let O be a terminology, a symbol S is said to be a* base symbol *if S only appears on the right hand side of axioms in O and a* name *symbol otherwise.*

Definition 7 (Base Interpretation [BCM+03])**.** *Let O be a terminology. An interpretation I of O is said to be a* base interpretation *of O if I only interprets the base symbols in O.*

Definition 8 (Extension [BCM+03])**.** *Let I be a base interpretation for a terminology O. An interpretation $\mathcal{J}$ that interprets the name symbols in O is an* extension *of I if it has the same domain as I, i.e., $\Delta^I = \Delta^{\mathcal{J}}$, and $\mathcal{J}$ agrees with I for the interpretation of the base symbols.*

A terminology O is said to be *definitorial* if every base interpretation of O has exactly one extension that is a model of O, i.e., if every base interpretation uniquely determines an interpretation of all symbols in a model of the terminology O.

It seems definitorial ontologies are intendend to capture the intuition that a word should not be its own definition or reference itself in its definition. For example, the definition of the word ‘cheese’ should not be simply ‘cheese’ or a phrase such as “Any food that is cheese”. To illustrate an analogous case in description logics, consider the ontology $O = \{A \sqsubseteq B\}$. Intuitively speaking, we do not consider A to be defined under O, even though $O \models A \equiv A$ (or similarly $O \models A \equiv A \sqcap B$).

In this thesis, we say a concept name A is *definable* under an ontology O if there exists some explicit definition C of A such that $O \models A \equiv C$ and $A \notin sig(C)$.

Definition 9 (Definable Concept Name)**.** *Let O be an ontology, and A a concept name in $sig(O)$. We say A is definable under O if there exists an explicit definition C of A in O such that $O \models A \equiv C$ and $sig(C) \subseteq sig(O) \setminus \{A\}$. We say A is definable with respect to a signature Σ under O if $\Sigma \subseteq sig(O) \setminus \{A\}$, there exists an explicit definition C' of A under O such that $O \models A \equiv C'$, and $sig(C') \subseteq \Sigma$.*

Observe that we may develop a decision procedure for testing whether a concept name A is definable under an ontology O with respect to a signature Σ by implementing and using the implicit definability test described in Definition 4.

Sometimes equivalences may falsely appear to make a concept name definable. As an example, consider the ontology $\{A \equiv \exists r.A\}$. Without giving it any proper thought, it would seem $\exists r.A$ is a definition for A under O; however, $\exists r.A$ references A, and therefore, following Definition 9 is not a definition for A.

Definition 10 (Cycles and Cyclic Concept names)**.** *Let O be a terminology and A a concept name in $sig(O)$. We say that A directly* uses *B in O if there is some $A \equiv C \in O$ and B occurs in C. We call* uses *the transitive closure of the relation directly* uses*. Let O be a terminology and A a concept name in $sig(O)$. We say A is* cyclic *if A uses itself. A terminology O is said to be cyclic if it contains a cyclic concept name; otherwise, O is called acyclic.*

In practice we would like to distinguish *uninteresting* cycles from *interesting* cycles. Uninteresting cycles involve concept names that use themselves in a TBox O but whose extension can always be uniquely determined by the corresponding interpretation of the base symbols in O. For example take O to be $\{A \equiv \exists p.B \sqcup (A \sqcap \neg A)\}$. A is cyclic in O, and thus O is cyclic. However, every extension of A is uniquely determined by the base interpretation of the (base) symbols $\{p, B\}$. Interesting cycles involve concept names that use themselves in a TBox O but whose extension can not be uniquely determined by the corresponding interpretation of the base symbols in O. For example take O to be $\{A \equiv \exists p.A\}$, and consider an interpretation I of O to be $p^I = \{(a, b, (b, b)\}$. A possible extension $\mathcal{J}$ of I can be $A^{\mathcal{J}} = \emptyset$. Another possible extension $\mathcal{J}'$ of I can be $A^{\mathcal{J}'} = \{a, b\}$. In general, when we talk about cycles we mean interesting cycles. Unfortunately, to the best of our knowledge, there is no known decision procedure to distinguish interesting cycles from uninteresting cycles [Neb90a]. Definitorial ontologies do not contain *interesting* cycles but may contain *uninteresting* cycles. For example let O to be $\{A \equiv \exists p.B \sqcup (A \sqcap \neg A)\}$, A is uses itself in O; however, since $(A \sqcap \neg A) \equiv \bot$ we have that O is equivalent to $\{A \equiv \exists p.B\}$.

The notion of *cyclicity* is also related to the notion of *unfolding*. Let O be a definitorial ontology, and C a concept expressed over $sig(O)$. Unfolding is the process of replacing the occurrence of any defined concept name $A \in sig(C)$, such that $A \equiv D \in O$, with D to get a new concept C' and then recursively repeating the same procedure on C' on all defined concept names until no further replacements are possible. In non-definitorial ontologies, the process is more complex, but the intuition is the same: replace the occurrence of a concept name with its definition or superclass until no further replacements are possible. If an ontology is *acyclic*, then the unfolding process terminates, otherwise, we may conclude that the ontology is cyclic, albeit not equivalent to an acyclic ontology. Thus, a naive algorithm for detecting cycles in definitorial ontologies is as follows: for every defined symbol A in O, unfold A, if A is encountered again during the unfolding process, then O is *cyclic*.

Definitorial *acyclic* ontologies guarantee that the definition of a concept can be expanded in an iterative fashion that terminates. Let C be the definition of a symbol A. Then every defined symbol in C can be replaced by its definition and this process can be repeated until the expansion of C consists only of base symbols (although the expansion can be exponential in the size of the original ontology [Neb90b]). Observe that if O is cyclic, this process does not terminate.

2.2.2 Module Extraction

Given a large ontology, a useful task for its users is to be able to extract *topic-specific* subsets of the ontologies. For example, given an ontology such as SNOMED CT [Don06], a medical practitioner may only be interested in knowing about the respiratory system, i.e., they only need a subset of SNOMED CT that encapsulates all knowledge related to the respiratory system. Extracting such topic-specific segments of an ontology is the motivation behind the research of *module extraction* in ontologies. Given an ontology O and a subset Σ of $sig(O)$, in a general sense, a module $\mathcal{M}$ should be a minimal subset of O that has the same entailments as O over Σ. This means that all the axioms that are entailed by O and can be expressed using only symbols in Σ are also entailed by $\mathcal{M}$. Intuitively speaking, to extract a module, a user provides a subset signature Σ of the ontology's signature that describes their topic of interest, a set of axioms capturing all knowledge entailed by the ontology relative to Σ is then extracted as the module.

Konev et al. [KLWW09] and Grau et al. [GPSK06, GHKS08], have formalised desirable properties that should characterise a module, perhaps the most important

of which is *conservative extensibility*. There are various notions of conservative extensions; however, the basic notions are *deductive* conservative extensions and *model* conservative extensions [SSZ09] formalised in Definition 11.

Definition 11 ([SSZ09])**.** *Let $\mathcal{L}$ be a description logic, $\mathcal{M} \subseteq \mathcal{T}$ be $\mathcal{L}$-TBoxes, and Σ a signature.*

1. *$\mathcal{T}$ is a* deductive Σ-conservative extension*($\Sigma - dCE$) of $\mathcal{M}$ with respect to $\mathcal{L}$ if for all GCI axioms α over $\mathcal{L}$ with $sig(\alpha) \subseteq \Sigma$, it holds that $\mathcal{M} \models \alpha$ iff $\mathcal{T} \models \alpha$.*

2. *$\mathcal{T}$ is a* model Σ conservative extension*($\Sigma - mCE$) of $\mathcal{M}$ if $\{I|_\Sigma \mid I \models \mathcal{M}\} = \{I|_\Sigma \mid I \models \mathcal{T}\}$.*

3. *$\mathcal{M}$ is a* dCE-based (mCE-based) module *for Σ of $\mathcal{T}$ if $\mathcal{T}$ is a Σ-dCE(Σ-mCE) of $\mathcal{M}$ with respect to $\mathcal{L}$.*

It follows from Definition 11 that if T is a $\Sigma - mCE$ of $\mathcal{M}$ then that T is a $\Sigma - dCE$ of $\mathcal{M}$. Deciding conservative extensions has been shown to be computationally expensive: it is double exponential (2-EXPTIME-complete) for $\mathcal{ALC}$ and $\mathcal{ALCQI}$ or even undecidable in cases such as $\mathcal{ALCQOI}$ [LWW07b]. Due to the complexity associated with deciding conservative extensions, syntactic approximations have been proposed [GLW06]. Syntactic approximations of modules, namely *locality*-based modules (LBMs) [GLW06], depend on the notion of *locality* which comes in two flavours: (1) syntactic locality (2) semantic locality. Syntactic locality relies on the syntax of an ontology to determine the axioms that are relevant to a given seed signature Σ for which we wish to extract a module. Both flavours of locality aim to preserve entailments over the seed signature Σ. An LBM is a subset of the axioms in a given ontology. The set of axioms are extracted based on a given seed signature which is a subset of the ontology's signature. Del Vescovo et al. [DV13] have noted that practical algorithms [SSZ09] for extracting approximations of locality based modules that have been shown to have a polynomial complexity and are included in the OWL API and is thus the preferred form of module extraction in this thesis.

To further describe LBMs, we must first define the properties that characterise them which include *inseparability, self-containment, depletion, and robustness*.

Definition 12 (Inseparability [SSZ09])**.** *For a given logic $\mathcal{L}$, an* inseparability relation *is a family $S = \{\equiv^S_\Sigma \mid \Sigma$ is a signature $\}$.*

Let O_1 and O_2 be ontologies and Σ be a signature, we define the following inseparability relations over two TBoxes O_1 and O_2

- *O_1 and O_2 are* Σ-concept name inseparable *($O_1 \equiv^c_\Sigma O_2$) if for all concept names A, B such that $A, B \in \Sigma$, we have that $O_1 \models A \sqsubseteq B$ iff $O_2 \models A \sqsubseteq B$.*

- *O_1 and O_2 are* Σ-subsumption inseparable *($O_1 \equiv^s_\Sigma O_2$) with respect to a logic $\mathcal{L}$ if for all concepts C and D such that $sig(C) \subseteq \Sigma$, $sig(D) \subseteq \Sigma$ we that that $O_1 \models C \sqsubseteq D$ iff $O_2 \models C \sqsubseteq D$.*

- *O_1 and O_2 are* Σ-query inseparable *($O_1 \equiv^q_\Sigma O_2$) if for all ABoxes $\mathcal{A}$, queries $q(x)$, and axioms, concepts, roles, or TBoxes α such that $sig(\mathcal{A}) \subseteq \Sigma$, $sig(q) \subseteq \Sigma$, and $sig(\alpha) \subseteq \Sigma$, we have that $(O_1, \mathcal{A}) \models q(\alpha)$ iff $(O_2 \mathcal{A}) \models q(\alpha)$.*

- *O_1 and O_2 are* Σ-model inseparable *($O_1 \equiv^m_\Sigma O_2$) if $\{I|_\Sigma \mid I \models O_1\} = \{I|_\Sigma \mid I \models O_2\}$.*

LBMs are model-inseparable. Inseparability relations are not only relevant for characterising modules, but are fundamental notions used in characterising knowledge structures that may be extracted from ontologies, including uniform interpolants. Inseparability relations are an important topic in their own right; however, the description in Definition 12 will serve our descriptive purposes for the rest of the chapter and thesis; further information about inseparability may be found in [KLWW09, BLR$^+$19, SSZ09].

We can now characterise self-containment and depletion.

Definition 13 (Modules, Self-containment and Depletion [SSZ09])**.** *Let S be an ineparability relation, O a TBox, $\mathcal{M} \subseteq O$, and Σ a signature. $\mathcal{M}$ is:*

1. *an* S_Σ-module *of O if $\mathcal{M} \equiv^S_\Sigma O$;*

2. *a* self-contained S_Σ-module *of O if $\mathcal{M} \equiv^S_{\Sigma \cup sig(\mathcal{M})} O$;*

3. *a* depleting S_Σ-module *of O if $\emptyset \equiv^S_{\Sigma \cup sig(\mathcal{M})} O \setminus \mathcal{M}$.*

LBMs are *self-contained* and *depleting* [SSZ09]. Intuitively speaking, this *roughly* means that given a notion of locality, an ontology O and a signature Σ, the locality based module $\mathcal{M}$ of O with respect to Σ *represents the same knowledge that O does over the signature Σ and over $sig(\mathcal{M})$.*

Definition 14 (Robustness [SSZ09])**.** *Let $\mathcal{L}$ be a DL. An inseparability relation S is called:*

1. robust under vocabulary restrictions *if, for all* $\mathcal{L}-TBoxes\ O_1, O_2$ *and all signatures* Σ, Σ' *such that* $\Sigma \subseteq \Sigma'$*, we have that: if* $O_1 \equiv^S_{\Sigma'} O_2$*, then* $O_1 \equiv^S_{\Sigma} O_2$.

2. robust under vocabulary extensions *if, for all* $\mathcal{L}-TBoxes\ O_1, O_2$ *and all signatures* Σ, Σ' *such that* $\Sigma' \cap (sig(O_1) \cup sig(O_2)) \subseteq \Sigma$*: if* $O_1 \equiv^S_{\Sigma} O_2$*, then* $O_1 \equiv^S_{\Sigma'} O_2$.

3. robust under replacement *if, for all* $\mathcal{L}-TBoxes\ O_1, O_2$*, all signatures* Σ*, and every* $\mathcal{L}-TBox\ O$ *such that* $sig(O) \cap (sig(O_1) \cup sig(O_2)) \subseteq \Sigma$*, we have that: if* $O_1 \equiv^S_{\Sigma} O_2$ *then* $O_1 \cup O \equiv^S_{\Sigma} O_2 \cup O$.

4. robust under joins *if, for all* $\mathcal{L}-TBoxes\ O_1, O_2$ *and all signatures* Σ *such that* $sig(O_1) \cap sig(O_2) \subseteq \Sigma$ *and* $i = 1, 2$*, we have that: if* $O_1 \equiv^S_{\Sigma} O_2$ *then* $O_i \equiv^S_{\Sigma} O_1 \cup O_2$.

LBMs have all the have all the robustness properties described in Definition 14 [SSZ09]

Syntactic LBMs come in two main flavours (formal definitions have been defined by Grau et al. [GHKS08]):

- 'Bottom'($\bot$) module: Let O be an ontology, and Σ a seed signature. Intuitively speaking, a *bottom* module for Σ under O captures all entailments that *preserve all the subsumers in* O of the all symbols in Σ.

  Example 2.2.1. Let O consist of the following axioms:

  1. $Anchovy \sqsubseteq Fish$
  2. $Prawn \sqsubseteq ShellFish$
  3. $ShellFish \sqsubseteq Fish$
  4. $Beef \sqsubseteq Meat$
  5. $Jalepeno \sqsubseteq SpicyVegetable$
  6. $Cheese \sqsubseteq DairyProduct$
  7. $SeafoodPizza \equiv Pizza \sqcap \exists hasTopping.Fish$
  8. $ShrimpPizza \equiv Pizza \sqcap \exists hasTopping.Shrimp$
  9. $Shrimp \equiv Prawn$
  10. $Sushi \sqsubseteq \exists hasTopping.Fish$
  11. $NonVeggiePizza \equiv Pizza \sqcap (\exists hasTopping.Meat \sqcup \exists hasTopping.Fish)$

12. $NonVeggiePizza \sqsubseteq WarmFood$
13. $WarmFood \sqsubseteq BakedPotato \sqcup Pizza \sqcup \exists hasTopping.Meat$

Let a seed signature $\Sigma = \{NonVeggiePizza\}$. A bottom module for Σ extracted from O consists of the following axioms:

1. $NonVeggiePizza \equiv Pizza \sqcap (\exists hasTopping.Meat \sqcup \exists hasTopping.Fish)$
2. $NonVeggiePizza \sqsubseteq WarmFood$
3. $WarmFood \sqsubseteq BakedPotato \sqcup Pizza \sqcup \exists hasTopping.Meat$

Observe that all subsumers of *NonVeggiePizza* (such as *WarmFood*, and *Pizza*) are entailed by this module, but also, all subsumers of *WarmFood* (i.e., $\{BakedPotato \sqcup Pizza \sqcup \exists hasTopping.Meat\}$) are also preserved by the module.

- 'Top'($\top$) module: Intuitively speaking, a *top* module for Σ captures all entailments that *preserve all the subsumees in O* of the all symbols in Σ.

Example 2.2.2. Let O be the same ontology from Example 2.2.1. Let a seed signature $\Sigma = \{NonVeggiePizza\}$. A bottom module for Σ extracted from O may consist of the following axioms:

1. $NonVeggiePizza \equiv Pizza \sqcap (\exists hasTopping.Meat \sqcup \exists hasTopping.Fish)$
2. $SeafoodPizza \equiv Pizza \sqcap \exists hasTopping.Fish$
3. $ShrimpPizza \equiv Pizza \sqcap \exists hasTopping.Shrimp$
4. $Shrimp \equiv Prawn$
5. $Anchovy \sqsubseteq Fish$
6. $Prawn \sqsubseteq Fish$
7. $Beef \sqsubseteq Meat$

Observe that all subsumees of *NonVeggiePizza* (such as *ShrimpPizza*, and *SeafoodPizza*) are entailed by this module, but also, all subsumers of *Fish* (i.e., $\{Shrimp, Anchovy, Prawn\}$) are also preserved by the module.

Communication protocols for agents that adopt module extraction would enable agents to leverage modules to extract and communicate specific portions of their knowledge. Doran et al. [DTPP09] have demonstrated that module extraction can be used

to reduce the number of concepts or axioms that have to be explored for ontology negotiation in the context of multi-agent systems.

In practice, because modules are self-contained, the signature of modules is usually larger than the seed signature provided; this is a strength in scenarios where the constraints on the seed signature are loose and all information relating to the seed signature. However, when there are strict constraints on the seed signature, this can be a disadvantage. Consider an agent who desires to extract the exact knowledge entailed by an exact signature under its given ontology. The agent may compute this knowledge by extracting a module, but with less signature precision: axioms of the module may contain symbols that are not in the signature provided by the agent, and as a result, the agent may find it challenging to use the module.

2.2.3 Forgetting and Uniform Interpolation

Given a knowledge base, or theory T and a set of predicates S such that $S \subseteq sig(T)$, roughly speaking, the goal of forgetting is to eliminate all symbols in S from T while preserving entailments over the rest of the signature of T. To our knowledge, Boole [Boo09] was the first to study forgetting under the description *elimination of the middle terms*. Subsequently forgetting was referred to as *variable elimination* in the context of classical logic.

We later observe two main forms of forgetting emerging from the literature, namely: *strong forgetting* (characterised by Lin and Reiter [LR94]) and *weak forgetting* (characterised by Zhang and Zhou [ZZ10]).

Definition 15 (Strong forgetting [LR94])**.** *Let F be a first-order logic formula and P a predicate. Then, F' is a result of strongly forgetting P in F if for every interpretation M, we have $M \models F'$ iff there is an interpretation M' with $M' \models F$ such that M' agrees with M on every predicate except P.*

Strong forgetting is sometimes also referred to as *semantic forgetting* or *model forgetting*.

Definition 16 (Weak Forgetting [ZZ10])**.** *Let F be a first-order logic formula and P a predicate. F' is the result of forgetting P iff for every formula G that does not contain P, we have $F' \models G$ iff $F \models G$.*

Weak forgetting is sometimes also referred to as *deductive forgetting*. We can thus observe that strong forgetting is concerned with models, as opposed to weak forgetting,

which is focused on logical entailments. Furthermore, observe from Definition 16 and Definition 11 that F is deductive $\{P\}$-conservative extension of F'.

In the context of first-order and second-order logic, forgetting has been studied as the problem of *second-order quantifier elimination* [Gab08]. Let F be a first-order formula and P be a predicate, the result of forgetting P from F is equivalent to the second-order formulae $\exists X.F[P/X]$, which reduces the problem of forgetting in first-order logic to the problem of second-order quantifier elimination. In the context of second-order logic, approaches to quantifier elimination include: (i) *hierarchical resolution* which is adopted by forgetting algorithms such as SCAN [GO92], and (ii) *skolemisation* which is used in the DLS system [Sza93].

From Definition 16 and Definition 17, we can see that forgetting is related to uniform interpolation in the sense that a uniform interpolant F for $\{S\}$ is equivalent to the result of *weakly* forgetting the symbols in $\{S\}$ from F. Thus, weak forgetting may be used to extract uniform interpolants for logics with uniform interpolation. It is worth noting there are logics that do not have uniform interpolation, an example is the modal logic $S4$ [GZ95].

Definition 17 ([Koo15]). *A logic $\mathcal{L}$* has uniform interpolation *if for every $\mathcal{L}$ formula F and every set of symbols S there exists a uniform interpolant of F for S.*

Forgetting and Uniform Interpolation in DLs

In the context of DL, uniform interpolation can intuitively be thought of as encapsulating the knowledge entailed by an ontology with respect to a given subset signature of the ontology. Given an ontology O and a subset Σ of the signature of O, let $\mathcal{V}$ denote a uniform interpolant relative to O, for every axiom α such that $O \models \alpha$ and $sig(\alpha) \subseteq \Sigma$, we have that $\mathcal{V} \models \alpha$ and $sig(\mathcal{V}) \subseteq \Sigma$. Another intuitive way of thinking about uniform interpolation is that given an ontology O and a subset signature Σ of O a uniform interpolant is an ontology that describes how all symbols Σ are related in O. Unlike module extraction, the signature of a uniform interpolant does not exceed the input signature; thus, uniform interpolation can be seen as a more precise knowledge extraction tool. However, uniform interpolation also has its downsides; for example, not all uniform interpolants can be expressed in standard DLs such as $\mathcal{ALC}$ [Koo15], additionally, for expressive DLs such as $\mathcal{SHROIQ}(\mathcal{D})$, no uniform interpolation methods are known.

Definition 18 (Forgetting and Uniform Interpolation). *Given an ontology O, a set of symbols Σ such that $\Sigma \subset sig(O)$, a* uniform interpolant *for O over Σ is an ontology $\mathcal{V}$*

such that $sig(\mathcal{V}) \subseteq \Sigma$, $O \models \mathcal{V}$, and $\mathcal{V}$ is a strongest such entailment for Σ, i.e., for any other entailment $\mathcal{V}'$ of O such that $O \models \mathcal{V}'$ and $sig(\mathcal{V}') \subseteq \Sigma$, then $\mathcal{V} \models \mathcal{V}'$. The ontology $\mathcal{V}$ is called a uniform interpolant of O for the signature Σ. We refer to Σ as the uniform interpolation signature. We also call $\mathcal{V}$ the result of forgetting $\bar{\Sigma}$ from O where $\bar{\Sigma}$ denotes $sig(O) \setminus \Sigma$.

A consequence of this definition is that if the DL $\mathcal{L}$ of O has the uniform interpolation property, then for every axiom ψ such that $sig(\psi) \subseteq \Sigma$, $O \models \psi$ iff $\mathcal{V} \models \psi$.

Example 2.2.3. Let O be the same ontology from Example 2.2.1.
Let $\Sigma = \{NonVeggiePizza\}$. A uniform interpolant for Σ extracted from O consists of the axiom $NonVeggiePizza \sqsubseteq \top$.

Now let $\Sigma = \{NonVeggiePizza, Pizza\}$. A uniform interpolant for Σ extracted from O consists of the axiom $NonVeggiePizza \sqsubseteq Pizza$.

This example illustrates a difference between uniform interpolants and modules: unlike modules the signature of uniform interpolants is always contained within the input signature.

Most of the recent practical efforts towards uniform interpolation in the context of DLs are based on the foundations provided by Lutz and Wolter. [LW11], second-order quantifier elimination [Gab08], and Ackermann-based approaches [Sch12]. Research on semantic (or strong) forgetting in the context of DLs include the works of Konev et al. [KWW09] Zhao and Schmidt [YZ15], and Sakr and Schmidt [MS21]. Research on deductive (or weak) forgetting in the context of DLs includes the works of Konev et al. [LK13] and Koopmann [Koo15, PK15a, PK15d, PK15c, PK15b].

Although the signature precision element of uniform interpolation is appealing, the downside to extracting uniform interpolants for ontologies is the computational complexity associated with extracting uniform interpolants. Lutz et al. [LW11] have shown that deciding the existence of uniform interpolants is 2-EXPTIME-complete, and the size of uniform interpolants is at most triple exponential in the size of the TBox of the original ontology.

Definition 19 (Soundness and Completeness For Uniform Interpolation)**.** *Let Int_X be a method for extracting uniform interpolants and let $\mathcal{L}$ be description logic. Int_X is said to be* interpolation sound *if for any $\mathcal{L}$ ontology O and signature Σ such that $\Sigma \subseteq sig(O)$, we have that Int_X terminates, and what Int_X returns is an $\mathcal{L}$ uniform interpolant of O with respect to Σ when applied to O.*
Let $\mathcal{V}$ be a uniform interpolant extracted for Σ under O using Int_X. Int_X is interpolation

complete *if for any $\mathcal{L}$ ontology O' such that $O \models O'$ and $sig(O') \subseteq \Sigma$, we have that, $\mathcal{V} \models O'$.*

Resolution

$$\frac{C_1 \sqcup A(t_1) \qquad C_2 \sqcup \neg A(t_2)}{(C_1 \sqcup C_2)\sigma}$$

Role Propagation

$$\frac{C_1 \sqcup (\forall r.D_1)(t_1) \qquad C_2 \sqcup (Qr.D_2)(t_2)}{(C_1 \sqcup C_2)\sigma \sqcup (Qr.D_{12})(t_1\sigma)}$$

Existential Role Restriction Elimination

$$\frac{C_1 \sqcup (\exists r.D)(t) \qquad \neg D(x)}{C_1}$$

Role Instantiation

$$\frac{C_1 \sqcup (\forall r.D_1)(t_1) \qquad r(t_2, b)}{C_1\sigma \sqcup D_1(b)}$$

where $Q \in \{\forall, \exists\}$, σ is the unifier of t_1 and t_2 if it exists, D_{12} is a possibly new definer representing $D_1 \sqcap D_2$ and no clause contains more than one negative definer literal of the form $\neg D(x)$, and none of the form $\neg D(a)$.

Figure 2.3: $RES_{\mathcal{ALC}}$ Calculus of LETHE [PK15d]

Definition 20 (Soundness and Refutational Completeness for Calculi)**.** *Let Calc be a calculus. Calc is said to be* refutationally complete *iff Calc derives $\perp$ when applied to any ontology O that is inconsistent. Calc is said to be* sound *if for any axiom ψ derived when Calc is applied to O, we have that $O \models \psi$.*

LETHE's method for extracting uniform interpolants (Int_{lethe}) is displayed in Algorithm 2.2.1

The implementation of uniform interpolation used in thesis is LETHE [PK15a] [1] for the description logic $\mathcal{ALC}$. LETHE is a saturation-based algorithm that applies a resolution calculus to eliminate symbols from an ontology. LETHE's resolution calculus is based on the rules in Figure 2.3. The calculus relies on the structural transformation of the ontology to a normal form given by a set of clauses of concept literals.

[1] LETHE can be accessed at `https://lat.inf.tu-dresden.de/~koopmann/LETHE/`.

Declausification:

$$\frac{O \sqcup \{\neg D \sqcup C_1, ..., \neg D \sqcup C_n\}}{O \sqcup \{D \sqsubseteq C_1, ..., D \sqsubseteq C_n\}} \text{ provided } D \text{ occurs only positively in } O$$

Non-cyclic definer elimination:

$$\frac{O \sqcup \{D \sqsubseteq C\}}{O[D/C]\}} \text{ provided } D \notin C$$

Definer purification:

$$\frac{O}{O[D/\top]\}} \text{ provided } D \text{ occurs only positively in } O$$

Cyclic definer elimination:

$$\frac{O \sqcup \{D \sqsubseteq C[D]\}}{O[D/\nu X.C[X]]\}} \text{ provided } D \notin C$$

where D is a definer symbol.

Figure 2.4: Rewrite rules to eliminate definers from clause sets [Koo15]

Int_{lethe}

Input: 1. An ontology O 2. A signature Σ.
Output: A uniform interpolant $\mathcal{V}$ of O for Σ.

1. Structurally transform O' following [Koo15] to get a set of clauses $\mathcal{N}$.
2. Saturate $\mathcal{N}$ by exhaustively applying $RES_{\mathcal{ALC}}$ calculus to $\mathcal{N}$ to derive $\mathcal{N}'$.
3. Remove clauses in $\mathcal{N}'$ containing symbols in $sig(O) \setminus \Sigma$.
4. Eliminate definer symbols from $\mathcal{N}'$ using the rules in Figure 2.4 and undo the structural transformation to get $\mathcal{V}$.

So-called definer symbols are denoted using D followed by a subscript and are introduced by the calculus during the normal form transformation to represent concepts that are under the scope of a quantifier. For example, the axiom $A \sqsubseteq \forall r.(C \sqcup B)$ is structurally transformed to the clause $\neg A \sqcup \forall r.(C \sqcup B)$ which is subsequently structurally

transformed to $\neg A \sqcup \forall r.D_1$, $\neg D_1 \sqcup C \sqcup B$ where D_1 is a definer symbol for the clause $C \sqcup B$. Resolution is restricted to definer symbols or concepts in the forgetting signature. Once all possible inferences have been made by the calculus (otherwise known as *saturation*), any clauses containing definer symbols or symbols in the forgetting signature are eliminated, resulting in a uniform interpolant.

Description logics that allow for value restrictions do not have the uniform interpolation property. This is because without any additional constructs to a description logic, uniform interpolants for general TBoxes expressed in DLs that allow value restrictions can not always be represented finitely due to the possibility of the existence of cycles that occur under roles. Consider the ontology O consisting of the axioms:

$$Human \sqsubseteq \exists hasParent.Human$$

$$Human(i)$$

O entails that the concept name $Human$ also belongs to the concept $\exists hasParent.Human \exists hasParent.Human \exists hasParent.Human \ldots$, and so on. A uniform interpolant $\mathcal{V}$ for O with respect to $\{hasParent\}$ can not capture this implicit chain under the *hasParent* role. As such, uniform interpolation methods such as [LPW10, PK15a, Nik11] employ the use of *fixpoint* operators to finitely represent such infinite chains. A description logic extended with fixpoint operators is usually suffixed by the symbol ν, for example, $\mathcal{ALC}\nu$ refers to $\mathcal{ALC}$ extended with least fixpoints. In this thesis, LETHE [PK15a] is the primary tool used to extract uniform interpolants. This means that our results on general ontologies (meaning ontologies that are not definitorial and may contain cycles) may contain fixpoints. Thus we expand on the fixpoint operators used by LETHE, namely, *greatest* fixpoint denoted $\mu X.C[X]$ and *least* fixpoint denoted $\nu X.C[X]$ where $C[X]$ is a concept in which X is a concept variable disjoint with $sig(O)$ that occurs only positively. X is said to be *bound* in concepts of the forms $\mu X.C[X]$ and $\nu X.C[X]$. If every concept variable in a concept is bound, the concept is closed; otherwise, it is open. Open concepts of the form $C[X]$ may be viewed as functions that take an arbitrary concept C_1 as an argument and return a concept $C[C_1]$ that is the result of replacing every occurrence of X in $C[X]$ by C_1. Thus, the greatest fixpoint $\mu X.C[X]$ denotes the smallest concept C' with respect to $\sqsubseteq$ for which $C[C'] \equiv C'$ and similarly, the least fixpoint $\nu X.C[X]$ denotes the largest concept C' with respect to $\sqsubseteq$ for which $C[C'] \equiv C'$. Let I be an interpretation and $f(X)$ a function that maps concept variables to subsets of a domain Δ^I such that $\cdot^I_{f(X)}$ maps open concepts

to subsets of Δ^I in the same way as $\cdot^I$ but with concept variables interpreted according to $f(X)$. Furthermore, let $f[X/\varepsilon]$ denote $f(X)$ modified by setting $f(X) = \varepsilon$, i.e., $f[X/\varepsilon](X) = \varepsilon$, then the semantics of $\mu X.C$ and $\nu X.C$ are defined as follows:

$$(\nu X.C)^I_{f(X)} = \bigcup\{\varepsilon \subseteq \Delta^I | C^I_{f[X/\varepsilon]} \subseteq \varepsilon\}$$

$$(\mu X.C)^I_{f(X)} = \bigcap\{\varepsilon \subseteq \Delta^I | \varepsilon \subseteq C^I_{f[X/\varepsilon]}\}$$

Let $\mathcal{L}$ be a description logic, $\mathcal{L}\mu$ denotes $\mathcal{L}$ with greatest fixpoint semantics and operators. Now we can understand a uniform interpolant $\mathcal{V}$ for the ontology $\{Human \sqsubseteq \exists hasParent.Human, Human(i)\}$ to be $\exists hasParent.\mu X.(\exists hasParent.X)(i)$.

Because fixpoints are not a standard DL construct, the implementation of LETHE provides an option to use definer symbols in place of cyclic concepts to allow uniform interpolants to be used by tools designed for DLs with the standard constructs discussed in Section 2.1. For example, given the ontology $\{Human \sqsubseteq \exists hasParent.Human, Human(john)\}$, a uniform interpolant for $\{hasParent, john\}$ extracted by the implementation LETHE that uses definer in place of cyclic concepts is $\{D_1 \sqsubseteq \exists hasParent.D_1, \exists hasParent.D_1(john)\}$ where D_1 is a definer symbol. We still suffix the DL of ontologies that use definer symbols with μ.

In practice we ignore (and discard where possible) concept descriptions that include fixpoint operators or definer symbols as we have no implemented means to evaluate such descriptions, i.e., any axiom α occuring in an ontology O whose signature includes a fixpoint or definer symbol is not considered in any observation or discussion.

Throughout this thesis, we use an implementation of LETHE that uses module-extraction as an optimisation step in order to narrow down the axioms for saturation. $RES_{\mathcal{ALC}}$ has been shown to be sound and refutationally complete for $\mathcal{ALC}$ [Koo15]. Int_{lethe} has been shown to be interpolation sound and complete for $\mathcal{ALC}$ [Koo15]

Thus, LETHE and its calculi can be considered sound and complete in all the senses discussed in Definitions 19 and 20 for $\mathcal{ALC}$ [Koo15].

2.2.4 Craig Interpolation

The Craig interpolation property for a logic $\mathcal{L}$ states that the for every implication $\phi \rightarrow \varphi$ that is true in $\mathcal{L}$, there exists a formula α in $\mathcal{L}$ such that $sig(\alpha) = sig(\phi) \cap sig(\varphi)$ and both $\phi \rightarrow \alpha$ and $\alpha \rightarrow \varphi$ are true in $\mathcal{L}$; α is then called an $\mathcal{L}$-interpolant for $\phi \rightarrow \varphi$.

Craig interpolation has been shown to have various applications, such as formal verification [vB08, McM03] and query rewriting in databases [Mar07, BLCT16]. The Craig interpolation property is tightly coupled with the Beth definability property (discussed in Section 2.2.1). The relationship between both properties is two-fold: (1) The proof of the Beth definability property can be constructed using the Craig interpolation property [Fit12, TCFS13]. As such, the Beth definability property is sometimes considered a consequence of the Craig interpolation property [AJM+21] (2) Craig interpolants can be used to extract definitions in logics that have the Beth definability property [TCFS13, AJM+21].

In the context of DL, the Craig interpolation property is formalised as follows [TCFS13]:

Definition 21 (Interpolation Property [TCFS13])**.** *A DL $\mathcal{L}$ is said to have the interpolation property if for all $\mathcal{L}$-concepts C_1, C_2 and all $\mathcal{L}$-TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$, if $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$, then there exists some $\mathcal{L}$-concept I such that*

- $sig(I) \subseteq (sig(C_1) \cup sig(\mathcal{T}_1)) \cap (sig(C_2) \cup sig(\mathcal{T}_2))$,
- $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq I$, *and*
- $\mathcal{T}_1 \cup \mathcal{T}_2 \models I \sqsubseteq C_2$

Such a concept is called an interpolant of C_1 and C_2 under $\langle \mathcal{T}_1, \mathcal{T}_2 \rangle$.

Ten Cate et al. [TCFS13] have proposed a tableaux-based algorithm to extract Craig interpolants for the DL $\mathcal{ALCF}$ and have shown that these interpolants can be extracted in double exponential time.

Craig interpolation and uniform interpolation differ in the sense that uniform interpolants generalise Craig interpolants as follows [AJM+21]: for a logic $\mathcal{L}$, given a formula (set of formulae) ϕ and a signature Σ such that $\Sigma \subseteq sig(\phi)$, a uniform interpolant φ for Σ under ϕ in $\mathcal{L}$ is a formula (set of formulae) φ such that $\phi \rightarrow \varphi$ is true in $\mathcal{L}$, $sig(\varphi) \subseteq \Sigma$, and for any formula α such that $\phi \rightarrow \alpha$ is true in $\mathcal{L}$ and $sig(\alpha) \subseteq \Sigma$, we have that $\varphi \rightarrow \alpha$ is true in $\mathcal{L}$. Intuitively speaking, uniform interpolants are more concerned with preserving the consequences of a formula with respect to a signature. Similarly, Craig interpolants can also be extracted for restricted signatures; in the style of Ten Cate et al. [TCFS13] given a formula (set of formulae) ϕ we can take advantage of the redundancy $\phi \leftrightarrow \phi$ to extract a Craig interpolant for a subset signature Σ of $sig(\phi)$ as follows: create a duplicate ϕ' of ϕ such that for every symbol $s \in sig(\phi) \setminus \Sigma$,

s is renamed to s' in ϕ', now extract the Craig interpolant of ϕ and ϕ' to get a Craig interpolant of ϕ with respect to Σ. In fact, this is the property that enables us use Craig interpolants to extract explicit definitions for concepts. Given a TBox $\mathcal{T}$ that defines a concept C_1, such that $\mathcal{T} \models C_1 \equiv C_2$, and $sig(C_2) \subseteq sig(C_1)$. Since we have that $\mathcal{T} \equiv \mathcal{T}$, an explicit definition for C_1 can be extracted using the following steps:

1. Create a duplicate TBox $\mathcal{T}'$ for the signature $\Sigma = sig(\mathcal{T}) \setminus sig(C_1)$ where every symbol $S \in sig(\mathcal{T}) \setminus \Sigma$ is renamed to S' in $\mathcal{T}'$ (note that $sig(\mathcal{T}) \setminus \Sigma = sig(C_1)$, therefore the symbols in C_1 are the only symbols renamed in $\mathcal{T}'$),

2. $\mathcal{T} \cup \mathcal{T}' \models C_1 \equiv C_1'$, as such, extract a Craig interpolant of C_1 and C_1' under $\langle \mathcal{T}, \mathcal{T}' \rangle$ to get a definition of C_1.

Craig interpolation is generally considered one of the most significant properties in logic [AJM+21] and its usefulness can be extended to agent communication: since Craig interpolants can be used to extract definitions of concepts, an agent may directly use Craig interpolants to convey the definition of a concept to another agent.

Uniform interpolants are stronger than Craig interpolants in that they capture the strongest entailments over an input signature.

Since Craig interpolation and uniform interpolation have similar computational costs (they both have double exponential complexity [TCFS13, Koo15]), we decided to investigate and evaluate uniform interpolation's role in agent communication because we had access to an existing tool that implements uniform interpolation (LETHE [2]) implementation; to the best of our knowledge, there are no tools that implement Craig interpolation for DL. Furthermore, uniform interpolation is more appropriate due to its strong entailment properties (i.e., it precisely and maximally preserves entailments over restricted signatures).

[2]LETHE can be accessed at: `https://lat.inf.tu-dresden.de/~koopmann/LETHE/`.

	Craig Interpolation	Uniform Interpolation
Input	1. An ontology O, 2. Two concepts C_1, C_2, 3. A signature Σ such that $\Sigma \subseteq sig(C_1, O) \cap sig(C_2, O)$.	1. An ontology O, 2. A signature Σ such that $\Sigma \subseteq sig(O)$.
Output	A *concept* I such that $sig(I) \subseteq \Sigma$, $O \models C_1 \sqsubseteq I$, and $O \models I \sqsubseteq C_2$ (i.e., I is a Craig interpolant of C_1 and C_2) if it exists.	An *ontology* $\mathcal{V}$ representing a uniform interpolant of O for Σ

Table 2.4: Basic differences between Craig interpolation and uniform interpolation

Example 2.2.4. Let O consist of the following axioms:

1. $NonVeggiePizza \equiv Pizza \sqcap (\exists hasTopping.Meat \sqcup \exists hasTopping.Fish)$
2. $NonVeggiePizza \sqsubseteq WarmFood$
3. $WarmFood \sqsubseteq BakedPotato \sqcup Pizza \sqcup \exists hasTopping.Meat$

Let:

- $C_1 = Pizza \sqcap (\exists hasTopping.Meat \sqcup \exists hasTopping.Fish)$
- $C_2 = BakedPotato \sqcup Pizza \sqcup \exists hasTopping.Meat$

We have that:

- $\models C_1 \sqsubseteq C_2$
- $\Sigma = sig(C_1) \cap sig(C_2) = \{Pizza, hasTopping, Meat\}$.

A Craig interpolant I of C_1 and C_2 is the *concept* $Pizza \sqcup \exists hasTopping.Meat$ while an $\mathcal{ALC}$ uniform interpolant $\mathcal{V}$ of O for Σ consists of the following axioms:

1. $Pizza \sqsubseteq \top$
2. $Meat \sqsubseteq \top$

This is because O entails only tautologies over Σ.
Now consider $\Sigma' = \{NonVeggiePizza, hasTopping, Meat\}$, let C_1 and C_2 be the same as before. We still have that Craig interpolant I of C_1 and C_2 is the *concept* $Pizza \sqcup \exists hasTopping.Meat$. However, an $\mathcal{ALC}$ uniform interpolant $\mathcal{V}$ of O for Σ consists of the axiom $NonVeggiePizza \sqsubseteq Pizza \sqcup \exists hasTopping.Meat$.

2.2.5 Strongest Necessary Conditions and Weakest Sufficient Conditions

Given a knowledge base, one may be interested in projecting knowledge to a subset signature of the knowledge base such as with the cases of communicating agents and summarising ontologies described above. Consider a logical formula X expressed in some knowledge base or ontology, one may be interested in expressing X is a subset signature Σ of O that may allow for better understanding [DLS07]. This is captured using *weakest sufficient conditions* (SNCs) and *strongest necessary conditions* (WSCs) [DLS07]. Let O be an ontology, C a concept, and Σ a subset signature of $sig(O)$, a strongest necessary condition of C with respect to Σ under O is a concept C' that is a closest [3] *implicate* or *subsumer* of C under O expressed in Σ. Similarly, a weakest sufficient condition is a concept D that is a closest *implicant* or *subsumee* of C of C under O expressed in Σ.

Observe that definitions for concepts may not always exist, and even when they exist, they may not exist with respect to a subset vocabulary of an ontology. Thus, in communication scenarios where logic-based agents wish to convey concepts that are not defined, the notions strongest necessary and weakest sufficient conditions may be used to capture descriptions of concepts in terms of their subsumers and subsumees.

Doherty et al. [DLS07] describe SNCs and WSCs as useful to *"Building Communication Interfaces between Agents"*. In the context of reasoning about *knowledge*, WSCs and SNCs can be used to characterise *definability* for terms, symbols, or concepts in knowledge bases [Lin01, DLS07]. Furthermore, they may be used to share concept descriptions using different vocabularies [DLS07].

In the context of reasoning about *actions*, incorporating WSCs and SNCs into agent frameworks can enable agents to reason about causality, premises, and consequences in relation to *states* or *events* [Lin01, DLS01, DLS07].

SNCs and WSCs have formal characterisations in propositional, first-order, and second-order logic but seem to be lacking formal descriptions in DL, as part of our contribution, we formalise SNCs and WSCs in DL, but first provide an overview of their characterisations in other logics.

[3] By "closest", we mean with respect to an ordering of implication, subsumption or logical entailment.

Formal Definitions

Propositional Logic [Lin01]

Definition 22. *Let T be a knowledge base, P a set of propositions in T , and q a proposition in T but not in P. A formula φ of P is said to be a* necessary condition *of q on P under T if $T \models q \to \varphi$. It is said to be a* strongest necessary condition *if it is a necessary condition, and for any other necessary condition φ', we have that $T \models \varphi \to \varphi'$.*

Definition 23. *Let T be a knowledge base, P a set of propositions in T , and q a proposition in T but not in P. A formula φ of P is said to be a* sufficient condition *of q on P under T if $T \models \varphi \to q$. It is said to be a* weakest sufficient condition *if it is a sufficient condition, and for any other sufficient condition φ', we have that $T \models \varphi' \to \varphi$.*

First Order Logic [DLS01]

Definition 24. *By a* necessary condition *of a formula α on the set of relation symbols P under a knowledge base T we shall understand any formula ϕ containing only symbols in P such that $T \models \alpha \to \phi$. It is a* strongest necessary condition*, denoted by $SNC(\alpha; T; P)$ if, additionally, for any necessary condition φ of α on P under T, we have that $T \models \phi \to \varphi$.*

Definition 25. *By a* sufficient condition *of a formula α on the set of relation symbols P under a knowledge base T we shall understand any formula ϕ containing only symbols in P such that $T \models \phi \to \alpha$. It is a* weakest sufficient condition*, denoted by $WSC(\alpha; T; P)$ if, additionally, for any sufficient condition φ of α on P under T, we have that $T \models \varphi \to \phi$.*

Second Order Logic [Wer12] In the context of second order logic, we only find a definition of strongest necessary conditions stated below.

Definition 26 (Definition 26 [Wer12])**.** *A strongest necessary condition of a formula G on a set of predicate symbols P under formula F is a formula H such that:*

1. *H contains only predicate symbols from P.*
2. *$F \models G \to H$.*
3. *For all formulas H' such that H' contains only predicate symbols from P and $F \models G \to H'$ it holds that $F \models H \to H'$.*

We can observe from the definitions that the properties of strongest necessary conditions and weakest sufficient conditions are similar across propositional, first order, and second order logic: (1) in all logics, the strongest necessary condition is expressed be within the provided signature (2) in all logics, the strongest necessary condition is a closest upper approximation of the input symbol or formula, with respect to logical implication.

Extracting the conditions

For the discussed logics, extracting strongest necessary and weakest sufficient conditions is usually based on uniform interpolation. We provide the corresponding characterisations below.

Propositional Logic [Lin01] Let ϕ be a propositional formula, and P a set of propositions, in the theorem below, the operator $forget(\phi;P)$ denotes the result of forgetting all propositions in P from ϕ.

Theorem 1 (Theorem 2 [Lin01])**.** *Let T be a knowledge base, P a set of propositions, and q a proposition in T but not P. Let P' be the set of propositions that are in T but not in $P \cup \{q\}$. Then we have the following:*

1. *The strongest necessary condition of q on P is $forget(T(q/true);P')$.*
2. *The weakest sufficient condition of q on P is $\neg forget(T(q/false);P')$.*

First Order Logic [Lin01]

Lemma 2 (Lemma 4.1 [DLS01])**.** *For any formula α, any set of relation symbols and a closed[4] knowledge base T:*

1. *the strongest necessary condition $SNC(\alpha;T;P)$ is defined by $\exists\phi.[T \wedge \alpha]$,*
2. *the weakest sufficient condition $WSC(\alpha;T;P)$ is defined by $\forall\phi.[T \rightarrow \alpha]$,*

where ϕ consists of all the relation symbols appearing in T and α but not in P.

[4] In fact, it suffices to assume that the set of free variables of T is disjoint from the set of free variables of α.

Briefly speaking, given a formula F, $\exists P.[F]$ denotes a second-order quantifier over predicates symbols appearing in P, which can then be eliminated using second-order quantifier elimination. We can now understand from Lemma 2 an algorithm to extract $SNC(A;T;P)$ by using second-order quantifier elimination: $SNC(\alpha;T;P)$ is equivalent to $\exists P'.[T \wedge \alpha]$ where P' denotes all predicate symbols appearing in T or α but not in P Approaches to uniform interpolation such as those proposed by Koopmann [PK15a] can be interpreted as implementing second-order quantifier elimination as an approach to uniform interpolation in description logics.

Example 2.2.5. Let the knowledge base T consist of the following formulas:

1. $\forall x.[NonVeggiePizza(x) \leftrightarrow Pizza(x) \wedge (\exists y.(hasTopping(x,y) \wedge Meat(y))$ $\vee \exists z.(hasTopping(x,z) \wedge Fish(z)))]$
2. $\forall x.[NonVeggiePizza(x) \rightarrow WarmFood(x)]$
3. $\forall x.[WarmFood(x) \rightarrow BakedPotato(x) \vee Pizza(x) \vee$ $\exists y.(hasTopping(x,y) \wedge Meat(y))]$

Let the set of relation symbols $P = \{Pizza, hasTopping, Meat\}$. Let a formula $\alpha =$ $NonVeggiePizza(i)$, then following Lemma 2,

$$SNC(NonVeggiePizza(i);T;P) =$$

$$\exists NonVeggiePizza[\forall x.[(Pizza(x) \vee (\exists y.(hasTopping(x,y) \wedge Meat(y))] \wedge NonVeggiePizza(i)]$$

Characterising Definitions in terms of SNCs and WSCs

Proposition 3 (Proposition 2 [Lin01])**.** *A propositional knowledge base T defines a proposition q on a set of propositions P iff $T \models \varphi \rightarrow \phi$ where φ is any strongest necessary condition of q on P and ϕ any weakest sufficient conditions of q on P, both under the knowledge base T.*

Proposition 3 reduces the problem of extracting a definition of q with respect to P to extracting the strongest necessary condition or weakest sufficient condition of q: if $T \models \varphi \rightarrow \phi$ holds, $T \models \varphi \leftrightarrow \phi$ (because ϕ is a weakest sufficient condition, $T \models \phi \rightarrow \varphi$) holds and consequently, $T \models q \leftrightarrow \varphi$ holds. To demonstrate the proof of this, observe that since ϕ is a weakest sufficient condition of q, it follows that $T \models \phi \rightarrow q$; combined with the fact that $T \models q \rightarrow \varphi$, and $T \models \varphi \rightarrow \phi$, it follows that $T \models q \equiv \varphi$.

Thus an algorithm for testing for the definability of a proposition q with respect to a set of propositions P under a knowledge base T can be based on extracting the strongest necessary and weakest sufficient conditions: it suffices to check if $T \models \varphi \rightarrow \phi$, where φ denotes the strongest necessary condition of q with respect to P and ϕ denotes the weakest sufficient condition of q with respect to P; if $T \models \varphi \rightarrow \phi$, then q is definable with respect to P under T and both ϕ and φ are suitable definitions for q.

2.2.6 Prime implicates and Prime implicants

Prime implicates are the 'strongest clausal consequences of a formula' [Bie09] (in DL terms, 'strongest subsumers of a concept') such that removing any literal from a prime implicate results in the prime implicate no longer being a consequence ensuring that redundancies are not contained in any prime implicate. Prime implicants are the dual notion of prime implicates: the weakest terms that imply a formula (in DL terms, the 'weakest subsumees of a concept'). Bienvenu [Bie08] formalises the notion of prime implicates in the description logic $\mathcal{ALC}$ by first defining a set of literals L, clauses Cl and cubal concepts Cb as follows:

$$L ::= \top|\bot|A|\neg A|\forall r.Cl|\exists r.Cb$$

$$Cl ::= L|Cl \sqcup Cl$$

$$Cb ::= L|Cb \sqcap Cb$$

A clause Cl is said to be a *prime implicate* of a concept C iff:

1. $\models C \sqsubseteq Cl$
2. For any Cl' if $\models Cl' \sqsubseteq Cl$ then $\models Cl \sqsubseteq Cl'$

Implicates and prime implicates are designed to characterise subsumptions between concepts and do not factor in background ontology, i.e., subsumption is only measured with respect to the empty ontology.

Example 2.2.6. This example is adopted from [Bie07]. Let the $E = A \sqcap (B \sqcup C) \sqcap \exists r.\top \sqcap \forall r.(B \sqcap (A \sqcup C)) \sqcap \forall r.(B \sqcup D)$. The prime implicates of E are the following:

1. A
2. $(B \sqcup C)$

3. $\exists r.(B \sqcap A) \sqcup \exists r.(B \sqcap C)$

4. $\forall r.B$

5. $\forall r.(A \sqcup C)$

An example of a non-prime implicate of E is $\phi = A \sqcap (B \sqcup C)$ because stronger consequences of ϕ are A and $(B \sqcup C)$.

In Chapter 5 we expand on the relationship between prime implicates, prime implicants, strongest necessary conditions, and weakest sufficient conditions.

2.2.7 Most Specific Concept

Let i be an individual occurring in an ABox (or set of assertions) $\mathcal{A}$. A most specific concept (msc) C of i is a *minimal* concept that i is an instance of with respect to $\mathcal{A}$. Here by minimal we mean that C is a subconcept of any other concept that i belongs to.

Definition 27 (Most Specific Concept [BCM+03])**.** *Let $\mathcal{A}$ be an ABox in a given description logic $\mathcal{L}$ and i and individual in $\mathcal{A}$. We say C is a* most specific concept *of i in $\mathcal{A}$ if for every* **L***-concept D such that $\mathcal{A} \models D(i)$, we have that $\models C \sqsubseteq D$.*

Most specific concepts have been shown to have applications in bottom-up construction of ontologies (i.e., constructing ontologies from a set of instances of concepts) [BCM+03].

Example 2.2.7. Let $\mathcal{A}$ be the set of the following assertions:

1. $A(i)$

2. $B(j)$

3. $r(i,k)$

The most specific concept of i is $A \sqcap \exists r.\top$.

Most specific concepts are focused on extracting descriptions for individuals and lack the appeal of a signature restriction element. Our application of communicating agents is focused on extracting descriptions for concepts and has a signature restriction element, thus we will not explore most specific concepts furthermore in this thesis.

2.2.8 Least Common Subsumers

Given a set of concepts Σ, the least common subsumer (lcs) is the *minimal* concept that subsumes all concepts in Σ. Here by *minimal*, we mean that any other concept that subsumes all the concepts in Σ is equivalent to the least common subsumer or a superconcept of the least common subsumer. Intuitively speaking, the least common subsumer of all the concepts in Σ encapsulates the commonalities among all concepts in Σ.

Definition 28 (Least Common Subsumer [BCM$^+$03])**.** *Let $\mathcal{L}$ be a description logic. An $\mathcal{L}$-concept C is the* least common subsumer *of the $\mathcal{L}$-concepts $C_1, ..., C_n$ iff:*

1. $\models C_i \sqsubseteq C$ *for all* $i = 1, ..., n$*, and*
2. *For any other concept C' that satisfies (1), we have that* $\models C \sqsubseteq C'$

It is worth noting that Definition 28 may be also extended to include a background $\mathcal{L}$ ontology O for entailment and would still count as a definition of Least Common Subsumers.

Observe that for any description logic $\mathcal{L}$ that allows for disjunction ($\sqcup$), the lcs of a set of concepts $C_1, ..., C_n$ is the disjunction of all such concepts: $C_1 \sqcup ... \sqcup C_n$, and thus trivial.

Least common subsumers have been shown to have applications in inductive learning and bottom-up construction of ontologies (i.e., constructing ontologies from a set of instances of concepts) [BCM$^+$03].

Example 2.2.8. This example is adapted from [Baa03]. Let Σ be the set of the following $\mathcal{EL}$-concepts:

1. $\exists r.A \sqcap \exists r.C$
2. $\exists r.B \sqcap \exists r.C$
3. $\exists r.C$

The least common subsumer of Σ is the $\mathcal{EL}$-concept $\exists r.C$ because:

1. $\exists r.A \sqcap \exists r.C \sqsubseteq \exists r.C$
2. $\exists r.B \sqcap \exists r.C \sqsubseteq \exists r.C$
3. $\exists r.C \sqsubseteq \exists r.C$

In Chapter 5 we expand on the relationship between least common subsumers and strongest necessary conditions.

Chapter 3

Background On Agents

Where used in the context of this thesis, *agent* refers to a *software entity* that can form a complex symbolic representation of an environment and continuously interact with this environment in an autonomous or intelligent fashion. We refer to these agents as *logic-based* agents or *knowledge-based* agents following Russell [Stu03] as the central component of such agents is their knowledge base represented using some logic-based formalism. The *environment* mentioned may consist of other processes or other agents. The semantics of the words *autonomy* and *intelligent* are often imprecise; however, we interpret both words as agents (or more generally, entities) that do not require regular human guidance or mediation.

This paradigm of agents is referred to as "Agent-oriented programming" (AOP) as formalised by Shoham [Sho93]. AOP implements concepts from artificial intelligence in the context of distributed systems, robotics, and software engineering. AOP is motivated by applications such as personal assistance systems, control systems, manufacturing, systems diagnostics, air traffic systems, and computer network management. Such systems are required to perform management and control tasks in complex and dynamic environments. Applying conventional software engineering techniques to develop such systems proves to be difficult and expensive to maintain [RG$^+$95]. From a robotics perspective, an agent-oriented program would consist of descriptions of the behaviour of the robots in various certain contexts: for example what a cleaning robot should do (the behaviour) when it observes a piece of garbage (the context). Ideally, an agent programming language consists of an agent-oriented middleware that relieves programmers of the task of developing an infrastructure for repetitive domain-independent processes, thereby enabling the programmers to focus on key system logic of the agents by building multi-agent systems [BPR01].

In AOP, agents are characterised using *states*. These states consist of components such as *beliefs, intentions,* and *desires*. AOP paradigms are specified using agent architectures and each architecture has strengths, weaknesses, and use-cases. There are three main architectures: *belief-desire-intention* (BDI), reactive, and layered architectures. The BDI architecture consists of three main components: *beliefs*, *desires*, and *intentions* which describe the *states* or *mental attitudes* of an agent. These components are usually modelled using some form of logic formalism, and as a result, the agents can practically reason about these mental states. Implementations of BDI agents include *procedural reasoning system* (PRS)[FMA92], *distributed multi-agent reasoning system* (dMARS) [dKLW05], and *AgentSpeak* [Rao96]. Reactive architectures consist of mappings of situations/observations to actions and implement decision making this way. Unlike logic-based architectures, the agents do not have any complex representation of the world and hence cannot perform any reasoning. Implementations of reactive architectures include Brooks's subsumption architecture [Bro86]. Layered architectures are composed of both BDI and reactive components. Layered architectures are also known as hybrid architectures and aim to adopt the advantages of both reactive and BDI architectures. Implementations of layered architectures include the InteRRaP architecture [MP94].

The aim of this chapter is to discuss AOP frameworks in the context of their knowledge/information-management, sharing and retrieval applications. Following research question 4 (**RQ4** from Section 1.2: "*How can the relevant knowledge extraction techniques be integrated into existing MAS frameworks or agent communication protocols?*"), we are especially interested in the communication issues that may be faced by agents in AOP frameworks with ontologies as a central component. The communication issues that may be faced by such agents are predicated-upon and subject-to the communication mechanisms that underpin the respective AOP frameworks, and as such, this chapter also investigates the communication mechanisms of AOP frameworks.

Overview of Research Methodology We surveyed several proceedings of well - established publishing venues whose main topic was about agents or included a track about agents. The venues explored include:

- International Conference on Autonomous Agents and Multiagent Systems (AAMAS)
- International Conference on Agents and Artificial Intelligence (ICAART)

- International Joint Conference on Artificial Intelligence (IJCAI)
- International Semantic Web Conference (ISWC)
- International Conference on Principles of Knowledge Representation and Reasoning (KR)

In addition, several search engines were used, these include:

- Google Scholar
- Semantic Scholar
- Microsoft Academic
- Connected Papers

Lastly, we explored the publications (via DBLP) of several notable contributors in the domain of logic-based agents, including:

1. Rafael H. Bordini, author of the JASON agent programming language.
2. Michael Woolridge
3. Alison R. Panisson
4. Viviana Mascardi
5. John-Jules Meyer
6. Jurriaan van Diggelen
7. Frank Dignum
8. Alessandro Ricci

When exploring and scanning the resources above, the aim was to focus on any publications about agent systems that included the use of ontologies or were based on a logic formalism in order to find existing work related to our motivation and research questions.

3.1 Agent Oriented Programming Frameworks

Shoham [Sho93] characterises an AOP framework as having at least the following 3 components:

- A formal language that provides the syntax and semantics necessary to describe the mental state of an agent [Sho93].
- A programming language that can be used to design and program agents with respect to the semantics of the mental state [Sho93].
- A compiler that compiles agent programs into executable programs [Sho93].

There are two predominant AOP languages and frameworks that have some of these components: JASON and JADE.
JASON is an implementation and extension of the langauge *AgentSpeak*. We characterise and understand *AgentSpeak* via the discussion of JASON as JASON is a direct implementation of AgentSpeak. We also use the terms "JASON", and "AgentSpeak" interchangeably. JADE is an AOP framework that is an implementation of specifications written by the *foundations for intelligent physical agents* (FIPA). The decision to discuss JADE and JASON is motivated by the fact that, to the best of our knowledge, most AOP frameworks either implement or extend AgentSpeak or FIPA specifications; examples include: [VDRA11, KB08, BPR01, LPL$^+$14, RBHC19, SG11]. Furthermore, to the best of our knowledge, JASON is the most extended AgentSpeak AOP framework, and JADE is the most used implementation of the FIPA specifications.

3.1.1 JASON

JASON [BHW07] is a programming language that implements the BDI architecture [Rao96] and is one of the dominant implementations of AgentSpeak. It is designed to be easily extended and experimented with.

JASON has a similar programming style to *Prolog*: atoms are represented with any sequence of characters starting with lowercase letters. Atoms are used to represent particular individuals or objects. Logical predicates start with uppercase letters. Components of the JASON language include *beliefs, goals, events, plans*, and *intentions*. Practically speaking, an agent programmer specifies an agent by writing the initial set of beliefs, plans, and goals of an agent in a file with the extension `.asl`.

The agent programmer can specify an environment with all the agents of an environment in a file with the extension `.mas2j`. In the `.mas2j` file, the agent programmer must specify an environment that is implemented in a Java program that extends the `jason.environment.Environment` class where the programmer can define the custom attributes of the environment such as custom actions and percept sources for the agents. The compiled environment class is added to the classpath before running the MAS. The JASON interpreter parses and interprets the agent and environment files into an executable program.

Beliefs form an agents representation of its environment. Beliefs in JASON are represented by predicates of first order logic, for example: $Friend(bob, alice)$. Architecturally speaking, beliefs are stored in the *belief base* of an agent. Beliefs can be added and updated in three main ways:

- By perception - this occurs when the agent *perceives* something in its environment. Beliefs from perception are annotated by $[source(percept)]$, for example, $Friend(bob, alice)[source(percept)]$ which means that the current agent believes that *bob* is a friend of *alice* and it acquired this belief through perception.

- By communication - this occurs when some new information is conveyed by another agent. Beliefs from communication are annotated by the name of the agent that conveyed the information, for example, $Friend(bob, alice)[source(fred)]$ indicates that an agent by the name $fred$ informed the current agent that *alice* is bobs friend.

- By intentions - this occurs when beliefs are added as a result of the execution of a plan. Beliefs from intentions are annotated by $source(self)$.

The belief base of an agent can also be instantiated by listing a set of predicates in the agent program.

Goals represent a desired state of the environment an agent wants to bring about. They have the same syntax as beliefs but are prefixed with either '!' to indicate an *achievement* goal or '?' to indicate a *test* goal. For example, the achievement goal $!B(t)$ indicates that the agent wants to achieve a state of its environment where $B(t)$ is a true belief. A test goal, written as $?B(t)$ indicates that the agent wants to test if the predicate $B(t)$ is a true belief or not given the current state of the environment. Goals can be acquired through communication (an agent can instruct another agent to *achieve* a goal) or intention (as a result of the execution of a plan: a plan may include a test

goal as a prerequisite for actions to be taken). Agent programmers can also specify the goals for an agent in the `.asl` file of the agent.

Events represent a change in an agent's beliefs or goals, for example, the addition of a belief or a goal. An agent reacts to events by executing plans.
Plans have the following syntax:

$$triggering\_event : context <- body$$

triggering_event denotes the event that the plan is meant to handle. *context* denotes the state of the environment that must be satisfied for the plan to be used. *body* denotes the sequence of actions to be taken given that the context is satisfied and the *triggering* event has occurred. An example of a plan is as follows:

$$+!Has(owner, oranges) : notAvailable(oranges, fridge)$$
$$<- .send(supermarket, achieve, Order(oranges, 5)).$$

This plan illustrates part of an agent that handles grocery shopping for its owner. The triggering event is the addition of the goal $+!Has(owner, oranges)$, the context is $notAvailable(oranges, fridge)$, i.e., whether there are oranges in the fridge, and the body or sequence of actions is $.send(supermarket, achieve, Order(oranges, 5)).$[1]. Here we only have one action: the agent telling another agent called *supermarket* to *order* 5 oranges. Presumably, the supermarket agent has a plan with the triggering event $+!Order(Oranges, \_)$ as well. Agent programmers must also specify the list of plans to handle events in the `.asl` file of the agent.

Intentions are instantiated plans: plans where some of the variables have been instantiated. Intentions keep track of the goals the agent is committed to and the course of action it chooses to achieve the corresponding goals Intentions are computed by the JASON interpreter (illustrated in Figure 3.1) during the execution cycle of an agent [2].

Execution Cycle of an AgentSpeak agent

Since JASON is an implementation of AgentSpeak, we provide a brief overview of the execution cycle of an AgentSpeak agent to illustrate the general mechanism of JASON (and other implementations of AgentSpeak) agents. The main components of an AgentSpeak agent include its belief-base, events, plans and intentions.

[1] The periods are part of the syntax.

[2] More details on the JASON programming language and its implementation can be found in [BHW07].

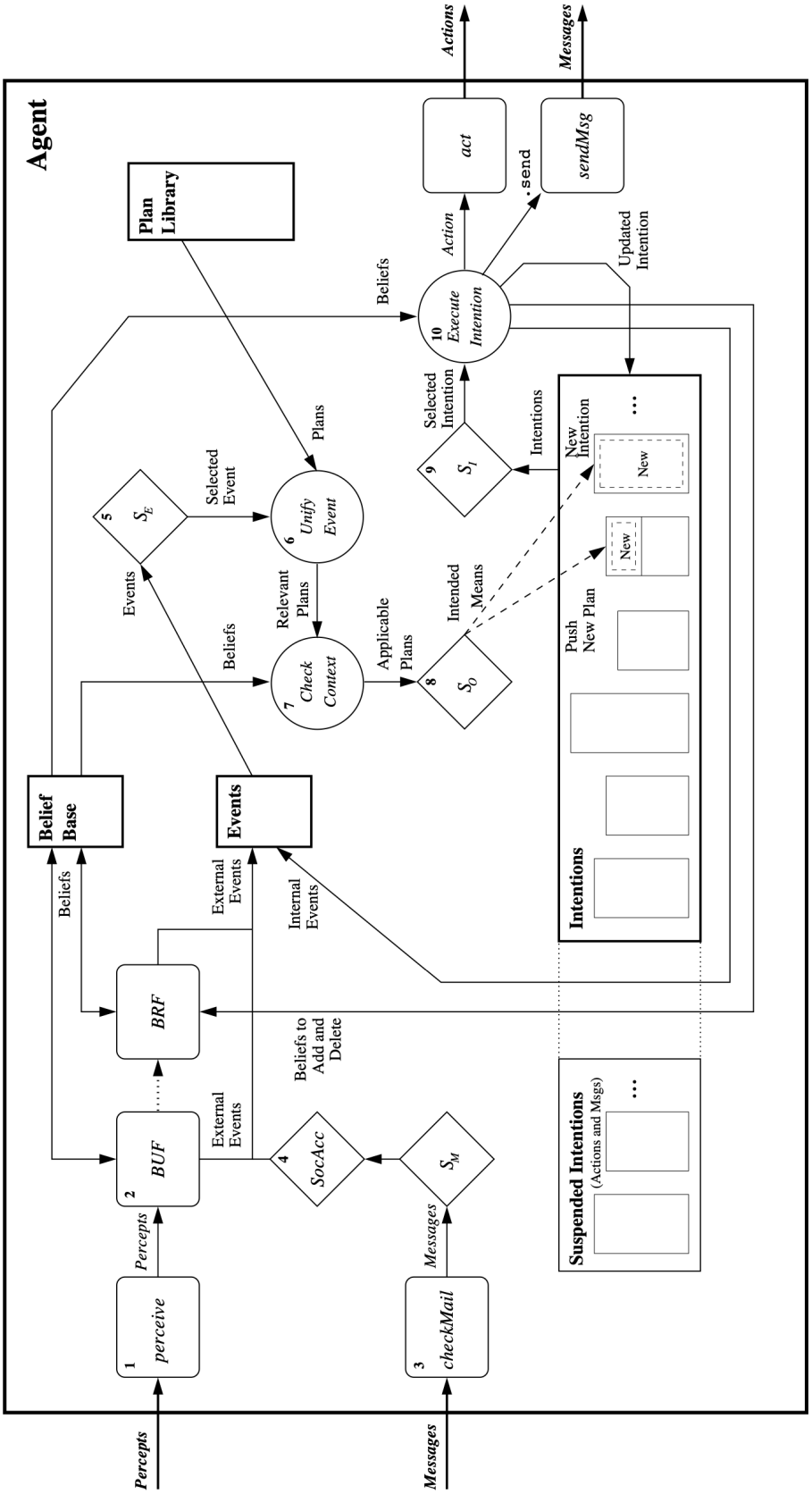

Figure 3.1: Illustration of the JASON interpreter cycle [BHW07]. 'BUF' stands for '*Belief Update Function*'. 'BRF' stands for '*Belief Revision Function*'. 'SocAcc' stands for '*Social Acceptance Function*'. 'S_M' stands for '*Message Selection Function*'. 'S_E' stands for '*Event Selection Function*'. 'S_O' stands for '*Plan Selection Function*'. 'S_I' stands for '*Intention Selection Function*'.

The execution cycle goes through the following steps:

1. *Perceiving the Environment:* This is the first stage of the execution cycle, the agent updates its belief-base with any new states observed from its environment. The agent architecture must include a component that enables the agent to perceive the environment as a list of literals where each literal is a percept that is a symbolic representation of the current state of the environment. As an example, consider a restaurant agent perceiving a customer (identified as p) paying a bill. It may then acquire the belief:

$$PayingCustomer(p)[source(percept)]$$

2. *Updating the Belief Base:* During this stage, the belief-base is updated to reflect the perceived changes to the environment. This process is usually customisable by the agent programmer.

3. *Receiving Communication from Other Agents:* This stage involves the interpreter checking for messages that might have been sent by other agents. This is important as messages sent by other agents may reflect a change in the environment that needs to be updated in the agent's belief base.

4. *Selecting 'Socially Acceptable' Messages:* This stage involves determining messages that can be processed by the agent. There is a customisable selection function that evaluates all messages in an agent's inbox to determine *acceptable* messages. The default implementation of the selection function accepts all messages; however, it can be customised for each individual agent by the agent programmer depending on the needs of the agent. For example, an agent may want to limit its communication with another agent depending on how much the other agent is trusted or depending on whether a goal has been accomplished.

5. *Selecting an Event:* In AgentSpeak, events correspond to perceived changes in the environment or goals. There may be various pending events; however, in each execution cycle, only one pending event is dealt with as various aspects of the environment may change during the first stage of an execution cycle. For example, the list of events in the interpreter of the JASON implementation of AgentSpeak is held in a queue, as such, events are handled and prioritised in a *first-in-first-out* FIFO manner, this means that the selected event is the event

that is always the event at the top of the queue. The event-selection policy is customisable.

6. *Retrieving all Relevant Plans:* Once an event is selected the interpreter proceeds to find a plan that will enable the agent to handle the event by going through all relevant plans in the agent's plan library (the available list of plans is usually created by the agent programmer). A plan is said to be *relevant* if there exists a substitution that allows the *triggering* event of the plan to be unified under the substitution of variables with the selected event. Let the list of events be represented by a set of tuples $\langle d, i \rangle$ where d is a triggering event and i is a corresponding intention. A plan p is of the form $b_1 \wedge b_2 ... \wedge b_n \leftarrow h_1; ...; h_m$ where $b_1 \wedge b_2 ... \wedge b_n$ is a set of belief literals and $h_1; ...; h_m$ is set of goals or actions. A plan denoted p is said to be relevant if there exists a most general unifier σ such that $d\sigma = p\sigma$, i.e., there exists a substitution θ for σ such that $\forall(b_1 \wedge b_2 ... \wedge b_n)\sigma\theta$ is a logical consequence of the set of beliefs (resulting from the update in second phase of the agents execution cycle).

   Example 3.1.1. As an example, consider a restaurant agent with the following plans:

   (a) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad UnhygienicWaiter(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X).$

   (b) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad FoodPoisoning(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X); \; .alertManager().$

   (c) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad ForeignFoodObject(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X); \; .alertManager().$

   All of the plans of the agent are relevant for the triggering event $+!UnhappyCustomer$.

7. *Determining the Applicable Plans:* Plans have a component called *contexts* which often corresponds to a set of boolean expressions that determine whether a plan can be used. Thus after filtering out all the relevant plans, the interpreter needs to determine which plans can be implemented given an event: we call these plans *applicable*. A relevant plan is *applicable* if the conjunction of all the boolean expressions in its context is true, which only occurs if the context is a logical consequence of the belief base.

Example 3.1.2. As an example, consider a restaurant agent with the following beliefs:

(a) $Customer(mary)[source(percept)]$

(b) $Experiences(mary, john)[source(percept)]$

(c) $UnhygienicWaiter(john)[source(percept)]$

and the following plans:

(a) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad UnhygienicWaiter(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X).$

(b) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad FoodPoisoning(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X); \; .alertManager().$

(c) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad ForeignFoodObject(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X); \; .alertManager().$

Only the first plan (a) is applicable given the current set of beliefs of the agent.

8. *Selecting One Applicable Plan:* Intentions are partially instantiated plans (plans whose variables have been initialised). Therefore, an applicable plan is called an *intended means* as the course of action defined by that plan will be the means through which the agent handles the event. After filtering the list of applicable plans, the interpreter uses a selection function to choose which applicable plan the agent will execute. The default selection function chooses an applicable plan based on the order in which they appear in the plan library but is also customisable. The list of plans in the plan library is ordered in the way they appear in the agent's program, therefore agent programmers should typically write applicable plans in descending order of preference or priority for plans with similar triggering events and contexts. Depending on the agent program or flow, several plans may be triggered resulting in several intentions, for example, as part of executing an action for a plan, an agent may add a new goal to its environment.

9. *Selecting an Intention for Further Execution:* Given that at any point in time an agent may have a list of intentions, the interpreter prioritises an intention for

further execution using a selection function for intentions. The default selection function depends on the version of AgentSpeak in use. JASON's selection function follows a round-robin scheduling nature, i.e., each applicable plan is selected in turn, and when selected, only one action in the body of the plan gets executed. The selection function can be customised by the agent programmer.

10. *Executing One Step of an Intention:* The last step of the execution cycle involves the interpreter deciding how to execute a step of the selected intention: recall that an action may involve a belief update or addition of a goal, as such, there is a myriad of variables that need to be taken into consideration when executing a step of an intention which is beyond the scope of this chapter.

The execution cycle may be summarised in three main steps: (1) an update of the agent's beliefs on its environment and other agents, (2) handling one of the possibly many generated events and then, (3) following up on one of the generated intentions, or in other words, executing an action.

The JASON programming language successfully achieves what it is designed for, i.e., implement a practical framework to write AgentSpeak programs in. In addition, JASON aims to be an easily customisable framework, to this end, all the components of the JASON MAS can be easily extended and customised as previously discussed and it is also open-source. JASON has been successfully applied in domains such as energy-saving systems [ECMC14] and modelling and simulation [LDGS11]. One of the limitations of JASON is that does not provide any knowledge extraction or projection techniques that can be used by agents. This can potentially hinder efficient communication among agents in scenarios where agents do not share all of the symbols in their respective belief-bases. An agent intending to convey knowledge that is represented in its belief base using symbols that are not shared by its communication partner(s) is at risk of being misunderstood. Knowledge extraction or projection mechanisms may be useful in such scenarios as agents may be able to convey such knowledge by projecting or extracting the knowledge in terms understood by the agent's communication partner(s). This means that the application of JASON knowledge management domains may be limited by the lack of knowledge extraction mechanisms. Our research addresses this limitation by investigating how existing knowledge extraction techniques can be adapted to the context of agent communication.

Incorporation of OWL: JASDL

JASDL (Jason DL) [KB08] is an extension of the JASON agent framework that incorporates the web ontology language to make improvements to several aspects of JASON, such as querying of the belief-base and plan retrieval.

The main idea is to leverage the expressivity of OWL to increase the expressivity of the belief-base of JASON agents, and in so doing, improve the triggering of plans associated with a goal or a belief. Because beliefs in JASON are written using ground-literals and prolog-like rules, a significant amount of plans need to be explicitly stated in order to capture a variety of belief combinations. The beliefs within JASDL are a combination of assertional statements and terminological axioms. A more expressive belief-base, expressed using an OWL ontology, can potentially reduce the number of belief combinations that may be otherwise needed to be stated explicitly, as beliefs may be inferred using axioms in the ontology.

3.1.2 JADE

JADE [BPR01] is an implementation of the FIPA specifications. FIPA was established in 1996 as an international non-profit association to develop a collection of standards relating to software agent technology. The association is composed of academic and industry experts whose goal was to specify standard specifications for software agent technologies. It has been maintained by its members and board of directors who produce standards and specifications through a series of discussions and debates over the years. At the core of its principles is the standardization of agent technology, but more specifically, structure and semantics required for open collaboration among agents [BPR01]. Agent communication is a core and primary concept of FIPA specifications amongst others such as agent management and agent architecture.

JADE is fully Java-based. In the JADE architecture, the agent platform can be split on different hosts. Each host provides a complete runtime environment for execution lifecycle management of agents and message passing facilities. These hosts are also referred to as *containers*. At least one of the hosts (also referred to as the Agent Management System (AMS)) is responsible for maintaining a registry of all other hosts on the platform so that agents can discover each other. JADE offers an additional boilerplate container called a Directory Facilitator (DF), which maintains a record of the services provided by all of the agents on the same platform. Agents advertise their services by contacting the DF.

JADE agents themselves are single-threaded Java agents. User-defined agents must extend the `jade.core.Agent` class which provides boilerplate methods for an agent. Each agent is provided with a globally unique name by default. A JADE agent is always in one of the following states:

1. Initiated, i.e., not yet registered with the AMS.
2. Active, i.e., registered to the AMS.
3. Waiting, i.e., waiting for something to happen.
4. Suspended, i.e., not executing anything.
5. Transit, i.e., has started a migration process.
6. Unknown, i.e., deregistered from the AMS.

The autonomy of JADE agents is specified through *behaviours*. A behaviour in JADE can be *proactive*, i.e., initialised by the agent, or *reactive*, i.e., performed in response to an event such as a sent message. The java package `jade.core.behaviours` is a sub package of `jade.core` that contains a hierarchy of the core application-independent behaviours. Behaviours are added to agents via the *addBehaviour* method of the core agent class. Behaviours are implemented by extending the `jade.core.behaviours.Behaviour` class which is abstract. The `jade.core.behaviours.Behaviour` class provides a *onStart* and *onEnd* methods where the details of the behaviour can be specified by users. The *action* method in the `jade.core.behaviours.Behaviour` is called by the JADE platform to run the behaviour of an agent.

JADE was developed by and successfully applied to telecommunication systems. JADE benefits from the fact that it is completely Java-based, thus, can be easily adopted by people with Java programming skills because it does not require learning additional syntax or semantics. JADE has been successfully applied in domains such as grid power management [HCZS14, AP19]and security analysis [SCE15].

Adaptation to OWL: AgentOWL

AgentOWL [LBBH06] is a framework designed to facilitate the use of OWL ontologies within a JADE multi-agent system. Agent constructs such as the environment are modelled using a central ontology. The beliefs of the agents are represented using

assertions/instantiations of concepts from the central ontology. AgentOWL thus ensures that agents within the same environment communicate accurately and effectively. However, if agents are required to communicate with agents from other environments, a translation, mapping or projection of the signature from the foreign environment is needed which may require some of the knowledge extraction and knowledge projection techniques discussed and developed in this thesis.

3.1.3 Ontology Based Agent Oriented Programming Frameworks

Semantic web technologies such as the web ontology language (OWL) have been shown to have useful advantages and applications in AOP [KB08, MVBH05]. Adopting ontologies into AOP has consequences on two aspects of AOP: (i) the agent-based systems themselves and (ii) the knowledge/information applications of AOP. MASs can harness domain-specific knowledge in the form of ontologies to make more informed decisions in environments and contexts due to the reasoning benefits that come with ontologies, for example, subsumption checking can help program agents with more expressive beliefs and more concise plans.

AOP frameworks seeking to be enhanced with ontologies face interoperability issues that arise from the numerous and heterogeneous ontologies available for a given domain of application. This means that the signature of the ontologies of the agents may be disjoint or overlapping, which may lead to issues such as miscommunication, semantic ambiguity, and false interpretations of the concepts in the ontologies. Typically, standardization of a domain could help alleviate this issue by defining a uniform vocabulary used in the domain; however, not every domain has standardizations and a domain may have several standardizations. Even at the scale of nations, humans seem to have a difficult time coming to a consensus for a universal system of measurement (e.g., metric vs imperial). This is not necessarily a disadvantage as different systems have their advantages and disadvantages, as do standardizations and signatures. The multiplicity of such signatures is not too much of an issue for humans as we have the ability to translate, transform and adapt knowledge. Therefore, for AOP and MASs to overcome these interoperability issues, AOP and MASs need to adopt and simulate techniques that enable agents to translate, transform and adapt knowledge. Ideally, such techniques would have specifications that can be described in a way that can be implemented or integrated into AOP frameworks or MASs using speech acts or other AOP components. The ontology services of CooL-AgentSpeak [VDRA11], for example, also include provisions for ontology alignment [ES07] to help mitigate the issues

of heterogeneous signatures used in the agent's ontologies.

Here we discuss the two relevant agent frameworks that have ontologies as the main component to illustrate the effects of integrating ontologies into a MAS framework.

AgentSpeak DL

AgentSpeak DL [MVBH05] is an AOP language based on description logic (DL) rather than classical (predicate) logic which is advantageous because the commonly used DLs are decidable fragments of predicate logic. AgentSpeak DL expands the belief base of an agent to contain complex concept assertions which yield several advantages including [MVBH05]:

1. More expressive queries to the belief base.

2. The belief update process of the execution cycle is enhanced as the consistency of a belief addition can be verified using ontological reasoning.

3. Search for relevant plans given an event is more flexible as relevant plans are not only dependent on unification but also subsumption relations between concepts.

4. Knowledge sharing may be facilitated among agents by using ontology languages such as OWL.

A motivation of AgentSpeak-DL is to enhance AgentSpeak programs with belief bases that have a more sophisticated structure that can perform in complex environments. The chosen DL for AgentSpeak DL is $\mathcal{ALC}$. Similar to AgentSpeak, in AgentSpeak-DL, beliefs, events, and contexts, are represented using ABox statements. The actual details of the system are beyond the scope of this chapter; we, however, provide an overview of the main benefits of ontological reasoning to the execution cycle of an AgentSpeak agent.

1. *Impact on updating the belief base:* In AgentSpeak, the only implication of the addition of a belief to the belief base is that a possible new event is added to the set of events for the agent to deal with. In AgentSpeak-DL, the addition of beliefs (ABox assertions) to the agent's belief base is only allowed if the result is consistent with the agent's background ontology.

2. *Impact on retrieving relevant plans:* In AgentSpeak, a plan is said to be relevant

if its specified triggering event can be unified with the selected event. The resulting implication for agent programmers is that they have to write plans specifically for each possible type of event. In AgentSpeak-DL, unification is not the only criteria for determining if a plan is relevant, subsumption also qualifies a plan for being relevant, i.e., finding plans whose triggering event subsumes (with respect to the agent's background ontology) the selected event and unifying terms for the selected event and the plans triggering event. This reduces the workload on agent programmers as they can use subsumption relations in ontologies to reduce the workload of having to write plans for each type of possible event.

Example 3.1.3. As an example, consider a restaurant agent with the following plans:

(a) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad UnhygienicWaiter(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X).$

(b) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad FoodPoisoning(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X); \; .alertManager().$

(c) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad ForeignFoodObject(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X); \; .alertManager().$

Given the background ontology $O = \{ForeignFoodObject \sqsubseteq FoodIssue, FoodPoisoning \sqsubseteq FoodIssue, RudeWaiter \sqsubseteq PoorService, UnhygienicWaiter \sqsubseteq PoorService\}$. The set of plans can be simplified to:

(a) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad UnhygienicWaiter(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X).$

(b) $+!UnhappyCustomer(X) : Customer(X) \quad \& \quad FoodIssue(Y) \quad \& \quad Experiences(X,Y) < -.offerDiscount(X); \; .alertManager().$

Because $O \models ForeignFoodObject \sqsubseteq FoodIssue, FoodPoisoning \sqsubseteq FoodIssue$.

3. *Impact on determining applicable plans:* In AgentSpeak, a plan is applicable if its context is a logical consequence of the belief base. In AgentSpeak-DL, a plan

is applicable if it is relevant and its context is a consequence of the agent's belief base and background ontology.

4. *Impact on selecting applicable plans:* In AgentSpeak-DL the selection function for applicable plans can be further customised using ontological reasoning. For example, the selection function may be designed to prioritise applicable plans according to *specificity*: the lower an applicable plan is in the subsumption hierarchy of the agent's ontology, the more specific it is.

Other stages of the execution cycle are not affected by ontological reasoning.

There are several implementations and extensions of AgentSpeak-DL. Cool -AgentSpeak [VDRA11] for example, is an extension of the JASON agent platform which extends AgentSpeak-DL with features such as plan exchange and ontology services that aim to make AgentSpeak-DL FIPA compliant.

ANEMONE

The ANEMONE system [VDBD+07a] consists of a set of protocols (or layers) to be used to enable flexible information exchange in agent communication using a minimal common ontology. An intriguing motivation of the system is that approaches such as standardisation and ontology matching are not suited for the semantic integration problems that are faced in certain applications of MASs. Its specification [VDBD+07a] is motivated by the specific example of an open community of agents that periodically download news articles in RSS news feeds on different topics.

The system consists of three main layers designed to facilitate flexible communication. The first layer is the *normal communication protocol* which guides communication when there is no disparity between the ontologies of the communicating agents. If any misunderstanding occurs, the agents switch to the middle layer in the protocol, the *concept definition protocol*. In the concept definition protocol layer, agents resolve misunderstandings of concepts by exchanging concept descriptions. The descriptions of concepts are expressed using concepts already understood by both agents. If the concept descriptions exchanged do not adequately describe the original misunderstood concept, the agents switch to the lowest layer in the protocol the *concept explication protocol*. In the concept explication protocol, the agents explain the meanings of concepts using non-symbolic communication, for example, exchanging instances or examples of the concept.

The various protocols specify in-depth information on how agents should send, interpret, and evaluate messages. Agents also need to be able to detect when there is a misunderstanding according to the layers.

The protocols in the ANEMONE system can be implemented in AOP frameworks such as AgentSpeak by programming custom speech-acts, actions into agents or other aspects of the execution cycle outlined previously.

JADL & JIAC

The JIAC (Java Intelligent Agent Componentware) framework is built upon the JADL (JIAC Agent Description Language) specifications and is intended to be robust enough to handle industry applications such as energy management systems [KHA06, LKK15].

JADL specifies a MAS architecture that consists of *plan elements*, *rules*, *ontologies*, and *services*. Within a JADL-based MAS, the state of the world is represented using assertions in ontologies and an AgentBean: a Java program through which an agent may interact with external entities which may be anything varying from databases to mechatronic components of a robot.

By means of a situation calculus and a three-valued logic, agents can reason about their environment. The outcome of such reasoning may be translated and stored into OWL ontologies. Plan elements include actions that describe the functional capabilities of an agent, which may then be used to automatically construct plans for the relevant goals of the agent.

The framework incorporates FIPA speech acts to enable communication via services which the agents may interact with.

It is not clear whether or not there are assumptions in the JADL framework that make it so that agents are guaranteed to utilise the same ontology signatures or if all ontologies in a JADL-based MAS are public and accessible to every agent. However, the fact that agents interact with external components such as databases makes it likely that an agent may encounter a symbol that is unique to its experience of the world. Thus, having the ability to translate such unique symbols into terms that other agents understand or are familiar with is imperative for effective communication within the framework.

3.2 Agent Communication

Agent programmers typically bear the responsibility of writing agent programs that ensure that an agent takes appropriate communication actions. The BDI architecture and AgentSpeak language are primarily concerned with the internal structure of agents and do not address the issue of communication interaction among agents [BHW07]. This is in contrast with the FIPA specifications and architecture which has agent communication and agent interaction at the center of the issues it aims to address. Extensions of AgentSpeak such as JASON address such issues carefully with respect to the BDI architecture.

Some MAS frameworks provide *agent communication languages* (ACL(s)) that agent programmers can use to implement communication in agent-based systems. ACLs are based on speech-act theory [SS69]: a linguistic philosophy on how speech can convey both *information* and *action*. In the context of MASs, agents use speech acts to attempt to change a state of its world including its beliefs. The various types of speech acts are typically referred to as *performatives* in the context of MASs. In this section, we focus on ACLs and frameworks that implement them. Specifically, we focus on KQML (Knowledge Query Manipulation Language) and the FIPA Agent Communication Language (FIPA-ACL) which are foundations for ACLs in AgentSpeak and FIPA-compliant AOPs. We provide a brief overview of each of these languages in this section, followed by an overview of agent communication as implemented in the MAS frameworks JASON and JADE.

3.2.1 Speech Acts

The underlying mechanism and theory of all ACLs is speech-act theory [SS69]. Speech act theory is concerned with the elements of speech and language that result in actions. Speech act theory was a successful effort at addressing uses of language besides making factual assertions. In the context of MAS, this can be seen as another mechanism an agent can use to bring about a desired change or update in its environment. In everyday conversation, for example, we bring about changes in our world by perhaps asking people to do things for us. As an example, you may ask another person to close a window upon feeling chilly in a room. Under the perspective of speech act theory, we view the question/request as an action that results in a window being closed. In the KQML ACL and JASON perspective, an agent requesting another agent to close a window can be emulated using the *achieve* performative. As an example, consider

MAS for a care home which consists of a user agent which monitors patients conditions such as their body temperatures and a smart home agent which controls the room and environment. The user agent may request for a window to be closed as part of a plan to handle dropping body temperatures as follows:

$$.send(smart\_home\_agent, achieve, closeWindows)$$

The environment in question is not limited to the external environment but the internal environment as well. For example, an agent may ask another agent a question so as to update its beliefs. This is also analogous to asking questions to improve knowledge or clarify statements in everyday conversation. This is premised on the assumption that the other person or agent in question responds as we expect.

Speech acts comprise of three main notions:

1. *Locutionary act:* Locutionary acts are the actual utterances and their semantics or apparent meaning. For example, the statement *' Are the windows open?'* is a locutionary act itself. Locutionary acts usually have a corresponding *Illocutionary act*.

2. *Illocutionary act:* The illocutionary act is the implied meaning or request that is invoked by a locutionary act. For example, the locutionary act *'Are the windows open?'* may imply the illocutionary act *'Can you please close the windows?'*.

3. *Perlocutionary act:* The percolutionary act is the actual act or effect that is wished to be brought about by the locutionary act or illocutionary act. For example, closing-the-windows or closed-windows is the act or state wished to be brought about by the locutionary act *'Are the windows open'* or illocutionary act *'Please close the windows'*.

Agent communication is centered around illocutionary acts, which are usually referred to as *performatives*. Following Searle [SS69] there are five main types of illocutionary acts:

1. Directive acts - acts that attempt to get the hearer to perform some action. In MASs, assertive acts are modelled using performatives such as *achieve*.

2. Assertive acts - acts that assert facts about the world. In MASs, assertive acts are modelled using performatives such as to *inform*.

3. Commissive acts - acts which commit the speaker to accomplish a task.

4. Expressive acts - acts that express a speaker's states.

5. Declarative acts - acts that are performed by virtue of the utterance, for example declaring war.

As seen, most AOP frameworks come with boilerplate communicative acts; however, different domains of application may require custom speech acts and protocols. The format and nature of speech acts dictate the range, expressivity and flexibility of knowledge/information exchange between agents.

3.2.2 KQML

KQML [FFMM94] is a product of the DARPA Knowledge Sharing Effort which is aimed at developing techniques and software for sharing and reuse of knowledge in large-scale knowledge bases. KQML specifies both a message format (i.e., KIF) and a message-handling protocol to support knowledge sharing among agents. KQML defines an extensible set of performatives which specify an agent's intentions in sending a message to another agent.

The KQML performative *tell*, for example, is used with the intention of updating the receiving agent's beliefs. A KQML message consists of a performative and its associated arguments which consist of the content of the message and other optional arguments which may describe the message content and identifiers of the sender and receiver.

KQML message format

(*performative*
:*message-content*
:*optional-argument-1*
:*optional-argument-2*
...
:*optional-argument-N*)

For example, the following KQML message:

```
( ask-one
        :content (PRICE TESLA ?price)
        :receiver stock-server
```

:language LProlog
:ontology NYSE-TICKS)

may be sent by an agent wishing to know the price of TESLA stock. The performative in this context is *ask-one*, which is used to ask a question that requires exactly one reply. The *content* argument specifies the message content while the *receiver* argument identifies the intended recipient of the message. The language argument specifies the language in which the content is expressed and the ontology argument specifies the terminology and semantics of interpretation for the message content. Other KQML performatives include *tell, ask, ask-all, achieve, generate, subscribe, advertise and monitor*. More complex domains and environments may require different performatives which can be specified by a MAS designer or agent programmer.

KQML has been successfully implemented in MAS experiments including a planning and scheduling system for military transportation logistics [FFMM94] and an information MAS called "CoBASE" [FFMM94].

3.2.3 FIPA ACL

The FIPA standard for agent communication is closely related to KQML but differs in its set of performatives and semantics. FIPA ACL Messages have the same structure as KQML messages and are extensible as well. Agent communication is broken down into three different aspects:

1. Communicative Acts.

2. Interaction Protocols.

3. Content Languages.

Communicative acts are analogous to performatives. Like KQML, FIPA specifications provide a set of communicative acts to satisfy the communication needs of a wide set of problems. FIPA specifications define a logical and semantic definition for each communicative act for sender and receiver agents in terms of BDI semantics. Just like KQML, communicative acts can be extended by MAS designers or agent programmers. Examples of FIPA ACL communicative acts include *accept-proposal, agree, cancel, confirm, disconfirm, inform, inform-if*.

The interaction protocols of the FIPA ACL aim to model social behaviour among

FIPA compliant agents. There are several FIPA ACL interaction protocols [3], examples include the *request interaction protocol*, *query interaction protocol*, *request when interaction protocol* to name a few. All interaction protocols specify conditions and procedures for agents to interact for a specific scenario. For example, the request interaction protocol specifies how one agent can request another agent to perform an action. Interaction protocols also specify how communicative acts are to be used as part of message sequences or workflows.

The FIPA ACL specifications include specifications for the content language *FIPA Semantic Language* (FIPA-SL). Several content languages may exist within a FIPA compliant MAS. The FIPA-SL specification includes language syntax and its associated semantics; however, the lexical and grammar specifications are beyond the scope of this chapter.

3.2.4 Agent Communication in JASON

JASON agents send messages using the following syntax:

$$.send(receiver, performative, message\_content)$$

In the JASON context *.send* is an internal action available to every agent in any environment. Another internal action for communication is *.broadcast* which is used to broadcast messages to all of the agents in an environment which has the following syntax:

$$.broadcast(performative, message\_content)$$

An agent's execution cycle eventually reaches a point where it checks for messages that may have been received from other agents. Messages received have the following structure

$$\langle sender, performative, content \rangle$$

This structure is not strict and may have additional arguments depending on the performative used[4]. JASON comes with a set of performatives which are a subset of performatives specified by the KQML language, they include *tell, untell, achieve, unachieve, askOne, askAll, tellHow, untellHow, askHow*. Agent programmers can customise the interpretation of performatives using AgentSpeak plans with special annotations or by

[3]They are all accessible at `http://www.fipa.org/repository/ips.php3`.

[4]Details of the semantics of how JASON automatically interprets messages can be found in [VMWB07].

customising the JASON interpreter. Additionally, agent programmers may also specify other communicative actions by creating custom internal actions for the agents.

3.2.5 Agent Communication in JADE

FIPA ACL messages in JADE are specified by creating an instance of the `jade.lang.acl.ACLMessage` class. Using the JADE `ACLMessage` class, an agent can specify parameters such as the communicative act associated with the message, the message recipient, an identifier for a message that is being responded to, and the message content. This is not an exhaustive list; however, the intuition is that it is a template Java class that provides a way to model KQML-style messages using setters and getters. Recall that agents in JADE are java classes extending the `jade.core.Agent` class. The Agent class has a built-in `send()` method which takes in an instance of the `ACLMessage` class, that enables agent programmers to specify communicative actions and behaviours for their agents.

3.3 Related Work

3.3.1 Ontology Negotiation

Ontology negotiation is the process by which communicating agents may try to resolve misunderstandings that occur during communication. Ontology negotiation is an AOP paradigm that involves agents autonomously interacting to solve problems with respect to ontologies. The term *ontology negotiation* was originally coined by Bailin and Truszkowski [BT02] and motivated by trying to solve agent communication issues involved at the boundaries between scientific disciplines. For example, in an attempt to achieve a goal, an agent managing a global warming knowledge base may seek to elicit information from an agent managing an ecology knowledge base. In this section, we provide an overview of some existing efforts to implement ontology negotiation.

The original proposal for ontology negotiation [BT02] presents two agent scenarios that are frequently referenced: (i) *Agents in Hollywood: The Cocktail Party Metaphor* and (ii) *Back at the Office: The Information Retrieval Task*. We provide an overview of the *information retrieval task* as it is closer to our interest in knowledge and information transformation and management.

The information retrieval task metaphor is used to understand various levels of discourse that may arise when an agent tries to request the help of another agent in

a task involving information retrieval. The metaphor is modelled after how humans embark on similar conversations when they have different specialisations. For example in an organisation, people in different departments have different specialisations and thus have access to different repositories of information. Agents (software and human) may start these conversations by describing what they specialise in and what kind of information they have access to. The most shallow level of conversation involves agents interacting as a form of courtesy.

However deeper levels of conversations may stem from this conversation, for example, if the agents tried to *understand* what they do, the services they provide, or the tasks they engage in. If the agents wanted to make practical use of that information (what they do, services they provide, etc), a deeper level of conversation is needed.

This scenario highlights one of the central motivations of *ontology negotiation*: conversations have varying levels of engagement. This scenario is broken down into 5 levels as the dialogue between the agents progresses:

1. *Respond to Query with Minimal Effort*. Like humans, agents may behave (due to their goals and effort) in a way that relieves them of a query request with minimal effort which may result in a superficial level of returned information. This may result in a poor understanding of the other agent's goals or constraints and as a result, the returned query may not satisfy the needs of the requesting agent.

2. *Express Dissatisfaction with Initial Response*. Upon realising the inadequacy of the returned information, the requester may reformulate the query by adding (being more specific) or removing (being more general) constraints. The agent may even choose to reformulate the constraints using a different set of terms or try to explain why the returned information was inadequate all of which may lead to the helper agent understand better the needs of the requester or request for further clarification.

3. *Enlist Additional Help*. An improved understanding may lead to the helper agent contacting another agent for help: any resulting information returned may be forwarded to the original requester agent.

4. *Negotiate Ontologies to Construct Integrated Story*. The first and second helper agents may then need to construct an integrated response to the original query: in communicating, they may discover that their ontologies are quite heterogeneous in the sense that there are differences in concepts in their respective ontologies.

5. *Evolve Ontologies*. The two helper agents may then *negotiate* and reach a shared understanding of concepts that were previously understood differently or not held in common. Agents may explore the history of a concept being negotiated. Concepts not understood may be defined in more basic terms and stand a better chance of being shared. The process may be iterated until the agents find fine-grained concepts in common. Given a common concept name, the second agent may add it to its ontology, find synonymous a concept name in its ontology, or respond with a different understanding of the common concept name. Following [BT02], "The third case represents the greatest potential for ontology evolution. If the two understandings of the same term are similar but not identical, then the agents have different interpretations of the world, and not just different foci of attention."

Levels four and five highlight another central motivation of *ontology negotiation*: different levels of conversational engagement require complex communication tools.

Some levels of conversation can be compared to the layers of communication in the ANEMONE system. Levels 1 and 2 correspond to the normal communication protocol of the ANEMONE system: here agents communicate normally and understanding is presumed; however, also within the normal communication protocol, one of the communicating agents may detect a misunderstanding and seek to escalate to a deeper level of conversation which corresponds to level 2 where an agent may express dissatisfaction. Levels 4 and 5 can correspond with either the concept definition protocol or the concept explication protocol: exchanging definitions can be a form of ontology negotiation and so can mapping concepts using instances or examples (i.e., the concept explication protocol).

To the best of our knowledge, besides [VDBD$^+$07b, TS19], there is no work that addresses definitions in the context of agent communication and DL ontologies. However, a lot of work has been done in terms of mapping concept names in heterogeneous ontologies in the field of *ontology alignment*, also referred to as *ontology matching*. In the ANEMONE system *ontology alignment* techniques fit into the concept explication protocol.

3.3.2 Ontology Alignment

Ontology alignment attempts to establish correspondences (i.e., mappings between corresponding concepts) in heterogeneous ontologies. Establishing such correspondences is not trivial and can be approached in different ways. Roughly speaking, the task of constructing an alignment between two ontologies O_1 and O_2 involves finding a set of $\langle \equiv, \sqsubseteq, \perp^5 \rangle$ relations between pairs of symbols $\langle S, S' \rangle$ where $S \in sig(O_1)$ and $S' \in sig(O_2)$. Constructing such alignments is often non-trivial [ES07]. Several strategies may be employed to create correspondences depending on the nature of the ontologies in question. An alignment strategy details which alignment techniques should be used and how they should be used. Techniques can range from comparing instances of concepts (as proposed in the concept explication protocol of the ANEMONE system) to string similarity measures, to using natural-language-processing methods such as the *word embeddings* [ZWL+14]. Ontology alignment is a research area in its own right and has applications that span beyond agent communication. Laera et al. [LBT+07] address ontology alignment in the context of agent negotiation. The employed approach involves *argumentation*: where each agent can decide and propose correspondences according to its preferences. Depending on the architecture of negotiation correspondences may or may not be publicly available to the agents. In the case where correspondences are only partially available, a type of protocol is required for the agents to share correspondences. This is the issue addressed by Payne et al. [PT14] where the agents share correspondences through an *inquiry dialogue*. The inquiry dialogue allows agents to assert, reject, and counter correspondences shared by other agents taking into consideration correspondences that may have been acquired from previous encounters with other agents.

A similar approach to alignment is learning mappings through examples [VDBD+07b, ADMF13]. This approach assumes a closed world of the agents and relies on the agents exchanging positive and negative examples of concepts they wish to align.

Background on Dominant Approaches

Most alignment techniques utilise lexical, structural, or reasoning techniques or a combination of such techniques [SJRG17a]. By lexical techniques, we refer to techniques that employ the use of language string similarity measures to establish correspondences among symbols in the ontologies that are to be mapped. Lexical techniques alone are

[5] Here $\perp$ is shorthand for 'disjoint'.

not sufficient as they do not take into account how concepts and symbols in the ontology are related. Ignoring such relations of the ontology can lead to mappings that cause inconsistencies or result in unsatisfiable symbols. Thus, alignments are usually evaluated on whether or not they cause inconsistencies, unsatisfiable symbols, or introduce potentially erroneous relations among concepts.

An alignment that causes inconsistencies or new unsatisfiable symbols in either of the mapped ontologies is said to be *incoherent*, and *coherent otherwise*. Semantic changes are captured using the notion of *deductive difference*. Let O_1 and O_2 be ontologies aligned via $\mathcal{A}$. The deductive difference between $O_i \in \{O_1, O_2\}$ and $O_1 \cup O_2 \cup \mathcal{A}$ is a summary of the semantic changes introduced by the alignment $\mathcal{A}$, thus debugging of alignments can be narrowed down to axioms in the deductive difference. Let O_1 and O_2 be ontologies, the deductive difference between O_1 and O_2 with respect to O_1 is the set of all axioms α such that $sig(\alpha) \subseteq sig(O_1)$, $O_2 \models \alpha$, and $O_1 \not\models \alpha$. Note that by this description, the deductive difference is a possibly infinite set. In practice, we are interested in a finite subset that encapsulates the deductive difference.

Definition 29 (Deductive Difference)**.** *Let O and O' be two ontologies, let $\Sigma = sig(O) \cap sig(O')$. We define the deductive difference between O and O' with respect to O to be a finite set of axioms (denoted $diff(O, O')$) that entails all axioms of the form $C \sqsubseteq D$ such that $O \not\models C \sqsubseteq D$, $O' \models C \sqsubseteq D$, $sig(C) \subseteq \Sigma$ and $sig(D) \subseteq \Sigma$.*

We extend the notion of deductive difference to ontologies and mappings as follows: let O_1 and O_2 be ontologies aligned via $\mathcal{A}$. For any ontology $O_i \in \{O_1, O_2\}$, by deductive difference, we refer to the set of all axioms α such that $sig(\alpha) \subseteq sig(O_i)$, $O_1 \cup O_2 \cup \mathcal{A} \models \alpha$, and $O_i \not\models \alpha$ [SJRG17b].

Unfortunately, capturing the deductive difference between ontologies more expressive than $\mathcal{EL}$ can be computationally expensive, and is undecidable in DLs as expressive as $\mathcal{ALCQIO}$ [SJRG17b]. In the ontology matching literature, an approximate version of the deductive difference that focuses on the class hierarchy of the input ontologies has thus been adopted as a means of detecting the semantic changes induced by alignments.

Definition 30 (Approximate Deductive Difference [SJRG17b])**.** *Let A, B be named concepts (including $\top, \bot$), Σ be a signature, O and O' be two ontologies. We define the approximation of the Σ-deductive difference between O and O' (denoted $diff_{\Sigma}^{\approx}(O, O')$) as the set of axioms of the form $A \sqsubseteq B$ satisfying: (i) $A, B \in \Sigma$, (ii) $O \not\models A \sqsubseteq B$, and (iii) $O' \models A \sqsubseteq B$.*

Consistency and Conservativity violations

In general, the notions of *conservativity violations* and *consistency violations* [SJRG17b, JRGHB11] are designed to capture and evaluate mappings within alignments that may cause inconsistencies or result in potentially erroneous semantic consequences (such as new relationships among concept names).

Consistency violations in an alignment indicate that the alignment causes one or both of the input ontologies to be inconsistent or contain unsatisfiable concepts. Conservativity violations in an alignment indicate that the alignment introduces new semantic relationships in the vocabulary of one of the input ontologies. There are two variants of conservativity violations, namely (i) subsumption violations (ii) equivalence violations. Intuitively, subsumption violations indicate that a potentially erroneous subsumption between named concepts is induced by an alignment while equivalence violations indicate that an potentially errorneous equivalence between named concepts is induced by an alignment.

Definition 31 (Conservativity Violation [SJRG17b]). *Let O_1 and O_2 be ontologies mapped by a coherent alignment $\mathcal{A}$, let $\Sigma = sig(O_i)$, let $O_i \in \{O_1, O_2\}$, and let A and B be atomic concepts in Σ. We define two sets of violations of $O_1 \cup O_2 \cup \mathcal{A}$ with respect to O_i (the anchor ontology):*

- subsumption violations, *denoted $subViol(O_i, O_1 \cup O_2 \cup \mathcal{A})$, as the set of $A \sqsubseteq B$ axioms satisfying: (i) $A \sqsubseteq B \in diff^{\approx}_{\Sigma}(O_i, O_1 \cup O_2 \cup \mathcal{A})$*[6]*, (ii) $O_i \not\models B \sqsubseteq A$, and (iii) there is no C in Σ such that $O_i \models C \sqsubseteq A$, and $O_i \models C \sqsubseteq B$*

- equivalence violations, *denoted as $eqViol(O_i, O_1 \cup O_2 \cup \mathcal{A})$, as the set of $A \equiv B$ axioms satisfying (i) $O_1 \cup O_2 \cup \mathcal{A} \models A \equiv B$, (ii) $A \sqsubseteq B \in diff^{\approx}_{\Sigma}(O_i, O_1 \cup O_2 \cup \mathcal{A})$ or $B \sqsubseteq A \in diff^{\approx}_{\Sigma}(O_i, O_1 \cup O_2 \cup \mathcal{A})$*

Subsumption violations are intended to capture new subsumption relationships between concept names that were previously unrelated while equivalence violations are intended to capture new subsumption relationships between concept names have already been involved in a subsumption. Subsumption violations are based on the assumption that "*all the direct siblings in a well-defined is-a hierarchy should be disjoint*", otherwise known as the *assumption of disjointness* [Sch05]. The assumption of disjointness is the motivation behind the third condition ('there is no C in Σ such that $O_i \models C \sqsubseteq A$, and $O_i \models C \sqsubseteq B$') of subsumption violations. We *assume* that if there

[6] Observe that this means $O_i \not\models A \sqsubseteq B$ following the definition of deductive difference.

is a concept C that is included in both A and B, then A and B intersect on some interpretation and that C is a witness concept of this intersection. If there is no witness to support that A and B intersect on some interpretation, then we assume that A and B are disjoint, per the *assumption of disjointness* [Sch05].

It is important to note that conservativity violations do not necessarily indicate that a mapping is wrong, a conservativity violation may indicate that the information in one of the aligned ontologies is incomplete, and it is usually up to ontology engineers, or agents to determine whether a conservativity violation indicates an error or additional information.

Detecting and Minimising Violations In order to minimise conservativity and consistency violations, several alignment techniques employ additional semantic techniques to refine mappings generated by lexical techniques [SJRG17b]. Semantic techniques include the use of structural techniques (i.e., evaluating the syntactic structure of the ontologies to be aligned), or reasoning techniques (i.e., evaluating the semantic structure of the ontologies by using standard OWL reasoners such as Hermit). In the process of employing semantic techniques to refine mappings, alignment techniques generate *repairs* to the generated mappings.

Given an alignment $\mathcal{A}$ that causes conservativity or consistency violations, a repair $\mathcal{R}$ to $\mathcal{A}$ is a set of proposed removals (or alterations) to mappings in $\mathcal{A}$ to minimise the violations caused by $\mathcal{A}$.

LogMap [JRCG11] for example, is an alignment tool that employs the use of both lexical and semantic techniques to build alignments. LogMap starts by generating mappings using lexical techniques. It then refines the generated mappings by projecting the input ontologies and generated mappings into a set of propositional horn clauses and then implements a satisfiability solver to detect and construct repairs for both consistency and conservativity violations. As a further refinement, LogMap creates a *semantic index* induced by concept hierarchies of the input ontologies to filter out unreliable mappings. Roughly speaking, the semantic index of an ontology is a graph that is built using the class hierarchy of each input ontology in which nodes correspond to concept names and edges correspond to subclass relationships. To refine mappings, LogMap constructs additional edges (induced by the candidate mappings) between the vertices in both semantic indices of the inputs ontologies and then uses a traversal algorithm to detect mappings that cause conservativity or consistency violations.

AgreementMakerLight (AML) [FPS+13] is another alignment tool that employs

semantic techniques to minimise conservativity and consistency violations. Similar to LogMap, AML creates a propositional horn encoding of the input ontologies and the mappings to detect incoherent concept names and mappings that cause incoherences. Unlike LogMap, AML employs further heuristics based on the confidence values of the mappings to create repairs. Other matchers that employ semantic techniques include Lily [Wan10] which uses a search engine to build semantic index and ContentMap [JRGHL09] which employs justification-based repairs.

Chapter 4

Setting the Scene for the rest of the Thesis

In Chapter 3 we discussed agent-based frameworks that include elements of logic based reasoning or ontologies. Some of the frameworks discussed (including JASON and JADL & JIAC) assume (sometimes explicitly) that all symbols used in representing each individual agent's knowledge are known and accessible to all agents. Other frameworks such as JADE (and other FIPA-compliant frameworks) may use logic-based knowledge bases to establish a common vocabulary for agent communication. A few of the frameworks discussed, such as Cool-AgentSpeak and ANEMONE, do not assume that all symbols used in representing each individual agent's knowledge are known and accessible to all agents and provide mechanisms for agents to convey knowledge to each other using symbols understood by all communicating partners. In Chapter 2 we discussed logic-based approaches for knowledge extraction from knowledge bases. While some agent frameworks do not assume that all agents have a common understanding of all the symbols used in knowledge representation, we observe that knowledge extraction techniques have not been studied in the context of such agent frameworks (specifically agent frameworks that integrate ontologies), despite the fact that some of the logic-based knowledge extraction techniques (such as strongest necessary conditions) are developed for the specific case of agent communication [Lin01, DLS01, DLS07]. Furthermore, while some of the logic-based knowledge extraction techniques are developed for agent communication, we find the study of such techniques to be primarily theoretical in nature (i.e., lacking implementations, evaluations or empirical studies).

Practical implementations are required for the evaluation and analysis of any algorithm. Our research primarily focuses on description logics, and as such, we are primarily interested in methods and implementations of knowledge extraction techniques that can be implemented or adopted for description logics. Some of the methods for extracting and projecting knowledge (such as uniform interpolation and module extraction) discussed in Chapter 2 have existing formal specifications and practical implementations in description logics. Other knowledge extraction methods such as Craig interpolation and strongest necessary conditions lack formal specifications and practical implementations in description logics. In a later section (Section 9.2) we address and discuss how knowledge extraction techniques that have practical implementations may be adopted into agent communication. In Chapter 5 we discuss logic-based extraction methods that either lack implementation for description logics or are novel. Specifically, we are interested in techniques that can be used to convey uncommon/inaccessible symbols using only terms that are shared/accessible by agents and their communication partners. Because we require that all conveyed knowledge should be represented only using symbols that all communicating agents understand, uniform interpolation will be an important aspect of our implementation and evaluation, as a uniform interpolant is a restriction of all the knowledge in a knowledge base with respect to a given signature.

Of the existing and discussed logic-based knowledge extraction techniques, strongest necessary conditions and weakest sufficient conditions are the most commonly associated with agent communication [Lin01, DLS01, DLS07, SLZ04]. Strongest necessary conditions and weakest sufficient conditions have been studied and implemented in propositional logic [Lin01], first-order logic [DLS01], and second-order logic [DLS07, Wer12, Wer14]; however, we find that description logic adaptations are lacking. Most of the existing implementations of strongest necessary conditions and weakest sufficient conditions are based on uniform interpolation [Lin01, DLS01, DLS07] or *projection* and *circumscription* [Wer12, Wer14], thus further highlighting the importance of uniform interpolation in this research.

4.1 DL of focus ($\mathcal{ALC}$)

Several practical implementations for extracting uniform interpolants exists for description logics as expressive as $\mathcal{ALCOQH}$, these include LETHE [PK15a] and FAME [YZ18]. However, LETHE is the choice of implementation that all the tools developed

in this chapter are based on because its implementation is the only one that supports general TBoxes and is guaranteed to extract interpolants by extending the language with fixpoint operators. The completeness of the uniform interpolation method used is imperative to providing a solid foundation for all empirical evaluations: any empirical evaluation that is performed using an incomplete interpolation method gives room for doubt of the validity of any empirical evaluation or analysis. If an incomplete uniform interpolation method (i.e., a uniform interpolation method that sometimes returns incomplete uniform interpolants) is used, and a negative result is obtained, it would be unclear as to whether the negative result is a true negative result or whether it is due to the incompleteness of the results of the interpolation method used. In this way, using an incomplete interpolation method undermines empirical evaluation, and analysis. Because of this constraint on completeness, all points of evaluation and analysis in this thesis are restricted to input ontologies with constructs that are no more expressive than $\mathcal{ALC}$. It is worth noting that while the theoretical specification of LETHE is a complete interpolation method for $\mathcal{ALC}$ (as opposed to FAME), in practice its software implementation may contain bugs or other overlooked issues that make the implementation incomplete.

4.2 Approach to *Conveying* Symbols

Let O be an ontology and Σ a subset signature of O such that $\Sigma \subseteq sig(O)$. Throughout this thesis, we are interested in the specific problem of *conveying* symbols that belong to $sig(O) \setminus \Sigma$ using only symbols in Σ. The term *convey* or *conveying* may change depending on the type of symbol being conveyed. Because we restrict ourselves to the description logic $\mathcal{ALC}$, we are only interested in conveying concept names and roles.

Let A be a concept name not in Σ. If A has a definition C such that $O \models A \equiv C$ and $sig(C) \subseteq \Sigma$, then C is an exact representation of A in Σ. However, if A has no such definition C, we are interested in extracting a superconcept C and subconcept D (depending on the scenario) such that $sig(C) \subseteq \Sigma$, $sig(D) \subseteq \Sigma$, $O \models A \sqsubseteq C$ and $O \models D \sqsubseteq A$. Furthermore, we would like both for C and D to be as close to A as possible (in terms of subsumption) to capture the maximally precise subsumer or subsumee of A with respect to Σ.

Let r be a role not in Σ. We are interested in conveying the closest approximation of the domain C or range D of r such that $sig(C) \subseteq \Sigma$, $sig(D) \subseteq \Sigma$, $O \models \exists r.\top \sqsubseteq C$ and $O \models \top \sqsubseteq \forall r.D$.

Thus when we say "conveying a concept C" what we mean is that we convey a definition, a subsumer, or a subsumee of C. Similarly, when we say "conveying a role r" what this means is that we are either conveying an approximation of a domain of r that does not contain r in its signature or similarly, an approximation of a range of r that does not contain r in its signature.

Chapter 5

Realising Knowledge Extraction for DL

This chapter is organised as follows: in Section 5.1 we realise and adopt the notions of strongest necessary conditions and weakest sufficient conditions to description logics. In Section 5.2 we discuss all the novel logic-based knowledge extraction techniques developed in the thesis, including *compiled superconcepts*, *compiled subconcepts*, *domain extraction*, and *range extraction*.

5.1 Defining Strongest Necessary and Weakest Sufficient Conditions in DL

5.1.1 Motivation

The knowledge extraction techniques discussed in Chapter 2 include *uniform interpolation*, *module extraction*, *strongest necessary conditions*, and *weakest sufficient conditions*. However, only uniform interpolation and module extraction have existing practical implementations in description logics. The communication scenario of interest in this thesis is conveying concepts using only symbols from a restricted signature (that reflects the common signature of an agent with a communication partner), thus it may be tempting to assume that methods such as uniform interpolation or module extraction will suffice. Uniform interpolation provides the ability to restrict the knowledge in an ontology to a specific signature, thus we may easily fix the input signature to be the signature of the communicating agents. Module extraction provides the ability to extract a subset of an ontology called a module that captures knowledge relating to an

input signature (in a fashion that ensures the original ontology is a conservative extension), thus to extract knowledge about a specific concept from an ontology, we may simply extract a module for the signature of the concept. However, given a restricted signature Σ, the existing implementations for extracting modules are not guaranteed to return modules whose signatures are within Σ. Given an ontology O, a concept A and a restricted signature Σ such that $A \in sig(O)$, $A \notin \Sigma$ and $\Sigma \subset sig(O)$, a simple algorithm to extract a description of A using only symbols in Σ is simply to extract a module $\mathcal{M}$ for input signature $\{A\}$ and restrict $\mathcal{M}$ to Σ using uniform interpolation (with the goal of the output being a concept that describes A). We demonstrate (through the use of examples), that this simple approach does not suffice.

Example 5.1.1. Take O to be:

$$\alpha_1 : A_1 \sqcap A_2 \sqsubseteq B_2 \sqcup B_3 \qquad \alpha_2 : A_3 \sqsubseteq B_3 \sqcup \neg A_1$$

Let $\Sigma = sig(O) \setminus \{A_1\}$.
Suppose an agent is attempting to convey a description of A_1 with respect to Σ using the method described above, i.e., by extracting the bottom module for A_1 (in order to filter axioms in O that are *generally* relevant to A_1) and then restrict the extracted module to Σ using uniform interpolation. The bottom module extracted for A_1 with respect to O is O. The uniform interpolant for O with respect to $sig(O) \setminus \{A_1\}$ is the empty set ($\emptyset$) which does not contain any information about A_1.
However observe that $O \models A_1 \sqsubseteq \neg A_3 \sqcup B_3$ and $B_3, A_3 \in \Sigma$. This means that there exists a description (specifically a subsumer) for A_1 that can be expressed only using symbols in Σ; however, this description can not be extracted with the sole use of uniform interpolation or module extraction.

Example 5.1.1 illustrates why uniform interpolation and module extraction alone are not sufficient for extracting descriptions of concepts. A possible solution to the approach used in Example 5.1.1 to ensure that the concept being conveyed is always within the signature of interest or added to the signature of interest (with the goal of the output being a set of axioms that describe A): observe that for the ontology O from Example 5.1.1, a uniform interpolant of O with respect to $\Sigma \cup A_1$ is O.

Even if the concept intended to be conveyed is within the signature of interest, another issue that arises with only using uniform interpolation and module extraction is that sometimes the information preserved does not explicitly state information about the concept to be conveyed.

Example 5.1.2. Take O to be:

$$\alpha_1 : A_1 \sqcap A_2 \sqsubseteq B_2 \sqcap B_3 \quad \alpha_2 : B_2 \sqsubseteq A_1 \quad \alpha_3 : A_1 \sqsubseteq C_1 \quad \alpha_4 : C_1 \sqsubseteq C_3$$
$$\alpha_5 : C_3 \sqsubseteq A_2 \qquad \alpha_6 : C_2 \sqsubseteq C_4$$

Let $\Sigma = sig(O) \setminus \{C_3\}$.
Suppose an agent is attempting to convey a description of A_1 with respect to Σ by extracting the bottom module for A_1 of interest and then restrict the module to Σ using uniform interpolation. The bottom module extracted for A_1 with respect to O is O. The uniform interpolant for O with respect to Σ is:

$$\alpha_1 : A_1 \sqcap A_2 \sqsubseteq B_2 \sqcap B_3 \quad \alpha_2 : B_2 \sqsubseteq A_1 \quad \alpha_3 : A_1 \sqsubseteq C_1 \quad \alpha_4 : C_1 \sqsubseteq A_2$$
$$\alpha_5 : C_2 \sqsubseteq C_4$$

On observing the axiom $A_1 \sqsubseteq C_1$, the agent may naively convey C_1 as an approximation of A_1; however, B_2 is a more accurate representation of A_1 with respect to Σ, as $O \models A_1 \equiv B_2$.

Example 5.1.2 also illustrates why uniform interpolation and module extraction alone are not sufficient: depending on the common vocabulary, a uniform interpolant may contain axioms unrelated to the concept an agent wants to share information about, we consider such axioms irrelevant. For example, the axiom α_5 in the uniform interpolant from Example 5.1.2, has nothing to do with the concept A_1; filtering out relevant information about A_1 in the midst of redundant axioms is critical to maximally convey a precise description of A_1, while uniform interpolation and module extraction may provide a basis for restricting axioms in an ontology to a signature, SNCs and WSCs are defined to be more suitable for extracting a superconcept and subconcept of a concept, which form a more precise description as opposed to a uniform interpolant or a module.

5.1.2 Realisation

SNCs and WSCs have been well defined in propositional logic, first order logic and second order logic. Our formal adaptations of strongest necessary and weakest sufficient conditions in the context of description logic is specified in Definition 32 and Definition 33.

Definition 32 ((Strongest) Necessary Conditions)**.** *Let $\mathcal{L}$ be a description logic, Σ be a set of concept and role names, X a concept in $\mathcal{L}$, and O an ontology in $\mathcal{L}$ such*

that $\Sigma \subseteq sig(O)$*. We define a* necessary condition *of* X *over* Σ *relative to* O *to be any concept* C *in* $\mathcal{L}$ *such that* $sig(C) \subseteq \Sigma$ *and* $O \models X \sqsubseteq C$*.* C *is a* strongest necessary condition*, denoted* $SNC(X;O;\Sigma)$*, if for any other necessary condition* C' *in* $\mathcal{L}$ *of* X *over* Σ *relative to* O *we have that* $O \models C \sqsubseteq C'$*.* C *is a* minimal *strongest necessary condition, denoted* $SNC_M(X;O;\Sigma)$*, if for any strongest necessary condition* C' *of* X *over* Σ *relative to* O *we have that* $sig(C) \subseteq sig(C')$*.*

Definition 33 ((Weakest) Sufficient Conditions)**.** *Let* $\mathcal{L}$ *be a description logic* Σ *be a set of concept and role names,* X *a concept in* $\mathcal{L}$*, and* O *an ontology in* $\mathcal{L}$*, such that* $\Sigma \subseteq sig(O)$*. We define a* sufficient condition *of* X *over* Σ *relative to* O *to be any concept* C *in* $\mathcal{L}$ *such that* $sig(C) \subseteq \Sigma$ *and* $O \models C \sqsubseteq X$*.* C *is a* weakest sufficient condition*, denoted* $WSC(X;O;\Sigma)$*, if for any other sufficient condition* C' *in* $\mathcal{L}$ *of* X *over* Σ *relative to* O *we have that* $O \models C' \sqsubseteq C$*.* C *is a* minimal *weakest sufficient condition, denoted* $WSC_M(X;O;\Sigma)$*, if for any strongest necessary condition* C' *of* X *over* Σ *relative to* O *we have that* $sig(C) \subseteq sig(C')$*.*

Observe that an SNC (WSC) forms a maximally precise subsumer (subsumee) of a concept X with respect to an ontology O and signature Σ for a given description logic $\mathcal{L}$. Thus SNCs (WSCs) may be used to form a maximally precise knowledge structure that describes a concept; which is one of the desirable properties of knowledge structures discussed in Chapter 1. If the description logic $\mathcal{L}$ is clear, we do not mention it explicitly. We lift Lemma 2 by Doherty and Lukaszewicz [DLS01] to construct an algorithm to extract strongest necessary conditions. Lemma 2 suggests that for any first order formula X, set of relation symbols Σ, and closed theory O, an SNC is a uniform interpolant for $O \wedge X$ over Σ. Because description logics are decidable fragments of first-order logic, one may assume that in DL we can extract the SNC for a concept name X over a signature Σ, and ontology O by extracting the uniform interpolant for $O \wedge X$ over Σ; however, $O \wedge X$ can not be simply represented in description logic (at least not without grounding X with an individual constant). To illustrate why we can not extract an SNC for a concept name X over a signature Σ, and ontology O by extracting the uniform interpolant for O over Σ (following Lemma 2), take O to be the following ontology:

$$\alpha_1 : \mathrm{A} \sqsubseteq \mathrm{B} \qquad \alpha_2 : \mathrm{W} \sqsubseteq \mathrm{A} \qquad \alpha_3 : \mathrm{C} \sqsubseteq \mathrm{D} \qquad \alpha_4 : \mathrm{E} \sqsubseteq \mathrm{F}$$

$$\alpha_5 : \mathrm{F} \sqsubseteq \mathrm{G} \qquad \alpha_6 : \mathrm{B} \sqsubseteq \neg\, \mathrm{D} \qquad \alpha_7 : \mathrm{B} \sqsubseteq \neg\, \mathrm{G} \qquad \alpha_8 : \mathrm{G} \sqsubseteq \neg\, \mathrm{D}$$

Let $\Sigma = \{A, D, E, G, B\}$, to extract the SNC for W with respect to O and Σ we extract a uniform interpolant over O, W which is $\{\alpha_1, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8\}$. As can been

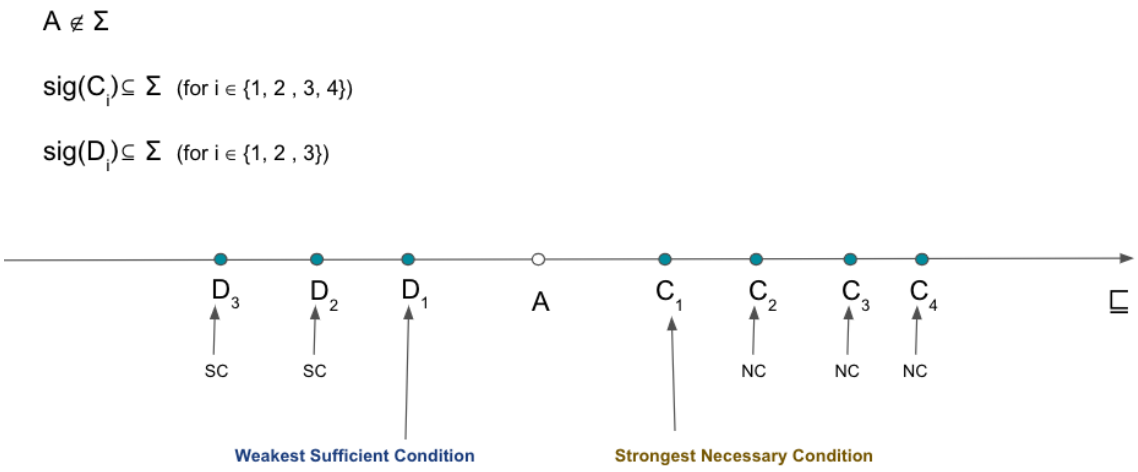

Figure 5.1: Illustration of (Strongest) Necessary Conditions and (Weakest) Sufficient Conditions. 'SC' stands for 'sufficient condition'. 'NC' stands for 'necessary condition'. The horizontal axis has a subsumption ordering such that the concept A appears to the left of a concept B iff $O \models A \sqsubseteq B$.

seen, there are axioms in the interpolant not related to W, the TBox axiom $E \sqsubseteq G$, for example, is not related to W; we call all such axioms irrelevant.

If we are to use uniform interpolation to extract SNCs and WSCs, we must find a way to filter concepts relevant to the concept of interest. We achieve this by grounding the concept of interest using a fresh individual. That is, to leverage Lemma 2 to extract a strongest necessary condition of a concept X with respect to a signature Σ under an ontology O, we ground X with some fresh individual a before extracting a uniform interpolant of O with respect to Σ. The idea is that concepts in the resulting uniform interpolant with a in their signatures are necessary conditions of X under O expressed only using symbols in Σ; this is proven in Proposition 4 below. Let O be an ontology, Σ a subset signature of O, a a fresh individual such that $a \notin O$ and X a concept name for which we wish to extract an SNC for, with respect to O and Σ. We extract a uniform interpolant $\mathcal{V}$ over Σ for $O \cup \{X(a)\}$ and solve the issue of redundancies in the result by returning a conjunction of concepts of the form $C(a) \in \mathcal{V}$: it follows from Proposition 4 that all such concepts C are necessary conditions of X, and thus their conjunction will satisfy Definition 32. This approach is outlined in more detail in Algorithm 5.1.1.

Proposition 4. *Let O be an $\mathcal{ALC}$ ontology, X a concept in $\mathcal{ALC}$, $\Sigma \subseteq sig(O)$, and a a fresh individual such that $a \notin sig(O)$ and $sig(X) \not\subseteq \Sigma$. Suppose $O \not\models X(a)$. Let $\mathcal{V}$ be a uniform interpolant for $\Sigma \cup \{a\}$ relative to $O \cup \{X(a)\}$, we have that $\mathcal{V} \models C(a)$ iff $O \models X \sqsubseteq C$ where C is an arbitrary $\mathcal{ALC}$-concept such that $sig(C) \subseteq \Sigma$.*

Proof. If $\mathcal{V} \models C(a)$, then $O \models X \sqsubseteq C$: Since $\mathcal{V}$ is a uniform interpolant for $\Sigma \cup \{a\}$ of $O \cup \{X(a)\}$, $O \cup \{X(a)\} \models \mathcal{V}$. Given that $\mathcal{V} \models C(a)$, we have that $O \cup \{X(a)\} \models C(a)$. This implies $O \cup \{X(a)\}, \neg C(a)$ is unsatisfiable. Hence $O \models X \sqsubseteq C$ since $O \models X \sqsubseteq C$ iff $O \cup X(a'), \neg C(a') \models \bot$ for some fresh individual a' [BCM+03].

If $O \models X \sqsubseteq C$, then $\mathcal{V} \models C(a)$: Because $O \models X \sqsubseteq C$, for every interpretation I such that $I \models O$, we have that: $x \in X^I$ implies $x \in C^I$ for any individual $x \in \Delta^I$. Therefore, given a fresh individual a not occuring in $sig(O)$, $O \cup \{X(a)\} \models C(a)$. Since $\mathcal{V}$ is a uniform interpolant for $\Sigma \cup \{a\}$ of $O \cup \{X(a)\}$, it follows from Definition 18 that $\mathcal{V} \models C(a)$. □

A consequence of Proposition 4 is that axioms in $\mathcal{V}$ containing a in their signature are necessary conditions of X; thus, because $\mathcal{V}$ is a uniform interpolant of O for Σ, $\bigsqcap_{C(a) \in \mathcal{V}} C$ is a strongest necessary of X. If $\bigsqcap_{C(a) \in \mathcal{V}} C$ is a not a strongest necessary of X, then $\mathcal{V}$ is not a uniform interpolant of O for Σ because there will exist some concept C' such that $sig(C') \subseteq \Sigma$, $O \models C'(a)$, and $\mathcal{V} \not\models C'(a)$.

ALGORITHM 5.1.1

Input: An ontology O, a concept X, a signature Σ, an individual a where $\Sigma \subseteq sig(O)$, $X \in sig(O)$, $X \notin \Sigma$, $a \notin sig(O)$.

Output: An $\mathcal{ALC}$ concept C which is an SNC of X.

Step 1: Add $X(a)$ to O to get O'.

Step 2: Add a to Σ to get Σ' and extract a uniform interpolant denoted $\mathcal{V}$ over O' with respect to Σ'.

Step 3: Return $\bigsqcap_{C(a) \in \mathcal{V}} C$ as $SNC(X; O; \Sigma)$.

Weakest sufficient conditions are a dual of strongest necessary conditions: let O be an ontology, X be a concept name. For any concept C such that $O \models C \sqsubseteq X$, we have that $O \models \neg X \sqsubseteq \neg C$, hence, if we extract a strongest necessary for $\neg X$ the result obtained will be a negation of a weakest sufficient condition for X, thus in order to extract a weakest sufficient conditions for a concept X, we extract a strongest necessary condition of $\neg X$ and negate the result.

ALGORITHM 5.1.2

Input: An ontology O, a concept X, a signature Σ, an individual a where $\Sigma \subseteq sig(O)$, $X \in sig(O)$, $X \notin \Sigma$, $a \notin sig(O)$.

Output: An $\mathcal{ALC}$ concept C which is a WSC of X.

Step 1: Add $\neg X(a)$ to O to get O'.

Step 2: Add a to Σ to get Σ' and extract a uniform interpolant denoted $\mathcal{V}$ over O' with respect to Σ'.

Step 3: Return $\neg \bigsqcap_{C(a) \in \mathcal{V}} C$ as $WSC(X;O;\Sigma)$.

When extracting information about a defined concept, it is desirable to capture its definition if possible as the definition is effectively a synonym for the concept.

Lemma 5. *Let O be an ontology, Σ a subset of $sig(O)$ and X a concept such that $sig(X) \subseteq sig(O)$. Then $O \models WSC(X;O;\Sigma) \sqsubseteq SNC(X;O;\Sigma)$.*

Proof. This follows trivially from Definitions 32 and 33 because:

1. $O \models WSC(X;O;\Sigma) \sqsubseteq X$
2. $O \models X \sqsubseteq SNC(X;O;\Sigma)$

□

Theorem 6. *Let O be an ontology, Σ a subset of $sig(O)$ and C a concept such that $sig(C) \not\subseteq \Sigma$, a concept D with $O \models C \equiv D$, and $sig(D) \subseteq \Sigma$. Then (i) $O \models C \equiv SNC(C;O;\Sigma)$, and therefore, (ii) $SNC(C;O;\Sigma)$ is an explicit definition for C under O.*

Proof. Let $O \models C \equiv D$. Then it follows that $O \models C \sqsubseteq D$. Since $sig(D) \subseteq \Sigma$, it follows from Definition 32 that $O \models SNC(C;O;\Sigma) \sqsubseteq D$. It also follows that $O \models D \sqsubseteq C$, and since $O \models SNC(C;O;\Sigma) \sqsubseteq D$, it follows that $O \models SNC(C;O;\Sigma) \sqsubseteq C$. Thus, $O \models SNC(C;O;\Sigma) \sqsubseteq C, C \sqsubseteq SNC(C;O;\Sigma)$ which means that $O \models C \equiv SNC(C;O;\Sigma)$ and (i) holds. If there exists a concept D such that $O \models C \equiv D$ and $sig(D) \subseteq \Sigma$ then C is explicitly defined, thus if (i) holds, then (ii) holds. □

It follows from Theorem 6 that strongest necessary conditions can be used as a means to extract definitions for concepts that are explicitly defined with respect to a restricted signature. In the same way Theorem 7 demonstrates that weakest sufficient conditions may also be used as a means to extract definitions.

Theorem 7. *Let O be an ontology, Σ a subset of $sig(O)$, C a concept such that $sig(C) \not\subseteq \Sigma$, If there exists a concept D such that $O \models C \equiv D$, and $sig(D) \subseteq \Sigma$, then (i) $O \models C \equiv WSC(C;O;\Sigma)$, and therefore, (ii) $WSC(C;O;\Sigma)$ is an explicit definition for C under O.*

The proof of Theorem 7 is analogous to the proof of Theorem 6

Strongest necessary conditions and weakest sufficient conditions may be useful in extracting descriptions of concepts but still maintain logical properties of definitions in the instance that the concept of interest is defined. Using the implicit definability test discussed in Section 2.2.1, we may determine if a concept is implicitly definable with respect to a restricted signature of an ontology, thus given that the Beth definability property holds for the description logic $\mathcal{ALC}$, we can conclude that if the implicit definability test passes for a concept, then the concept has an explicit definition.

Example 5.1.3. Take O to be:

$$\alpha_1 : \mathrm{A} \equiv \mathrm{B} \sqcup \mathrm{C} \qquad \alpha_2 : \mathrm{B} \sqsubseteq \neg\, \mathrm{C}$$

C is explicitly definable under O with respect to $sig(O) \setminus \{C\}$. A strongest necessary condition for C with respect to $\{A, B\}$ extracted using Algorithm 5.1.1, is $A \sqcap \neg B$ which is equivalent to C under O.

5.1.3 Complexity

Algorithm 5.1.1 and Algorithm 5.1.2 call the forgetting algorithm at most once, and thus bounded by the computational complexity of the forgetting algorithm used. Extracting a uniform interpolant using LETHE has been shown to have a 2EXPTIME worst-case space complexity [WDP19, PK15c] in the number of symbols contained in the ontology; thus our implementations of Algorithm 5.1.1 and Algorithm 5.1.2 have a 2EXPTIME worst-case space complexity.

In Chapter 6 we empirically evaluate the performance Algorithm 5.1.1 in extracting definitions from ontologies.

5.1.4 Relation to Prime Implicates and Prime Implicants

Prime implicates and prime implicants specify a notion of *primeness* among concepts with respect to an empty ontology. This notion of primeness *loosely* corresponds with the property of being a *maximally precise* subsumer of a given concept of interest *with respect to the empty ontology*. However, for our application of interest (i.e., an agent

being able to describe a concept in its ontology with respect to a restricted vocabulary), the ontology of the agent and the restricted vocabulary need to be taken into account. Prime implicates and implicants do not factor-in a restricted vocabulary. We may extend the description of prime implicates (see Section 2.2.6) to fit our application needs as follows:

Let O be an ontology and Σ be a signature such that $\Sigma \subseteq sig(O)$. A clause Cl (as specified in Section 2.2.6) is said to be a *prime implicate* of a class expression C iff:

1. $sig(Cl) \subseteq \Sigma$
2. $O \models C \sqsubseteq Cl$
3. For any Cl' such that $sig(Cl') \subseteq \Sigma$, if $O \models Cl' \sqsubseteq Cl$ then $O \models Cl \sqsubseteq Cl'$.

This description achieves maximal precision for *clausal* concepts; however, for our application of communicating agents, it does not matter if the concept to be conveyed is clausal or not, i.e., we are interested in free-form concepts. Dropping the clausal requirement of the above-extended description of prime implicates results in a definition analogous to the definition of strongest necessary conditions. The dual case holds for prime implicants and weakest sufficient conditions. Thus, a notion of prime implicates and prime implicants adjusted to fit our application of communicating agents will yield notions analogous to strongest necessary and weakest sufficient conditions.

5.1.5 Relation to Least Common Subsumers

Since by definition an LCS is a subsumer, an algorithm for extracting LCSs may be used to extract a subsumer for a concept C by providing a singleton set $\{C\}$ as input to the algorithm. Restricting the definition of LCS (Definition 28) to a single concept yields the following description:

Let $\mathcal{L}$ be a description logic and O be an $\mathcal{L}$-ontology. An $\mathcal{L}$-concept C is the *least common subsumer* of an $\mathcal{L}$-concept C_0 iff:

1. $O \models C_0 \sqsubseteq C$, and
2. For any other concept C' that satisfies (1), we have that $O \models C \sqsubseteq C'$

Observe that this description defines the LCS of a concept C to be its maximal subsumer with respect to O; however, in all cases, the LCS of a concept C will always

be C. Furthermore, this description does not take into account a restricted signature Σ which is important for our intended application of communicated agents. Adding such a restriction will result in the following description:

Let $\mathcal{L}$ be a description logic, O be an $\mathcal{L}$-ontology, and Σ be a subset of $sig(O)$. An $\mathcal{L}$-concept C is the *least common subsumer* of an $\mathcal{L}$-concept C_0 iff:

1. $O \models C_0 \sqsubseteq C$,
2. $sig(C) \subseteq \Sigma$, and
3. For any other concept C' that satisfies (1) and (2), we have that $O \models C \sqsubseteq C'$

Observe that with the added signature restriction constraint, we obtain a description analogous to strongest necessary conditions.

5.1.6 Limitations

Without restrictions on the destination description logic, strongest necessary conditions do not always exist within certain description logics without allowing for additional constructs.

It would seem strongest necessary conditions do not always exist in the description logic $\mathcal{ALC}$ or any extension of $\mathcal{ALC}$ that do not allow for role inverses for input $\mathcal{ALC}$ ontologies and concepts. For example, if we fix the description logic to be $\mathcal{ALC}$, and take O to be:

1. $A_1 \sqsubseteq \forall r.\neg A_2$

Fix Σ to be $\{r, A_1\}$. A strongest necessary condition of A_2 for Σ in $\mathcal{ALCI}$ under O is $\forall r^-.\neg A_1$, and thus would seem inexpressible in $\mathcal{ALC}$. However, there is some work by De Giacomo [Gia96] suggesting that $\mathcal{ALCI}$ ontologies can be rewritten to $\mathcal{ALC}$ ontologies which suggests that $\forall r^-.\neg A_1$ is expressible without the use of role inverses.

Theorem 8. *Strongest necessary conditions do not always exist in the description logic $\mathcal{EL}$ for input $\mathcal{EL}$ ontologies and concepts.*

Proof. The proof is by example. Fix the description logic to be $\mathcal{EL}$, and take O to be:

1. $A_1 \sqsubseteq B$

2. $A_1 \sqcap A_2 \sqsubseteq \bot$

Fix Σ to be $\{B, A_2, C\}$. A strongest necessary condition of A for Σ in $\mathcal{ALC}$ under O is $B \sqcap \neg A_2)$; however, this is not expressible in $\mathcal{EL}$. □

Theorem 9. *Strongest necessary conditions do not always exist in the description logic $\mathcal{ALC}$ or any extensions of $\mathcal{ALC}$ that do not allow for unqualified number restrictions or similar constructors for input $\mathcal{ALC}$ ontologies and concepts.*

Proof. The proof is by example. Fix the description logic to be $\mathcal{ALC}$, and take O to be:

1. $A \sqsubseteq \exists r.\neg B$

2. $A \sqsubseteq \exists r.B$

Fix Σ to be $\{r\}$. A strongest necessary condition of A over Σ is not expressible in $\mathcal{ALC}$.

Remark: It is worth noting that in a more expressive DL such as $\mathcal{ALCN}$ a strongest necessary condition of A over Σ under O can be expressed as $\geq 2r.\top$. □

Theorem 10. *Strongest necessary conditions do not always exist in the description logic $\mathcal{ALC}$ or any extensions of $\mathcal{ALC}$ that do not allow for qualified number restrictions or similar constructors for input $\mathcal{ALC}$ ontologies and concepts.*

Proof. The proof is by example. Fix the description logic to be $\mathcal{ALC}$, and take O to be:

1. $A \sqsubseteq \exists r.B_1$

2. $A \sqsubseteq \exists r.B_2$

3. $B_1 \sqcap B_2 \sqsubseteq \bot$

4. $B_1 \sqsubseteq C$

5. $B_2 \sqsubseteq C$

Fix Σ to be $\{r, C\}$. A strongest necessary condition of A for Σ in $\mathcal{ALCQ}$ under O is $\geq 2r.C$; however, this is not expressible without the use of qualified number restrictions. □

A limitation of Algorithm 5.1.1 and Algorithm 5.1.2 is that the size of their results are completely dependent on the uniform interpolation algorithm used. As an example, take LETHE to be the uniform interpolation algorithm used to implement Algorithm 5.1.1 and Algorithm 5.1.2. Because LETHE is saturation based, this means that it is possible for superfluous information to be included in the result, thus ensuring that it is very unlikely that Algorithm 5.1.1 and Algorithm 5.1.2 can be used to extract minimal strongest necessary or weakest sufficient conditions. Let O be $\{A \equiv B, A \sqsubseteq C\}$. A strongest necessary of A with respect to $sig(O) \setminus \{A\}$ is B. However, the result returned by an implementation of Algorithm 5.1.1 that is based on an implementation of LETHE returns $B \sqcap C$ (observe that $O \models B \sqcap C \equiv B$).

Up to this point we have discussed methods for extracting descriptions of concepts in an ontology using SNCs and WSCs. In the next section, we discuss methods developed to extract descriptions of concepts that SNCs and WSCs are not guaranteed to capture. Furthermore, we discuss methods developed to extract descriptions of roles.

5.2 Developed Approaches

5.2.1 Compiled Superconcepts and Compiled Subconcepts

A caveat that is associated with SNCs and WSCs is that their desirable logical properties only hold with respect to the ontology of interest. Here by *desirable logical properties associated with SNCs and WSCs*, we mean the fact that a strongest necessary condition (or weakest sufficient condition) of a concept X with respect to a signature Σ under an ontology O, is the *strongest* (*weakest*) (in terms of subsumption as specified in Definition 32 and Definition 33) subsumer (subsumee) of X with respect to O. Intuitively this means a strongest necessary condition (weakest sufficient condition) of X is the closest upper (lower) approximation of X *given* the background ontology O; outside the context of the background ontology, the SNC (WSC) of X is not necessarily the closest approximation of X. In the context of communicating agents, the guarantees provided by the logical properties SNCs and WSCs do not necessarily hold because the agents have different ontologies. Given an agent AG_1 with ontology O_1 such that AG_1 has a common signature Σ with a communication partner AG_2, $SNC(X; O_1; \Sigma)$ for an uncommon concept name X in AG_1's ontology is not necessarily specific or informative enough for AG_2 to understand the meaning of X.

Example 5.2.1. Consider agent AG_1 with ontology O_1:

$$\alpha_1 : \mathrm{A} \sqsubseteq \mathrm{B} \qquad \alpha_2 : \mathrm{B} \sqsubseteq \exists \mathrm{r}.(\mathrm{C} \sqcap \mathrm{D}) \qquad \alpha_3 : \mathrm{D} \sqsubseteq \exists \mathrm{r}.\mathrm{E}$$

And agent AG_2 with ontology O_2:

$$\beta_1 : \mathrm{B} \sqsubseteq \mathrm{F} \qquad \beta_2 : \mathrm{F} \sqsubseteq \forall \mathrm{r}.\mathrm{P} \qquad \beta_3 : \mathrm{G} \equiv \exists \mathrm{r}.(\mathrm{C} \sqcap \mathrm{D})$$

The common signature is $\Sigma = \{B, r, D\}$. A strongest necessary condition for A with respect to Σ under O_1 is B. If A_1 conveys B in place of A, this could lead to communication errors as the interpretation of B in O_1 is completely different from the interpretation of B in O_2: $O_1 \models B \sqsubseteq \exists r.D$, and $O_2 \models B \sqsubseteq \forall r.P$.

As another example consider agent AG_1 with ontology $O_1 = \{A_1 \sqsubseteq A_4, A_3 \equiv A_1, A_4 \sqsubseteq \exists r.C_1, A_1 \sqsubseteq B_1\}$ and AG_2 with ontology $O_2 = \{A_2 \sqsubseteq B_1, A_2 \sqsubseteq \forall r.C_1, A_5 \equiv \exists r.C_1\}$, the common signature is $\{C_1, B_1, r\}$, if AG_1 wishes to requested for instances of A_3 from AG_2, it may extract a strongest necessary condition for A_3 as B_1 and thus convey B_1 in place of A_3, and agent AG_2 may then proceed to return instances of B_1 to AG_1; however, $O_1 \models A_3 \sqsubseteq \exists r.C_1$ and $O_2 \models A_5 \equiv \exists r.C_1$, thus there may be instances of A_5 or $\exists r.C_1$ that may also be instances of A_3 that will be excluded from the set of instances returned by agent AG_2. An ideal subsumer for A_3 with respect to the common signature is $\exists r.C_1 \sqcap B_1$, which when conveyed to agent AG_2 will ensure that agent AG_2 also contains instances of A_5 and $\exists r.C_1$ as possible instances of A_3.

The only instance in which the desirable properties of SNCs and WSCs hold is if the common signature has the same interpretation in the ontologies of the respective communicating agents. Thus a desirable prerequisite for agents attempting to communicate using SNCs/WSCs is for all symbols in their common signature to have the same interpretation in their respective ontologies. However, such a prerequisite may be too strong and unattainable in some cases. In cases where such a requirement is too strong or unattainable, a desirable ability for an agent to have is to extract everything its ontology says about a concept. We formalise this ability as the ability to extract *Compiled Superconcepts* (Definition 34) and *Compiled Subconcepts* (Definition 35).

Definition 34 (Compiled Superconcept)**.** *Let $\mathcal{L}$ be a description logic, O be an $\mathcal{L}$-ontology, A an $\mathcal{L}$-concept name and C an $\mathcal{L}$ concept. We call C an $\mathcal{L}$* compiled superconcept *of A if $O \models A \sqsubseteq C$ and for any other $\mathcal{L}$-concept C' such that $O \models A \sqsubseteq C'$, we have that $\{A \sqsubseteq C\} \models A \sqsubseteq C'$. We denote the compiled superconcept $CSP(A;O)$ (CSP for short) and $CSP(A;O;\Sigma)$ if it is restricted to a signature Σ (i.e., $sig(CSP(A;O;\Sigma)) \subseteq \Sigma$).*

Definition 35 (Compiled Subconcept)**.** *Let $\mathcal{L}$ be a description logic, O be an $\mathcal{L}$-ontology, A an $\mathcal{L}$-concept name and C an $\mathcal{L}$ concept. We call C an $\mathcal{L}$* compiled subconcept *of A if $O \models C \sqsubseteq A$ and for any other $\mathcal{L}$-concept C' such that $O \models C' \sqsubseteq A$, we have that $\{C \sqsubseteq A\} \models C' \sqsubseteq A$. We denote the compiled subconcept $CSB(A;O)$ (CSB for short) and $CSB(A;O;\Sigma)$ if it is restricted to a signature Σ (i.e., $sig(CSB(A;O;\Sigma)) \subseteq \Sigma$).*

Remark 1. *In Definition 34 we use the condition $\{A \sqsubseteq C\} \models A \sqsubseteq C'$. At a first glance, it may be tempting to assume that this is unnecessary as it seems that $\{A \sqsubseteq C\} \models A \sqsubseteq C'$ is equivalent to $\emptyset \models C \sqsubseteq C'$. For example, the compiled superconcept of A w.r.t $O = \{A \sqsubseteq B, B \sqsubseteq C\}$ is $A \sqcap B \sqcap C$, and indeed, for every concept D such that $O \models A \sqsubseteq D$, we have that $\emptyset \models A \sqcap B \sqcap C \sqsubseteq D$. However, it is worth noting that if A exists in an interesting cycle (see Section 2.2.1), $\emptyset \models C \sqsubseteq C'$ does not capture the essence of a compiled superconcept. As an example consider the ontology $O = \{A \sqsubseteq \exists r.A, B \sqsubseteq C\}$; A compiled superconcept of A w.r.t to O is $\exists r.A$, but we have that $O \models A \sqsubseteq \exists r.\exists r.A$ and $\emptyset \not\models \exists r.A \sqsubseteq \exists r.\exists r.A$. However, observe that $\{A \sqsubseteq \exists r.A\} \models A \sqsubseteq \exists r.\exists r.A$.*

The dual case holds for compiled subconcepts.

Property 1 (CSPs are Strongest Necessary Conditions)**.** *If C is an $\mathcal{ALC}$ compiled superconcept of a concept name A with respect to an $\mathcal{ALC}$ ontology O retricted to a signature Σ, then C is also a strongest necessary condition of A with respect to O .*

Proof. It follows from Definition 34 that:

1. $O \models A \sqsubseteq C$.
2. $sig(C) \subseteq \Sigma$.

As such, for any other $\mathcal{ALC}$-concept C' such that $O \models A \sqsubseteq C'$ and $sig(C') \subseteq \Sigma$, we have that $\{A \sqsubseteq C\} \models A \sqsubseteq C'$. Since $O \models A \sqsubseteq C$, we have that $O \equiv O \cup \{A \sqsubseteq C\}$, and thus it also follows that for any $\mathcal{ALC}$-concept C' such that $O \models A \sqsubseteq C'$, we have that $O \models C \sqsubseteq C'$, and thus C is a strongest necessary condition of A with respect to O . □

Property 2 (CSBs are Weakest Sufficient Conditions)**.** *If C is an $\mathcal{ALC}$ compiled subconcept of a concept name A with respect to an $\mathcal{ALC}$ ontology O retricted to a signature Σ, then C is also a weakest sufficient condition of A with respect to O.*

Proof. It follows from Definition 35 that:

1. $O \models C \sqsubseteq A$.

2. $sig(C) \subseteq \Sigma$.

As such, for any other $\mathcal{ALC}$-concept C' such that $O \models C' \sqsubseteq A$ and $sig(C') \subseteq \Sigma$, we have that $\{C \sqsubseteq A\} \models C' \sqsubseteq A$. Since $O \models C \sqsubseteq A$, we have that $O \equiv O \cup \{C \sqsubseteq A\}$, and thus it also follows that for any $\mathcal{ALC}$-concept C' such that $O \models C' \sqsubseteq A$, we have that $O \models C' \sqsubseteq C$, and thus C is a weakest sufficient condition of A with respect to O . □

Property 3 (Equisatisfiability of CSPs)**.** *If both C and C' are compiled superconcepts of a concept X with respect to an ontology O, then $O \models C \equiv C'$.*

Proof. Following Definition 34, $O \models C \sqsubseteq C'$ and similarly $O \models C' \sqsubseteq C$. □

Property 4 (Equisatisfiability of CSBs)**.** *If both C and C' are compiled subconcepts of a concept X with respect to an ontology O, then $O \models C \equiv C'$.*

Proof. Following Definition 34, $O \models C \sqsubseteq C'$ and similarly $O \models C' \sqsubseteq C$. □

From Definition 34 and Definition 35, we can observe that compiled subconcepts are the dual of compiled superconcepts. Similar to the dual nature between strongest necessary conditions and weakest sufficient conditions, we have that an algorithm for extracting compiled superconcepts may be easily modified into an algorithm for extracting compiled subconcepts. Also, it follows from Property 2, Property 1, and Lemma 5 that for a given ontology O, concept X, and signature Σ such that $\Sigma \subseteq sig(O)$, $O \models CSB(X;O;\Sigma) \sqsubseteq CSP(X;O;\Sigma)$. Let O be an ontology, X be a concept name, for any concept C such that $O \models C \sqsubseteq X$, we have that $O \models \neg X \sqsubseteq \neg C$, hence, if we extract a compiled superconcept for $\neg X$ the result obtained will be a negation of a compiled subconcept for X

Extracting Compiled Superconcepts and Compiled Subconcepts

Our approach to extracting compiled superconcepts is presented in Algorithm 5.2.1 and leverages LETHE's resolution calculus ($RES_{\mathcal{ALC}}$) to attempt to extract a compiled superconcept. Let O be an ontology, X a concept for which we wish to extract a CSP, and a a fresh individual. The intuition here is that we saturate $O \cup \{X(a)\}$ using $RES_{\mathcal{ALC}}$ to extract all non-trivial entailments and then conjoin concept expressions $C(a)$ to form a CSP of X. This is similar to the way LETHE extracts uniform interpolants. To extract a uniform interpolant, LETHE saturates a structurally transformed ontology to derive all possible entailments and then eliminates remnant clauses that include the unwanted signature. By *saturating a set of clauses*, we mean to exhaustively apply LETHE's calculus to a set of clauses to derive a set of inferences.

ALGORITHM 5.2.1

Input:

1. An ontology O consisting only of TBox axioms.
2. A concept name A

Output: A Compiled Superconcept C of A.

Step 1: Add $A(a)$ to O to get O'.

Step 2: Saturate O' using $RES_{\mathcal{ALC}}$ to get O_{RES}.

Step 3: Undo the structural transformation to eliminate all definer symbols in O_{RES}.

Step 4: Return $\bigsqcap_{C(a)\in O_{RES}} C$ as $CSP(A;O)$.

Conjecture 1. Let O be an $\mathcal{ALC}$ TBox and X a concept name occuring in O. Let O' denote the result of saturating $O \cup \{X(a)\}$ using the LETHE resolution calculus $RES_{\mathcal{ALC}}$ where a is a fresh individual not occuring in O. Let C_X denote the conjunction of all concepts C such that $C(a) \in O'$. Then for any concept C' such that $O \models X \sqsubseteq C'$, we have that $\{X \sqsubseteq C_X\} \models X \sqsubseteq C'$.

Recall from Remark 1 that the reason we use the notation $\{X \sqsubseteq C_X\} \models X \sqsubseteq C'$, as opposed to $\emptyset \models C_X \sqsubseteq C'$ is to capture cases where X may be involved in an interesting cycle in O. We show in Lemma 11 that this is not necessary in acyclic terminologies.

Lemma 11. *Let C_X denote the conjunction of all concepts C such that $C(a) \in O'$ and $X \notin sig(C)$. If O is an acyclic terminology, then we have that $\{X \sqsubseteq C_X\} \models X \sqsubseteq C'$ iff $\emptyset \models C_X \sqsubseteq C'$.*

Proof. If $\emptyset \models C_X \sqsubseteq C'$, then $\{X \sqsubseteq C_X\} \models X \sqsubseteq C'$: Assume $\emptyset \models C_X \sqsubseteq C'$ and $\{X \sqsubseteq C_X\} \not\models X \sqsubseteq C'$. Hence, there is an interpretation I such that $I \models \{X \sqsubseteq C_X\}$ and $X^I \not\subseteq C'^I$. Also $C_X^I \subseteq C'^I$ since $\emptyset \models C_X \sqsubseteq C'$. However $X^I \subseteq C_X^I$ and $X^I \not\subseteq C'^I$ contradicts $C_X^I \subseteq C'^I$, thus it must be the case that $\emptyset \models C_X \sqsubseteq C'$ implies $\{X \sqsubseteq C_X\} \models X \sqsubseteq C'$.

If $\{X \sqsubseteq C_X\} \models X \sqsubseteq C'$, then $\emptyset \models C_X \sqsubseteq C'$: Assume $\{X \sqsubseteq C_X\} \models X \sqsubseteq C'$ and $\emptyset \not\models C_X \sqsubseteq C'$. Let I be an interpretation such that $C_X^I \not\subseteq C'^I$. Recall that C_X denotes the conjunction of all concepts C such that $C(a) \in O'$ and $X \notin sig(C)$, thus $X \notin sig(C_X)$, and we can assume without loss of generality that $X \notin sig(C')$. Thus, we can create an extension $\mathcal{J}$ of I with $X^{\mathcal{J}} = C_X^I$. It then follows that $\mathcal{J} \models X \sqsubseteq C_X$ and (since $\{X \sqsubseteq$

$C_X\} \models X \sqsubseteq C'$), $\mathcal{J} \models X \sqsubseteq C'$. However, since $\mathcal{J}$ is an extension of I, and $X^{\mathcal{J}} = C_X^I$, we have that $X^{\mathcal{J}} = C_X^{\mathcal{J}}$, and since $C_X^I \not\subseteq C'^I$, it follows that $C_X^{\mathcal{J}} \not\subseteq C'^{\mathcal{J}}$ which contradicts the assumption that $\{X \sqsubseteq C_X\} \models X \sqsubseteq C'$. Thus, it must be the case that if $\{X \sqsubseteq C_X\} \models X \sqsubseteq C'$, then $\emptyset \models C_X \sqsubseteq C'$. □

Let O be a TBox, and A be a concept name, such that $A \in sig(O)$. Definition 34 roughly states that for a compiled superconcept C of A w.r.t O, for any other concept C' such that $O \models A \sqsubseteq C'$, $\{A \sqsubseteq C\} \models A \sqsubseteq C'$. Similarly, Conjecture 1 states that given an ontology O' which denotes the result of saturating $O \cup \{A(a)\}$ using the LETHE resolution calculus $RES_{\mathcal{ALC}}$ and a concept C_X which denotes $\bigsqcap_{C(a) \in O'} C$, for any other concept C' such that $O \models A \sqsubseteq C'$, we have that $\{A \sqsubseteq C_X\} \models A \sqsubseteq C'$; thus, Conjecture 1 may be used to construct an algorithm for extracting compiled superconcepts, which is what Algorithm 5.2.1 does. Algorithm 5.2.1 leverages Conjecture 1 to extract a compiled superconcept of A w.r.t O by saturating $O \cup A(a)$ using $RES_{\mathcal{ALC}}$. The intuition here is that Algorithm 5.2.1 uses $RES_{\mathcal{ALC}}$ to derive all relevant clauses of the form $C(a)$ from O and since $O \cup \{A(a)\} \models C'(a)$ for every concept C' such that $O \models A \sqsubseteq C'$, Algorithm 5.2.1 returns $\bigsqcap_{C(a) \in O_{RES}} C$; following Conjecture 1 we believe that $\bigsqcap_{C(a) \in O_{RES}} C$ is a compiled superconcept of A, however, this is yet to be proven.

Example 5.2.2. Consider the ontology O consisting of the following axioms:

1. $A \sqsubseteq \forall r.(\neg B \sqcup C)$
2. $A \sqsubseteq \forall r.B$
3. $A(a)$

We have that $O \models A \sqsubseteq \forall r.C$. Since $RES_{\mathcal{ALC}}$ is interpolation complete, it must be able to derive $\forall r.C(a)$ if it is to be used to extract a uniform interpolant of O for the signature $\{r, C\}$. Following [Koo15, PK15a], the structural transformation of O is as follows:

1. $\neg A(x) \sqcup \forall r.D_1(x)$
2. $\neg D_1(x) \sqcup \neg B(x) \sqcup C(x)$
3. $\neg A(x) \sqcup \forall r.D_2(x)$
4. $\neg D_2(x) \sqcup B(x)$

5. $A(a)$

   The saturation process results in the following clauses:

6. $\forall r.D_1(a)$ (Resolution 1, 5 ; $\sigma = [x/a]$)

7. $\neg A(x) \sqcup \forall r.D_{12}(x)$ (Role Propagation 1, 3)

8. $\neg D_{12}(x) \sqcup D_1(x)$

9. $\neg D_{12}(x) \sqcup D_2(x)$

10. $\neg D_{12}(x) \sqcup B(x)$ (Resolution 9, 4)

11. $\neg D_{12}(x) \sqcup \neg B(x) \sqcup C(x)$ (Resolution 8, 2)

12. $\forall r.D_2(a)$ (Resolution 3, 5 ; $\sigma = [x/a]$)

13. $\neg D_{12}(x) \sqcup C(x)$ (Resolution 10, 11)

14. $\forall r.D_{12}(a)$ (Resolution 7, 5 ; $\sigma = [x/a]$)

where all symbols D_x (where $x \in \mathbb{N}$) are introduced definer symbols.
Observe from clauses 6 and 2, we can derive $\forall r.(\neg B \sqcup C)(a)$. Similarly, from clauses 12, and 4, we can derive $\forall r.B(a)$. From clauses 13, and 14, we can derive $\forall r.C(a)$.

Example 5.2.3. Consider the ontology O consisting of the following axioms:

1. $\exists r.A \sqsubseteq \exists r.\exists r.A$

2. $X \sqsubseteq \exists r.A$

3. $X(a)$

Observe that in the first axiom, we have that $sig(\exists r.A) \subseteq sig(\exists r.\exists r.A)$ and that $O \models \exists r.\exists r.A(a)$. It is trivial to see that resolving between axioms 2. and 3. using $RES_{\mathcal{ALC}}$ would derive $\exists r.A(a)$. We will now demonstrate that $\exists r.\exists r.A(a)$ can be derived by applying $RES_{\mathcal{ALC}}$ to O. Following [Koo15, PK15a], the structural transformation of O is:

1. $\forall r.D_1(x) \sqcup \exists r.D_2(x)$

2. $\neg D_1(x) \sqcup \neg A(x)$

3. $\neg D_2(x) \sqcup \exists r.D_3(x)$

4. $\neg D_3(x) \sqcup A(x)$

5. $\neg X(x) \sqcup \exists r.D_3(x)$

6. $X(a)$

   The saturation process results in the following clauses:

7. $\neg X(a) \sqcup \exists r.D_3(a)$ (Resolution 5, 6 ; $\sigma = [x/a]$)

8. $\exists r.D_3(a)$ (Resolution 7, 6 ; $\sigma = [x/a]$)

9. $\neg D_2(x) \sqcup \exists r.D_2(x) \sqcup \exists r.D_{13}(x)$ (Role Propagation 1, 3)

10. $\neg D_{13}(x) \sqcup D_1(x)$

11. $\neg D_{13}(x) \sqcup D_3(x)$

12. $\neg X(x) \sqcup \exists r.D_{13}(x) \sqcup \exists r.D_2(x)$ (Role Propagation 1, 5)

13. $\exists r.D_{13}(a) \sqcup \exists r.D_2(a)$ (Resolution 12, 6 ; $\sigma = [x/a]$)

14. $\neg D_{13}(x) \sqcup A(x)$ (Resolution 11, 4)

15. $\neg D_{13}(x) \sqcup \neg A(x)$ (Resolution 10, 2)

16. $\neg D_{13}(x)$ (Resolution 14, 15)

17. $\neg X(x) \sqcup \exists r.D_2(x)$ (Existential Role Restriction Elimination 16, 12)

18. $\exists r.D_2(a)$ (Existential Role Restriction Elimination 16, 13 ; $\sigma = [x/a]$)

Observe from clauses 18, 3, and 4, we can derive $\exists r.\exists r.A$.

Conjecture 1, is the justification for Step 4 of Algorithm 5.2.1 and suggests the that Algorithm 5.2.1 can extract compiled superconcepts.

Example 5.2.4 highlights the disposition of Algorithm 5.2.1 to also include redundant class expressions in an extracted compiled superconcept. Given a class expression $C = C_1 \sqcap C_2 \sqcap C_3 \sqcap \ldots \sqcap C_N$ representing a compiled superconcept, we deem a class expression C_i (for $i \in \{1,2,3,\ldots,N\}$) *redundant* if $\models C_i \sqsubseteq \bot$ or $\models \top \sqsubseteq C_i$.

Example 5.2.4. Consider the ontology

$$\alpha_1 : X \sqsubseteq B \sqcup \exists r.(D_1 \sqcap D_2) \quad \alpha_2 : D_1 \sqsubseteq A \quad \alpha_3 : B \sqsubseteq C \quad \alpha_4 : D_2 \sqsubseteq \neg A$$

A CSP for X with respect to $sig(O)$ extracted using the LETHE saturation algorithm is:

$$\begin{gathered}(\exists r.(A \sqcap D_1) \sqcup B) \sqcap \\ (\exists r.(D_1 \sqcap D_2) \sqcup B) \sqcap \\ (\exists r.(D_1 \sqcap \neg D_1) \sqcup B) \sqcap \\ (\exists r.(D_2 \sqcap \neg D_2) \sqcup B) \sqcap \\ (\exists r.(\neg A \sqcap A) \sqcup B) \sqcap \\ (\exists r.(\neg A \sqcap D_2) \sqcup B) \sqcap \\ B \sqcap C \sqcap X\end{gathered}$$

However, we have that $\models (\exists r.\bot \sqcup B) \equiv (\exists r.(D_1 \sqcap \neg D_1) \sqcup B)$, $\models (\exists r.\bot \sqcup B) \equiv (\exists r.(D_2 \sqcap \neg D_2) \sqcup B)$, $\models (\exists r.\bot \sqcup B) \equiv (\exists r.(\neg A \sqcap A) \sqcup B)$, $\models \exists r.\bot \equiv \bot$ which highlight three redundant class expressions in the result.

A better CSP is as follows:

$$\begin{gathered}(\exists r.(A \sqcap D_1) \sqcup B) \sqcap \\ (\exists r.(D_1 \sqcap D_2) \sqcup B) \sqcap \\ (\exists r.(D_1 \sqcap \neg D_1) \sqcup B) \sqcap \\ (\exists r.(\neg A \sqcap D_2) \sqcup B) \sqcap \\ B \sqcap C \sqcap X\end{gathered}$$

In order to get a more precise CSP, first we *normalise* using the rules in Figure 5.2 adopted from the description logic handbook [BCM+03]. The aim of *normalisation* is to reduce redudancies that may occur in concept expressions. For example, the concept expression $B \sqcup \neg(C \sqcup \neg C)$ contains the redundancy $\neg(C \sqcup \neg C)$ and can be simplified to B. In practice, saturation often results in such redundancies which further contributes to redundancies being present in the result of Algorithm 5.2.1. A concept expression can be normalised by applying a set of rules and transformations such as DeMorgan's laws. The normalisation rules in Figure 5.2 consist mainly of two transformations called *norm* and *simp* which are applied recursively to concept expressions to eliminate redudancies.

Example 5.2.5. Again consider the ontology

$$\alpha_1 : \mathrm{X} \sqsubseteq \mathrm{B} \sqcup \exists \mathrm{r}.(\mathrm{D}_1 \sqcap \mathrm{D}_2) \quad \alpha_2 : \mathrm{D}_1 \sqsubseteq \mathrm{A} \quad \alpha_3 : \mathrm{B} \sqsubseteq \mathrm{C} \quad \alpha_4 : \mathrm{D}_2 \sqsubseteq \neg\, \mathrm{A}$$

A CSP for X with respect to $sig(O)$ extracted using the LETHE saturation algorithm with the inclusion of the normalisation rules in Figure 5.2 is:

$$
\begin{gathered}
(\exists r.(A \sqcap D_1) \sqcup B) \sqcap \\
(\exists r.(D_1 \sqcap D_2) \sqcup B) \sqcap \\
(\exists r.(\neg A \sqcap D_2) \sqcup B) \sqcap \\
B \sqcap C \sqcap X
\end{gathered}
$$

$$
\begin{aligned}
Norm(A) &= A \text{ for atomic concept name } A \\
Norm(\neg C) &= Simp(\neg Norm(C)) \\
Norm(C_1 \sqcap \ldots \sqcap C_n) &= Simp(\sqcap\{Norm(C_1)\} \cup \ldots \cup \{Norm(C_n)\}) \\
Norm(C_1 \sqcup \ldots \sqcup C_n) &= Norm(\neg(\neg C_1 \sqcap \ldots \sqcap \neg C_n)) \\
Norm(\forall R.C) &= Simp(\forall R.Norm(C)) \\
Norm(\exists R.C) &= Norm(\neg \forall R.\neg C) \\
Simp(A) &= A \text{ for atomic concept name } A
\end{aligned}
$$

$$
Simp(\neg C) = \begin{cases} \bot & \text{if } C = \top \\ \top & \text{if } C = \bot \\ Simp(D) & \text{if } C = \neg D \\ \neg C & \text{otherwise} \end{cases}
$$

$$
Simp(\sqcap \mathbf{S}) = \begin{cases} \bot & \text{if } \bot \in S \\ \bot & \text{if } \{C, \neg C\} \sqsubseteq \mathbf{S} \\ \top & \text{if } \mathbf{S} = \emptyset \\ \text{Simp}(\mathbf{S} \setminus \{\top\}) & \text{if } \top \in \mathbf{S} \\ \text{Simp}(\sqcap \mathbf{P} \cup \mathbf{S} \setminus \{\sqcap\{\mathbf{P}\}\}) & \text{if } \sqcap\{\mathbf{P}\} \in \mathbf{S} \\ \sqcap \mathbf{S} & \text{otherwise} \end{cases}
$$

$$
Simp(\forall \mathbf{S}) = \begin{cases} \top & \text{if } C = \top \\ \forall R.C & \text{otherwise} \end{cases}
$$

Where $\sqcap \mathbf{S}$ denotes $C_1 \sqcap \ldots \sqcap C_n$ for all $C_i \in \mathbf{S}$.

Figure 5.2: Normalisation for $\mathcal{ALC}$ [BCM+03]

To saturate an ontology O, LETHE's calculus must be applied to all symbols in $sig(O)$. This is an expensive operation, as such we developed Algorithm 5.2.2 which is an optimised version of Algorithm 5.2.1. Intuitively Algorithm 5.2.2 restricts the saturation of the input ontology to symbols that occur in the ABox. Algorithm 5.2.2 can be used to extract CSPs with respect to a restricted signature by providing a uniform interpolant as the input O. Let X be a concept name in an ontology O such that

we wish to extract a CSP for X with respect to a subset signature of $sig(O)$ Σ, first we extract $SNC(X;O;\Sigma)$ and a uniform interpolant $\mathcal{V}$ for Σ under O, then we provide $\mathcal{V}$ as the input ontology O and $SNC(X;O;\Sigma)$ as the input concept X to Algorithm 5.2.2.

Complexity Algorithm 5.2.2 includes a saturation process that produces a set of clauses that can be double exponential in size relative to the input ontology [Koo15]. The saturation process is executed at most N times where N is the number of symbols in the ontology, thus adding a linear complexity to Algorithm 5.2.2. Thus Algorithm 5.2.2 has a worst case double exponential size complexity relative to the size of the input ontology.

ALGORITHM 5.2.2

Input: 1. An ontology O consisting only of TBoxes 2. A concept X.
Output: A Compiled Superconcept C of X.

---

1. Add $X(a)$ to O to get O'.
2. Structurally transform O' following [PK15d] to get a set of clauses $\mathcal{N}$.
3. initialise an empty queue Q of symbols to be saturated.
4. initialise an empty set $\mathcal{F}$ to keep track of symbols that have been saturated.
5. **for** every clause C in $\mathcal{N}$ with a in its signature **do:**
   for every role or concept symbol $S \in sig(C)$ **do:**
   if $S \notin \mathcal{F}$ **do:**
   · $Q \leftarrow Q \cup S$
   endfor
   while $Q \neq \emptyset$ **do:**
   · $S' = dequeue(Q)$
   · $\mathcal{F} \leftarrow \mathcal{F} \cup S'$
   · Perform all possible inferences on S' to the set $\mathcal{N}$ using the rules in Figure 2.3 and add all inferences to $\mathcal{N}$.
   for every clause C in $\mathcal{N}$ with a in its signature **do:**
   for every role or concept symbol $S \in sig(C)$ **do:**
   if $S \notin \mathcal{F}$ **do:**
   · $Q \leftarrow Q \cup S$
   · Eliminate all definer symbols from $\mathcal{N}$ following [PK15d]
   endwhile
   endfor
6. Reverse the structural transformation on $\mathcal{N}$ to get $\mathcal{V}_S$
7. Normalise $\mathcal{V}_S$ using the rules in Figure 5.2.
8. Return a conjunction of all concepts C for every $C(i) \in \mathcal{V}_S$ to get $CSP(X;O;\Sigma)$.

5.2.2 Conveying Information about Roles

All the methods discussed so far are designed to convey concept names in an ontology; however, there are instances in which it may be useful for an agent to have the ability to convey some information about the roles in its ontology. In description logics that allow for constructs such as role-hierarchies and role-chaining, roles may be described using other roles. However, since we restrict our investigation to $\mathcal{ALC}$, we can only describe roles using concept names. Specifically, in $\mathcal{ALC}$ and less expressive description logics such as $\mathcal{EL}$, we may describe a role in terms of approximates of its domain and range because they give some information about what can be inferred from role assertions. This section studies and develops algorithms based on uniform interpolation that can be used to extract approximates of the domain and range of a role.

Let r be a role, the domain of r is $\exists r.\top$, and the range of r is $\exists r^{-}.\top$. However, because we are interested in conveying the domain of r without using r, we rely on finding concepts C such that $r \notin sig(C)$ and $\exists r.\top \sqsubseteq C$ or a strongest necessary condition of all such concepts C to find the closest approximation of the domain of r. Similarly, because we are interested in conveying the range of r without using r, we rely on finding concepts D such that $r \notin sig(D)$ and $\top \sqsubseteq \forall r.D$ (because $\exists r^{-}.\top$ is not easily expressible in $\mathcal{ALC}$) or a strongest necessary condition of all such concepts D to find the closest approximation of the domain of r.

Even if the approximation of domain or range of a role is explicitly stated, there is no guarantee that the signature used is within the common signature of communicating agents. Let O be $\{C \equiv \exists p.A, \exists r.\top \sqsubseteq C \sqcup D, D \equiv B\}$ and let $\Sigma = \{p, A, B\}$. Then it follows that an approximation of domain for r only using symbols in Σ is $\exists p.A$. In such a case, the approximate domain (or range) of r must be restricted to the common vocabulary.

There may be several approximate domains and ranges for r in O. With concept names, we aim at capturing the maximally precise superconcepts using SNCs and maximally precise subconcepts using WSCs. Similar to the case with concept names and SNCs, WSCs, we want to be able to capture maximally all approximate domains (ranges) of r using a single class expression; to capture this, we have formally defined notions of *strongest domain* (Definition 36) and *strongest range* (Definition 37).

Definition 36 ((Strongest) Domain of a Role)**.** *Let Σ be a set of $\mathcal{ALC}$ concept names and role symbols, r a role, and O an ontology, such that $\Sigma \subseteq sig(O)$ and $r \notin \Sigma$. We define a* domain *of r over Σ* relative *to O to be any $\mathcal{ALC}$ concept C such that $sig(C) \subseteq \Sigma$ and $O \models \exists r.\top \sqsubseteq C$. C is a strongest domain, denoted $dom_s(r; O; \Sigma)$, if for any other*

domain C' of r over Σ relative to O we have that $O \models C \sqsubseteq C'$. C is a minimal *strongest domain, denoted $dom_s^M(r;O;\Sigma)$ if for any strongest domain C' of r over Σ relative to O we have that $sig(C) \subseteq sig(C')$.*

Definition 37 ((Strongest) Range of a Role)**.** *Let Σ be a set of $\mathcal{ALC}$ concept names and role symbols, r a role, and O an ontology, such that $\Sigma \subseteq sig(O)$ and $r \notin \Sigma$. We define a range of r over Σ relative to O to be any $\mathcal{ALC}$ concept C such that $sig(C) \subseteq \Sigma$ and $O \models \top \sqsubseteq \forall r.C$. C is a strongest range, denoted $ran_s(r;O;\Sigma)$ if for any other range C' of r over Σ relative to O we have that $O \models C \sqsubseteq C'$. C is a* minimal *strongest range, denoted $ran_s^M(r;O;\Sigma)$, if for any strongest range C' of r over Σ relative to O we have that $sig(C) \subseteq sig(C')$.*

Since the domain of a role r is the concept $\exists r.\top$, a strongest domain of r is equivalent to a strongest necessary condition of $\exists r.\top$ with respect to a signature that does not contain r. Thus, Algorithm 5.1.1 may be repurposed to extract a strongest domain of a role r by adding $\exists r.\top$ as the input X to the algorithm.

We use uniform interpolation to extract a strongest range of a role. Proposition 12 enables us to extract ranges using uniform interpolation. Let O be an ontology and r a role in O, to extract the range for r under O with respect to a signature Σ, we add the assertion $r(a,b)$ to O where a and b are fresh individuals and extract a uniform interpolant $\mathcal{V}$ for O with respect to $\Sigma \setminus \{r\}$. A consequence of Proposition 12 and Definition 18 is that the conjunction of concepts D of the form $D(b) \in \mathcal{V}$ form a strongest range of r for Σ with respect to O.

Proposition 12. *Let O be an $\mathcal{ALC}$ ontology, r a role in $sig(O)$, Σ a subset of $sig(O)$, and a, b fresh individuals such that $a \notin sig(O)$, $b \notin sig(O)$, and $r \notin \Sigma$. Let $\mathcal{V}$ be a uniform interpolant with respect to $O \cup r(a,b)$ for Σ. We have that $\mathcal{V} \models D(b)$ iff $O \models \top \sqsubseteq \forall r.D$ for any D such that $sig(D) \subseteq \Sigma$.*

Proof. If $\mathcal{V} \models D(b)$, then $O \models \top \sqsubseteq \forall r.D$: Since $\mathcal{V}$ is a uniform interpolant for $\Sigma \cup \{a,b\}$ of $O \cup r(a,b)$, we have that $O \cup r(a,b) \models \mathcal{V}$. Given that $\mathcal{V} \models D(b)$, and $O \cup r(a,b) \models \mathcal{V}$, we have that $O \cup r(a,b) \models D(b)$. Therefore, it follows that $O \cup r(a,b), \neg D(b)$ is unsatisfiable, and hence $O \models \top \sqsubseteq \forall r.D$ since $O \models \top \sqsubseteq \forall r.D$ iff $O \cup r(a',b'), \neg D(b') \models \bot$ for some fresh individuals a' and b'.

If $O \models \top \sqsubseteq \forall r.D$, then $\mathcal{V} \models D(b)$. Assume $O \models \top \sqsubseteq \forall r.D$. Since $\mathcal{V}$ is a uniform interpolant for $\Sigma \cup \{a,b\}$ of $O \cup r(a,b)$, we have that $O \cup r(a,b) \models \mathcal{V}$. Since $O \models \top \sqsubseteq \forall r.D$, then it follows that $O \cup r(a,b) \models D(b)$. Since $sig(D) \subseteq \Sigma$, and $\mathcal{V}$ is a uniform

interpolant for $O \cup r(a,b)$ with respect to $\Sigma \cup \{a,b\}$, it follows from Definition 18 that $\mathcal{V} \models D(b)$. □

Algorithm 5.2.3 details how a strongest range of a role can be extracted using uniform interpolation.

ALGORITHM 5.2.3

Input: An $\mathcal{ALC}$ ontology O, a role r, a signature Σ, a pair of individuals a and b where $r \notin \Sigma$, $a \notin sig(O)$, $b \notin sig(O)$.

Output: A concept D which is a strongest range of r.

Step 1: Add $r(a,b)$ to O to get O'.

Step 2: Extract a uniform interpolant denoted $\mathcal{V}$ with respect to O' for $\Sigma \cup \{a,b\}$.

Step 3: $\bigsqcap_{D_i(b) \in abox(\mathcal{V})} D_i$.

Complexity

Algorithm 5.2.3 calls the forgetting algorithm at most once, and is thus bounded by the computational complexity of the forgetting algorithm used. Extracting a uniform interpolant using LETHE has been shown to have a 2EXPTIME worst-case space complexity [WDP19, PK15c] in the number of symbols contained in the ontology; thus our implementation of Algorithm 5.2.3 has a 2EXPTIME worst-case space complexity.

Remark 2. *Although the implementation of LETHE is interpolation complete, in practice, we find that its ability to forget roles is not robust. This is because role forgetting is a difficult problem in general. Several of our experiments with the LETHE implementation on ontologies have demonstrated that forgetting roles using the implementation is not robust enough to conduct experiments, as it sporadically fails to terminate (for days sometimes) when attempting to forget a role in some instances. This is taken into account in the design of all the experiments in this thesis.*

Up to this point in the thesis we have discussed and developed several knowledge structures and extraction methods. Tables 5.1 and 5.2 summarise the key inputs and outputs of relevant knowledge structures or extraction methods discussed so far.

5.3 Contribution Summary

The contributions of this chapter are as follows:

- A formal description of strongest necessary conditions and weakest sufficient conditions in description logic, alongside appropriate algorithms for extraction of the conditions (for the DL $\mathcal{ALC}$).
- The development and introduction of compiled superconcepts and compiled subconcepts which account for potential shortcomings of strongest necessary conditions and weakest sufficient conditions.
- Criteria for where strongest necessary and weakest sufficient conditions exist.
- A novel algorithm for extracting the domains and ranges of roles in $\mathcal{ALC}$ ontologies.

Structure or method	Input(s)	Output	Output constraints	Existence
Top Module	1. An ontology O, 2. A signature Σ such that $\Sigma \subseteq sig(O)$.	A subset O' of O	1. $sig(O')$ may exceed Σ. 2. For all α such that $sig(\alpha) \subseteq sig(O')$, $O' \models \alpha$ iff $O \models \alpha$.	Always
Bottom Module	1. An ontology O, 2. A signature Σ such that $\Sigma \subseteq sig(O)$.	A subset O' of O	1. $sig(O')$ may exceed Σ. 2. For all α such that $sig(\alpha) \subseteq sig(O')$, $O' \models \alpha$ iff $O \models \alpha$.	Always
Uniform interpolant	1. An ontology O, 2. A signature Σ such that $\Sigma \subseteq sig(O)$.	An ontology $\mathcal{V}$	1. $sig(\mathcal{V}) \subseteq \Sigma$, 2. For any α such that $sig(\alpha) \subseteq \Sigma$, $O \models \alpha$ iff $\mathcal{V} \models \alpha$.	Not expressible in some DLs
Craig interpolant	1. An ontology O, 2. Two concepts C_1, C_2.	A concept I	1. $sig(I) \subseteq sig(C_1, O) \cap sig(C_2, O)$, 2. $O \models C_1 \sqsubseteq I$, 3. $O \models I \sqsubseteq C_2$.	Not expressible in some DLs
SNC	1. An ontology O, 2. A signature Σ such that $\Sigma \subseteq sig(O)$. 3. A concept C.	A concept C'	1. $O \models C \sqsubseteq C'$ 2. $sig(C') \subseteq \Sigma$ 3. For any D such that $sig(D) \subseteq \Sigma$ and $O \models C \sqsubseteq D$, $O \models C' \sqsubseteq D$	Not expressible in some DLs
WSC	1. An ontology O, 2. A signature Σ such that $\Sigma \subseteq sig(O)$. 3. A concept C.	A concept C'	1. $O \models C' \sqsubseteq C$ 2. $sig(C') \subseteq \Sigma$ 3. For any D such that $sig(D) \subseteq \Sigma$ and $O \models D \sqsubseteq C$, $O \models D \sqsubseteq C'$	Not expressible in some DLs

Table 5.1: Knowledge Structures and Knowledge Extraction Methods in Logic

Structure or method	Input(s)	Output	Output constraints	Existence
CSP	1. An ontology O, 2. A signature Σ such that $\Sigma \subseteq sig(O)$. 3. A concept C.	A concept C'	1. $O \models C \sqsubseteq C'$ 2. $sig(C') \subseteq \Sigma$ 3. For any D such that $sig(D) \subseteq \Sigma$ and $O \models C \sqsubseteq D$, $\models C' \sqsubseteq D$	Not expressible in some DLs
CSB	1. An ontology O, 2. A signature Σ such that $\Sigma \subseteq sig(O)$. 3. A concept C.	A concept C'	1. $O \models C' \sqsubseteq C$ 2. $sig(C') \subseteq \Sigma$ 3. For any D such that $sig(D) \subseteq \Sigma$ and $O \models D \sqsubseteq C$, $\models D \sqsubseteq C'$	Not expressible in some DLs
Strongest domain	1. An ontology O, 2. A signature Σ such that $\Sigma \subseteq sig(O) \setminus \{r\}$. 3. A role r.	A concept C	1. $O \models \exists r.\top \sqsubseteq C$ 2. $sig(C) \subseteq \Sigma$ 3. For any C' such that $sig(C') \subseteq \Sigma$ and $O \models \exists r.\top \sqsubseteq C'$, $O \models C \sqsubseteq C'$	Not expressible in some DLs
Strongest range	1. An ontology O, 2. A signature Σ such that $\Sigma \subseteq sig(O) \setminus \{r\}$. 3. A role r.	A concept D	1. $O \models \top \sqsubseteq \forall r.D$ 2. $sig(D) \subseteq \Sigma$ 3. For any D' such that $sig(D') \subseteq \Sigma$ and $O \models \top \sqsubseteq \forall r.D'$, $O \models D \sqsubseteq D'$	Not expressible in some DLs

Table 5.2: Knowledge Structures and Knowledge Extraction Methods in Logic

Chapter 6

Conveying Maximally Precise Descriptions

In Chapter 5 we discussed and proposed algorithms for extracting knowledge from a knowledge base with respect to a subset signature of the knowledge base. The methods discussed and developed were all based on uniform interpolation. We have shown that existing algorithms for extracting strongest necessary and weakest sufficient conditions in other logic formalisms (such as first-order logic and propositional logic) can not be simply adopted in a description logic setting.

A simplifying assumption in this thesis is that the common signature for a pair of communicating agents is already established. However, as we do not make any assumptions on the interpretation of symbols in the common signature, we have also discussed and developed other knowledge extraction notions, namely *compiled superconcepts*, and *compiled subconcepts*. Compiled superconcepts and compiled subconcepts are aimed at capturing knowledge about non-common concept names in an ontology in a fashion that does not rely on the background information in the ontology.

Most of the methods discussed and developed rely on a uniform interpolation method that can extract uniform interpolants from ontologies that have concept assertions. We aim to practically evaluate the developed methods using existing implementations of LETHE thus our analysis is restricted to $\mathcal{ALC}$ ontologies. This is because, at the time of writing, the LETHE implementation available could not extract uniform interpolants from ontologies that had concept assertions or were more expressive than $\mathcal{ALC}$.

This chapter tries to answer **RQ5** "*What is the most precise way to convey a term with respect to a subset vocabulary of an ontology?*" (see Section 1.2). Because the

definition of a concept name is synonymous with the concept name, a concept name's definition is its maximally precise description with respect to the ontology of interest: a definition C of a concept X is a maximal subsumer and maximal subsumee of X. Alternatively, a compiled superconcept may be viewed as a maximally precise subsumer of a concept name with respect to the ontology of interest *and* the empty ontology.

For example, given the ontology $O = \{A \equiv B, D \sqsubseteq A, A \sqsubseteq C\}$, and signature $\Sigma = \{B, C, D\}$, B is a better description of A with respect to Σ than C or D because B is equivalent to A: in every interpretation that is a model of O, the set of instances of A will be the be equal to that of B; however, the instances of A will be a subset of the instances of C and the instances of D will be a subset of the instances of A. Similarly, the set of instances of $B \sqcap C$, (which is a compiled superconcept of A, and equivalent to A under O) will be equal to that of A.

The first section of this chapter empirically evaluates definition extraction for concept names using strongest necessary conditions, and thus this chapter also tries to answer **RQ2** "*How do the relevant knowledge extraction techniques perform in practice on real-world ontologies?*". We are primarily interested in definitions and definability in this chapter. Given the computational cost of uniform interpolation, the first point of inquiry in our evaluation is how successful the Algorithm 5.1.1 is when it comes to extracting definitions for concept names occurring in general TBoxes. However, the cost of uniform interpolation is not the only source of concern for extracting strongest necessary conditions. Least fixpoints operators are needed to represent uniform interpolants of ontologies that have general TBoxes with *cycles* (see Section 2.2.3). In such TBoxes, definitions extracted by strongest necessary conditions following Algorithm 5.1.1 may contain greatest fixpoint operators or definer symbols. We do not consider such definitions to be valid as we have no practical means (such as off-the-shelf reasoners) to perform reasoning with fixpoints. The last point of concern is the in terms of signature size of the definitions extracted using strongest necessary conditions. Because the implementation of uniform interpolation has not been optimised for redundancy elimination, there is a likelihood for superfluous information to be contained in the definitions extracted. Signature size could be an indicator of superfluous information as the process of saturation (used to implement the uniform interpolation algorithm) can result in tautologous concepts being included in axioms. Thus, we also evaluate the signature size of the extracted definitions. Because this may vary depending on the common signature, we focus the evaluation with respect to the signature of the entire ontology minus the concept name of interest.

The second section deals with evaluating the extraction of definitions with respect to the primitive signature of an ontology. To the best of our knowledge there are no publicly available datasets of a pair of ontologies (representing the knowledge of agents) with overlapping vocabularies. Thus, we use the primitive signature of an ontology to emulate the signature of an ontology that an agent has in common with its communication partner. Let O be an ontology. Here, we understand the primitive signature of O to be the set of all symbols S such that there exists no explicit or implicit definition C for S with respect to $sig(O) \setminus \{S\}$. Let C be an uncommon term that an agent AG_1 wishes to convey to an agent AG_2, such that $sig(C) \not\subseteq \Sigma$ and Σ is the vocabulary AG_1 has in common with AG_2. If C has a definition D such that $sig(D) \subseteq \Sigma$ under AG_1's ontology, then AG_1 may synonymously convey C to AG_2 (again, recall that this chapter may be viewed as an exploration of **RQ5** from Section 1.2). This chapter explores (empirically) whether all defined concept names have definitions that can be expressed in the primitive signatures of ontologies. If this proves to be the case, the agent can focus its efforts on making sure (perhaps via ontology alignment) that its primitive signature is common with any communication partner. Doing so will ensure that all defined concept names will have definitions with respect to the primitive signature and give less likelihood for misunderstanding between the agents.

Definitions may also help agents find hidden correspondences [Gel18]. Let A be a concept name in AG_1's ontology that has definition C such that $sig(C) \subseteq \Sigma$ and Σ is AG_1's primitive signature which it has in common with a communication partner AG_2 [Gel18]. If there is a defined concept name Z in AG_2's vocabulary which is equivalent to C, then AG_1 and AG_2 may create a correspondence between A and Z stating that $A \equiv Z$ [Gel18]. Furthermore, if there was no corresponding term Z in AG_2's ontology, then definition extraction will enable AG_1 to convey A to AG_2. Thus, this chapter may also be viewed as an exploration of **RQ7** "*What are the considerations on the common or subset vocabulary that should be accounted for when conveying terms?*". As such, we investigate the definability of concept names under the primitive signature of ontologies, as well as the natural occurrence of primitive symbols in definitions of explicitly defined concept names.

In the final section of this chapter, we empirically evaluate the extraction of compiled superconcepts to understand and report how our algorithm for extracting them performs in practice.

6.1 The corpus

We obtained the code used by Matentzoglu [Mat16] to retrieve ontologies from the BioPortal ontology repository [WNS+11] and restricted the dataset to ontologies whose expressivity did not exceed $\mathcal{ALCH}$. The code used by Matentzoglu [Mat16] uses the API feature of BioPortal [1] to download ontologies. For each ontology on BioPortal, we downloaded the latest snapshot [2] with a forty-five minute timeout for downloads. Any ontology that could not be downloaded or parsed using the OWL API [HB11] (version 5.1.7) was discarded. Of all the ontologies downloaded, 81 were in $\mathcal{ALCH}$. For ontologies with role hierarchies, we deleted the role hierarchies present. Of the 81 ontologies, twenty had defined concepts (this was detected using an implementation of implicit definability test described in Section 2.2.1). Our experiments focus on these twenty ontologies.

Although this may be considered a relatively small dataset, the ontologies vary in size and expressivity; full details are presented in Table 6.1. The dataset had to be further reduced to sixteen ontologies because the ontologies $ONTOMA$, EO, $EPIE$ and $EPIP$ only had unsatisfiable defined concepts, i.e., all the defined concepts in $ONTOMA$, EO, $EPIE$ and $EPIP$ were equivalent to $\bot$.

[1] `https://bioportal.bioontology.org/`

[2] The latest snapshot as of 21 March 2021.

Ontology ID	DL	Class Count	Role Count	Defined	Expl Defs	Impl Defs
MHCRO	$\mathcal{ALEH}$	2370	25	1254	1253	0
PDON	$\mathcal{ALE}$	632	7	630	630	0
TCO	$\mathcal{ALE}$	578	3	558	558	0
CHMO	$\mathcal{ALCH}$	2963	25	102	102	0
RXNO	$\mathcal{ALCH}$	866	14	71	71	0
ACESO	$\mathcal{ALC}$	295	14	67	67	0
HPIO	$\mathcal{ALCH}$	275	6	36	13	23
MOP	$\mathcal{ALCH}$	3635	10	31	31	0
~~EPIE~~	~~$\mathcal{ALC}$~~	~~392~~	~~0~~	~~17~~	~~0~~	~~17~~
~~EPIP~~	~~$\mathcal{ALC}$~~	~~263~~	~~0~~	~~17~~	~~0~~	~~17~~
BNO	$\mathcal{AL}$	100	0	14	0	14
~~ONTOMA~~	~~$\mathcal{ALC}$~~	~~424~~	~~0~~	~~13~~	~~3~~	~~10~~
CCTOO	$\mathcal{ALC}$	1133	6	11	8	3
ONTOPARON SOCIAL	$\mathcal{ALEH}$	1282	6	7	7	0
HP_O	$\mathcal{ALC}$	199	5	5	5	0
ISSVA	$\mathcal{ALC}$	194	1	3	2	1
DLORO	$\mathcal{ALE}$	295	5	2	2	0
GECKO	$\mathcal{ALE}$	154	3	2	2	0
~~EO~~	~~$\mathcal{ALC}$~~	~~768~~	~~0~~	~~1~~	~~0~~	~~1~~
MEDEON	$\mathcal{ALE}$	66	4	70	0	1
XEO	$\mathcal{ALE}$	145	4	243	0	1

Table 6.1: Main ontology corpus. The ontologies crossed out in red are ontologies that contain only unsatisfiable defined concept names.

6.2 Evaluating Definition Extraction using SNCs

Definition 38 (Explicitly defined Concepts, Explicit Definitions and Implicitly Defined Concepts)**.** *Let O be an ontology. For every concept name $A \in sig(O)$, we say A is* implicitly defined *if there exists an explicit definition C of A such that $O \models A \equiv C$ and $A \notin sig(C)$. For every concept name $A \in sig(O)$, we call A* explicitly defined *if A is implicitly defined, and there exists an axiom of the form $A \equiv C \in O$ such that $A \notin sig(C)$. For an explicitly defined concept A such that an axiom $A \equiv C \in O$, we call C an* explicitly stated definition *of A. The set of defined concept names in an ontology is the set of implicitly defined concept names.*

We define a partition of concept names in an ontology into defined concepts and the primitive signature.

Definition 39 (Defined concepts and Primitive Signature)**.** *Let Σ_D denote the set of defined concept names in an ontology O. The primitive signature Σ_P is the rest of the ontology signature, i.e., $\Sigma_P = sig(O) \setminus \Sigma_D$.*

6.2.1 Results on extracting definitions for explicitly defined concept names

The aim of the evaluation in this section is to investigate how successful definition extraction is, in the context of general TBoxes. Some of the ontologies in the corpus contain cyclic concept names that are definable. To the best of our knowledge there is no known decision procedure for detecting *interesting* cycles (See Section 2.2.1), and furthermore, Algorithm 5.1.1 can not extract definitions that may involve cyclic concept names. Therefore, by *successful* definition extraction, we mean that for an explicitly defined concept name X, Algorithm 5.1.1 extracted an explicit definition C of X such that $X \notin sig(C)$, $sig(C) \subseteq sig(O)$, $O \models X \equiv C$, and there were no definer symbols in $sig(C)$. Another point of inquiry in this evaluation is to investigate the degree to which superfluous symbols (or expressions) exist in definitions extracted using strongest necessary conditions when compared to explicitly stated definitions. This naturally restricts the scope to explicitly defined concepts in the ontologies of interest.

For a given ontology with explicitly defined concept names, the experiment pipeline was as follows:

1. **for** every satisfiable explicitly defined concept $A \in O$ **do:**
   (a) Extract any equivalent class axioms $A \equiv C$ for A.
   (b) Find the definition C with the smallest signature to use as the definition φ of A.
   (c) Extract a strongest necessary condition of A with respect to $sig(O) \setminus \{A\}$ under O, i.e., $SNC(A; O; sig(O) \setminus \{A\})$ using Algorithm 5.1.1.
   (d) Record the signature sizes of φ and $SNC(A; O; sig(O) \setminus \{A\})$.

The results are displayed in Table 6.2, Table 6.3, Figure 6.1, and Figure 6.2. Table 6.2 displays the success rate of extracting definitions for each explicitly defined concept in each ontology in the corpus using strongest necessary conditions extracted by Algorithm 5.1.1. Table 6.3 illustrates the magnitude of the difference in signature size between the definitions extracted using strongest necessary conditions (Algorithm 5.1.1), and the definitions explicitly stated in the ontology. The disparities in signature sizes for ontologies with more than fifty explicitly defined concepts are shown in Figures 6.1 and 6.2. The reason for splitting the results into two different figures is to maintain a scale where results can be distinguished and observed properly.

Ontology	Explicitly defined concepts	Extracted Definitions	Percentage
CHMO	102	102	100%
RXNO	71	71	100%
ACESO	67	67	100%
MHCRO	1254	1251	99%
TCO	558	554	99%
PDON	630	607	96%
MOP	31	31	100%
CCTOO	8	8	100%
ONTOPARON	7	7	100%
HP_O	5	5	100%
ISSVA	2	2	100%
DLORO	2	2	100%
GECKO	2	2	100%

Table 6.2: Success Rate of SNC definition extraction

From Table 6.2, we observe that definitions extracted using strongest necessary conditions are successful: definitions were extracted for more than 90 percent of explicitly defined concepts in all of the ontologies. In cases with less than a 100 percent

success rate such as MHCRO, TCO, PDON, the failure was due to the explicitly defined concepts being cyclic.

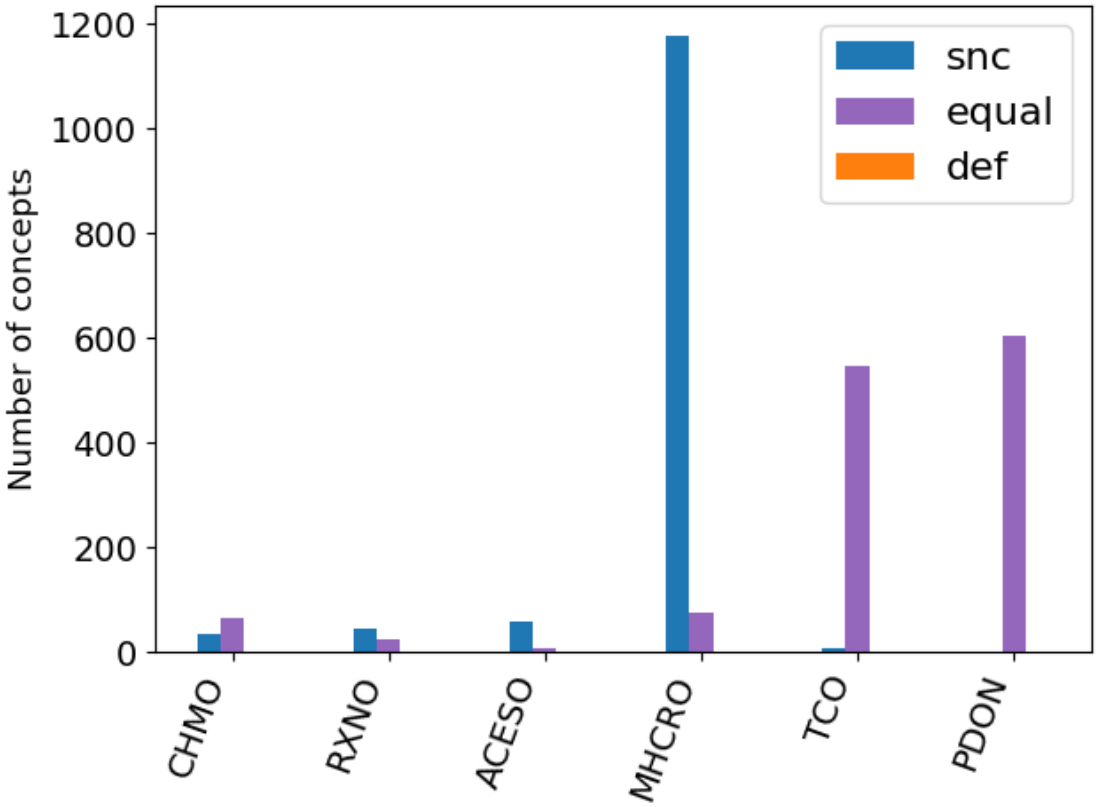

Figure 6.1: Compared signature sizes for ontologies with more than fifty defined concepts. The purple bar indicates the number of concepts for which the signature size of the SNC was equal to the signature size of the explicitly stated definition. The blue bar indicates the number of concepts for which the signature size of the SNC was greater than the signature size of the explicitly stated definition. The orange bar indicates the number of concepts for which the signature size of the SNC was greater than the signature size of the explicitly stated definition.

The charts in Figure 6.1 and 6.2 illustrate the difference in signature sizes: blue bars are cases in which the signature size of the extracted necessary condition was greater than the signature size of the explicitly stated definition, purple bars indicate cases in which the signature sizes were equal, and orange bars indicate cases in which the signature size of the explicitly stated definition was greater than that of the extracted strongest necessary condition. It is clear that in most cases (except from TCO, PDON, MOP, and ONTOPARON_SOCIAL), there is some superfluous information in the strongest necessary condition. This is partly because there is no constraint on

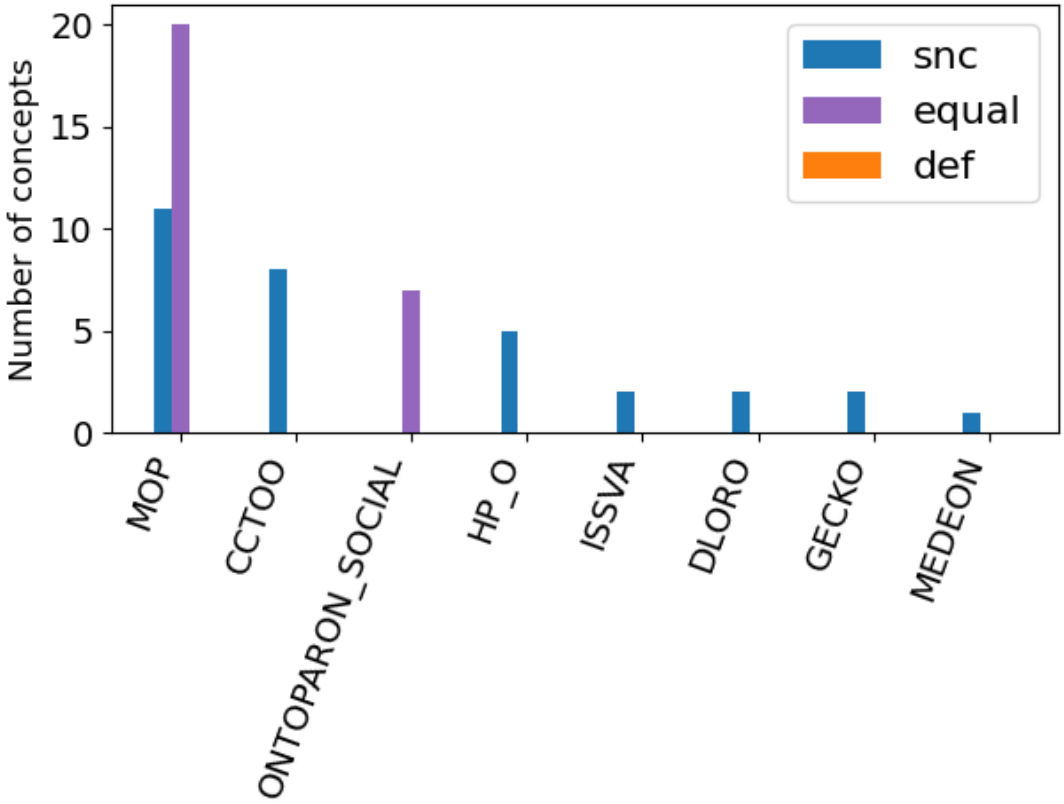

Figure 6.2: Signature sizes for ontologies with less than fifty defined concepts.

strongest necessary conditions that enforces minimality (in terms of signature). Another factor that may contribute to the presence of superfluous information is that the implementation of LETHE used is not aimed at conciseness. Example 6.2.1 provides an illustration.

Example 6.2.1. Let O be:

$$\alpha_1 : \mathrm{A} \equiv \exists \mathrm{p.B} \qquad \alpha_2 : \mathrm{A} \sqsubseteq \mathrm{C}$$

The definition of A in O is $\exists p.B$; however, a strongest necessary condition of A with respect to $sig(O) \setminus \{A\}$ under O (extracted using LETHE) is $C \sqcap \exists p.B$. We have that $O \models A \equiv C \sqcap \exists p.B$, and thus a strongest necessary condition is a definition for A; however, a more concise definition is $\exists p.B$, and C can be considered superfluous information in the definition.

Table 6.3 illustrates the magnitudes of difference in the signature size of definitions extracted using strongest necessary conditions and that of explicitly stated definitions.

We observe that the signature sizes are equal a signficant number of times; however, when there is a signature difference, it is mostly as a result of the strongest necessary conditions having more symbols. Interesting cases include ACESO, ISSVA, and DLORO, in which we observe that there are equivalent atomic concepts for the explicitly defined concepts; however, the extracted strongest necessary conditions still contain more symbols. These cases in specific stress the cost (in terms of signature) of Algorithm 5.1.1. This highlights the fact that using a reasoner to extract equivalent concept names may be preferable to Algorithm 5.1.1 for definition extraction tasks in cases where there are equivalent atomic concepts in the common signature for the concept of interest.

Ontology	Q1		Q2		Q3		Q4	
	SNC	DEF	SNC	DEF	SNC	DEF	SNC	DEF
CHMO	3.0	3.0	3.0	3.0	4.0	3.0	3.0	3.0
RXNO	3.0	3.0	4.0	3.0	5.0	3.0	3.0	2.0
ACESO	2.0	1.0	3.0	1.0	5.0	4.0	1.0	2.0
MHCRO	6.0	5.0	7.0	6.0	10.0	8.0	4.0	3.0
TCO	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
PDON	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
MOP	3.0	3.0	3.0	3.0	4.0	3.0	3.0	3.0
CCTOO	5.75	2.0	6.0	2.0	7.0	3.0	5.0	2.0
ONTOPARON	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
HP_O	2.0	1.0	2.0	1.0	3.5	1.5	2.0	1.0
ISSVA	2.0	1.0	2.0	1.0	2.0	1.0	2.0	1.0
DLORO	3.25	1.0	4.5	1.0	5.75	1.0	2.5	1.0
GECKO	3.0	2.0	3.0	2.0	3.0	2.0	3.0	2.0

Table 6.3: Quantiles of signature sizes. The column titled ‘Q1‘ denotes the first quantile, the column titled ‘Q2‘ denotes the second quantile, while the column titled ‘Q3‘ denotes the third quantile. The column titled ‘SNC’ denotes the quantile of signature size of the SNC for the corresponding ontology. The column titled ‘DEF’ denotes the quantile of signature size of the explicitly stated definition for the corresponding ontology.

6.3 Evaluating Definition Extraction using SNC w.r.t Primitive Signatures for General TBoxes

As discussed in Section 2.2.1, given a terminology O, we may partition the signature of O into a set of *named* concepts, and *base* symbols. In a terminology, explicitly defined concepts can be unfolded (see Section 2.2.1) into equivalent concepts whose signature consist solely of base symbols. This implies that in a terminology, the strongest necessary condition of any explicitly defined concept with respect to the base symbols must be a definition for the explicitly defined concept. Given an agent with a terminology, if all base symbols are shared with another agent, all uncommon named symbols can be conveyed without any loss of information. Thus for an agent with a terminology that has a large proportion of defined concepts, it may be better to focus on explicating or aligning the base symbols if it is to be engaged in any long-term conversation with another agent.

Similar to the partition of *name* and *base* symbols in a terminology, we may partition general TBoxes into a set of defined concepts and *primitive* concepts using the implicit definability test as stated in Definition 4. Because $\mathcal{ALC}$ has the Beth Definability property (see Section 2.2.1), we can be confident that regardless of whether a concept is explicitly or implicitly definable under a signature, an explicit definition can be extracted for the concept, thus evaluating the extraction of definitions with respect to the primitive signature for general TBoxes is interesting.

Given the partitioning of an ontology's signature into a primitive signature and a defined signature, we may further partition the primitive signature into two sets, namely (i) primitively defined signature (ii) not-primitively defined signature.

Definition 40 (Primitively Defined Signature)**.** *Let Σ_D be the set of defined concept names in $sig(O)$ and let Σ_P be the set of primitive concept names in $sig(O)$. The primitively defined signature Σ_D^P of O is the subset of Σ_D such that for each concept name $A \in \Sigma_D^P$, there exists a definition C of A under O where $sig(C) \subseteq \Sigma_P$.*

Definition 41 (Not-Primitively Defined Signature)**.** *Let O be an ontology, let Σ_D be the set of defined concept names in $sig(O)$ and let Σ_P be the primitive signature in $sig(O)$. The not-primitively defined signature Σ_D^N of O denotes the set $\Sigma_D \setminus \Sigma_D^P$.*

As a trivial example, consider the ontology $\{\mathrm{A} \equiv \mathrm{B}\}$. We have that A and B both pass the implicit definability test, thus making the set of defined concepts $\{\mathrm{A}, \mathrm{B}\}$, and the primitive signature $\emptyset$, hence no defined concept can be defined in terms of the

primitive signature $\emptyset$ [3]. However, both A and B are defined, thus the not-primitively defined signature here is $sig(O)$ and the primitive signature is the empty set. It is worth noting that this phenomenon is not unique to atomic equivalences; as an example, consider the ontology $\{A \equiv B \sqcup C, B \sqsubseteq \neg C\}$ which entails that $B \equiv A \sqcap \neg C$ and $C \equiv A \sqcap \neg B$, we have that all concept names are implicitly definable and thus the primitive signature is the empty set, while the not-primitively defined signature is $sig(O)$. The partition of the ontology's signature into a primitively defined signature and a not-primitively defined signature is relevant when considering extracting definitions with respect to the primitive signature because it is impossible to extract a definition for any concept name in the not-primitively defined signature.

However, observe that it is possible to treat some of the not-primitively defined concepts in the signature as though they were primitive, and in doing so make some of the not-primitively defined signature definable. Again, consider the ontology $\{A \equiv B \sqcup C, B \sqsubseteq \neg C\}$, if B and C are considered primitive, then A becomes definable. Indeed this is an approach taken in partitioning terminologies into named and base symbols: again consider the terminology $\{A \equiv B\}$. Here the not-primitively defined signature is $sig(O)$; however, B is considered a base symbol as it does not occur on the left hand side of an equivalence. Similarly, in general TBoxes, we may treat some of not-primitively defined signature as primitive in order to make some of the not-primitively defined signature definable. As an example, consider the ontology O with the following axioms:

α_1: $\mathrm{A}_1 \equiv \mathrm{B}_1 \sqcap \mathrm{B}_2$

α_2: $\mathrm{B}_1 \equiv \mathrm{B}_3$

α_3: $\mathrm{A}_2 \equiv \mathrm{A}_1 \sqcup \mathrm{A}_3$

The primitive signature is $\Sigma_P = \{B_2, A_3\}$; however, A_1, B_1, B_3, and A_2 are not definable under Σ_P. However, if we add A_1 to Σ_P, then A_2 becomes definable. If we add B_3 to Σ_P, then A_1, B_1, and A_2 become definable under Σ_P. Notice that depending on the choice of which symbol from the not-primitively defined signature we decide to treat as primitive, the size of the not-primitively defined signature can shrink greatly, in the example above adding A_1 to Σ_P only made A_2 definable under Σ_P; however, adding B_3 to Σ_P, makes A_1, B_1, and A_2 definable under Σ_P. There are 2^n (where n is the

[3] It is worth noting that in the context of terminologies, B is a base symbol and should not be a named symbol in the ontology.

size of the not-primitively defined signature) choices of which symbols from the not-primitively defined signature to treat as primitive. A significant reason to consider treating some of the not-primitively defined concepts in the signature as though they were primitive is to minimise the symbols in the primitive signature, as the symbols in the primitive signature would need to be common to both communicating agents or aligned. Recall that the motive to investigate and use definition extraction in the context of agent communication is to reduce the number of symbols that have to be shared or aligned between agents. Let O be an ontology belonging to an agent AG_1 wishing to communicate with another agent AG_2, let Σ_P denote the primitive signature of O and let Σ_D denote the defined signature of O such that every symbol $S \in \Sigma_D$ has a definition C under O and $sig(C) \subseteq \Sigma_P$. With definition extraction techniques, agent AG_1 does not need to worry about aligning symbols in Σ_D because all symbols in Σ_D can be conveyed accurately using only symbols in Σ_P. Furthermore, as previously mentioned, definition extraction may help agents find hidden correspondences among symbols or help agents convey concepts that have no correspondences.

Let Σ_P denote the primitive signature and ${\Sigma_P}^N$ denote a not-primitively defined signature of an ontology O. We call the problem of finding a minimal subset ${\Sigma_P}^{N\prime}$ of ${\Sigma_P}^N$ such that all symbols in ${\Sigma_P}^N \setminus {\Sigma_P}^{N\prime}$ are definable under $\Sigma_P \cup {\Sigma_P}^{N\prime}$ the *primitive signature problem*. Alternatively, for this analysis, we simply disregard all concept names in not-primitively defined signature and only focus on the primitively defined signature when considering extracting definitions with respect to the primitive signature. This is additionally motivated by the fact that there is sufficient variability in the dataset to facilitate an evaluation and also the significant computational cost uniform interpolation included in all the algorithms developed in Chapter 5.

Related work

Geleta et al. [GPT17] investigate a similar problem, known as the *signature coverage problem* which is the problem of finding whether a given task signature $\mathfrak{S}$ can be *covered* by another restricted signature $\mathfrak{R}$, such that $\mathfrak{S}$ and $\mathfrak{R}$ are both subset signatures of the same ontology. Here by *covered by signature* $\mathfrak{R}$, we mean *"defined only using symbols in* $\mathfrak{R}$*"*. The signature coverage problem is investigated for ontologies of varying expressivity, and it has been shown (by Ten Cate et al. [TCFS13]) that the Beth definability property does not hold in DLs that allow for nominals or role hierarchies. Thus the definition of a *primitive* signature is subject to the expressivity of the ontologies investigated. The task signature is fixed (in the signature coverage problem).

However, our problem can be reduced by setting the task signature to the primitive signature of the ontology.

To find an approximate solution to the signature coverage problem, all *minimal definition signatures* of all the defined concepts in the ontology are used to incrementally build a *cover set* that covers all symbols in the task signature. For a defined symbol S in an ontology O, a *minimal definition signature*, is a signature denoted $MDS(S;O)$, such that $S \notin MDS(S;O)$ and, for any signature Σ such that $S \notin \Sigma$ for which S is definable under O with respect to Σ, we have that $|MDS(S;O)| < |\Sigma|$. A symbol may have multiple minimal definition signatures. Geleta [Gel18] has proposed algorithms and implementations for extracting approximates of minimal definition signatures from ontologies written in DLs as expressive as $\mathcal{SHROIQ}$.

Evaluation objectives

Driven by **RQ7**, for a given ontology in the dataset, we investigate the following points of interest:

1. How much of the primitive signature already exists in the naturally occurring definitions of explicitly defined concepts?

2. How much of the primitive signature already exists in strongest necessary conditions (extracted by Algorithm 5.1.1) of explicitly defined concepts with respect to the ontology signature?

3. How many defined concepts in an ontology are definable with respect to the entire signature of the ontology versus the primitive signature as defined in Definition 39?

4. What are the signature sizes associated with extracting a definition for an explicitly defined concept in terms of the primitive signature? In other words, to what *degree* are the signatures of definitions extracted with respect to the primitive signature larger or smaller than the signatures of the naturally occurring definitions of the explicitly defined concepts?

5. Does the strongest necessary condition (with respect to the primitive signature) of an explicitly defined concept name contain more superfluous concepts than the strongest necessary condition (with respect to the primitive signature) of its explicitly stated definition?

For Points 1 and 2, we simply reuse the results of Evaluation 1 in conjunction with the explicitly stated definitions in the ontology. For Point 3, we use the implicit definability test from Definition 4. For Points 4 and 5, we extract the SNC of the concept names with respect to the primitive signature and compare it to the extracted SNC (with respect to the primitive signature) of the corresponding explicitly stated definitions of the concept names.

Definability Report

Our first point of inquiry is the definability of concepts in ontologies contained in the corpus, specifically with respect to the primitive signature.

Report aim	Investigate the definability of defined concept names with respect to the primitive signature each respective ontology in the corpus.
Reported property	The size of Σ_D^P relative to Σ_D for each ontology.
Reported process	1. Determine Σ_D: for each concept name $A \in sig(O)$ if A is definable under $sig(O) \setminus \{A\}$, add A to Σ_D. 2. For each concept name $A \in \Sigma_D$ check if A is definable under Σ_P using the implicit definability test from Definition 4 (implemented using Hermit [GHM+14]).

We run another set of implicit definability tests for each ontology O in the corpus as follows:

1. $\Sigma_D = \emptyset$.
2. $\Sigma_P = \emptyset$.
3. **for** each concept name in $sig(O)$ **do:**
   (a) **if** A is implicitly definable w.r.t $sig(O) \setminus \{A\}$ **do:**
      i. $\Sigma_D \leftarrow \Sigma_D \cup \{A\}$.

   else:
      i. $\Sigma_P \leftarrow \Sigma_P \cup \{A\}$.
4. For each concept name $A \in \Sigma_D$ we check if A is definable under O with respect to the primitive signature Σ_P.

Ontology	Expl #	Impl #	Total	Def under Prim (Σ_D^P) #	Def under Prim (Σ_D^P) %
MHCRO	1254	0	1254	1253	**99%**
PDON	630	0	630	630	**100%**
TCO	558	0	558	558	**100%**
CHMO	102	0	102	102	**100%**
RXNO	71	0	71	71	**100%**
ACESO	67	0	67	21	**31%**
HPIO	13	23	36	0	**0%**
MOP	31	0	31	31	**100%**
BNO	0	14	14	0	**0%**
CCTOO	8	3	11	4	**36%**
ONTOPARON	7	0	7	7	**100%**
HP_O	5	0	5	0	**0%**
ISSVA	2	1	3	3	**100%**
DLORO	2	0	2	0	**0%**
GECKO	2	0	2	2	**100%**
MEDEON	0	1	1	1	**100%**
XEO	0	1	1	1	**100%**

Table 6.4: Entities definable under Σ_P. The column titled 'Expl #' stands for 'number of explicitly defined concepts'. The column titled 'Impl #' stands for 'number of implicitly defined concepts'. The column titled 'Def under Prim (Σ_D^P) #' stands for 'number of defined concept names definable under primitive signature'. The column titled 'Def under Prim (Σ_D^P) %' stands for 'percentage of defined concept names definable under primitive signature'.

The results are shown in Table 6.4. For most cases, we observe that 100% of the defined entities are also definable under the primitive signature. Notably, four of the ontologies, HPIO, BNO, HP_O, DLORO, do not have any concept names definable under the primitive signature. In these cases the reason that there are no concept names definable under the primitive signature is that most definitions only reference other definable concepts (similar to the ontology with the axioms $\{A \equiv B, B \sqsubseteq \neg C\}$). This is also the case with the ACESO ontology which has only thirty-one percent of its definable concepts definable with respect to the primitive signature, in which there is a significant number of atomic equivalences of the form $A \equiv B$ that account for this. However, in some cases, such as that of *CCTOO*, we observe that the reason some of the definable concept names are not definable with respect to the primitive signature is that there are cyclic concept names.

Naturally Occurring Primitive Signature

Our next point of inquiry is to investigate how much of the primitive signature already exists in naturally occurring definitions and SNCs. The results from Section 6.2 demonstrate the tendency for definitions extracted using SNCs to contain superfluous information. Depending on whether the signature of the superfluous information is also defined, this also has an impact on the superfluous information that may be contained in the definition of a concept with respect to the primitive signature of the ontology. Example 6.3.1 illustrate this effect.

Report aim	Investigate and compare the primitive signature in explicitly stated definitions versus that in strongest necessary conditions extracted using Algorithm 5.1.1
Observed property	The primitive signature that exists in explicitly stated definitions occuring in equivalence class axioms of defined concept names and strongest necessary conditions.

Example 6.3.1. Consider the ontology $\{A_1 \equiv \exists r.C, A_1 \sqsubseteq A_2, A_2 \equiv \exists p.B\}$, the primitive signature is $\{p, r, B, C\}$. A valid extraction of A_1 with respect to the primitive signature is $\exists r.C$; however, when Algorithm 5.1.1 is used to extract a strongest necessary condition, we get the definition $\exists p.B \sqcap \exists r.C$ which is also a definition of A_1 but with the superfluous signature $\{p, B\}$ (because the expression $\exists p.B$ is superfluous).

For a given ontology with explicitly defined concept names, the pipeline was as follows:

1. **for** every satisfiable explicitly defined concept $A \in O$ **do:**
   (a) Extract any equivalent class axioms $A \equiv C$ for A
   (b) Take the conjunction of every definition C as the definition φ of A.
   (c) Extract a strongest necessary condition of A with respect to $sig(O) \setminus A$ under O, i.e., $SNC(A; O; sig(O) \setminus \{A\})$ using Algorithm 5.1.1.
   (d) Compare the primitive signatures occuring in φ and $SNC(A; O; sig(O) \setminus \{A\})$.

In this inquiry, we focus only on concept names that have explicitly stated definitions, as we want to use explicitly stated definitions as a point of reference for evaluating the signatures of the SNCs. We want to compare the percentage of the

signature of explicitly stated definitions that are primitive with the percentage of the signature of SNCs that are primitive prior to extracting these definitions to see if there is a significant difference. For concept names that have more than one explicitly stated definition, we consider all explicitly stated definitions listed. Note that here, although the notion of minimal definition signatures may apply, there are strict conditions on the signatures used (i.e., the naive partitioning of the ontology), which make it difficult to establish a notion of minimality. For example, consider the ontology $\{A \equiv B \sqcup C, B \sqsubseteq \neg C, A \equiv \exists p.(D \sqcap E)\}$. We may consider $\{B, C\}$ to be a minimal definition signature for A, but we also have that B and C are definable, and thus would not be considered as part of the primitive signature. Since explicitly stated definitions are intentionally written by the ontology engineer, we consider these definitions the *ideal* definitions. The charts in Figures 6.3 and 6.4 show the percentage of SNC signature (with respect to the entire ontology signature) and definition signatures that are primitive for ontologies that contained more than fifty defined concept names. For ontologies that contained less than fifty defined concept names, the results are shown in Table 6.5. As can be observed from Figure 6.3, Table 6.5, most of the definitions of the explicitly defined concepts do not consist solely of the primitive signature.

From Figures 6.3 and 6.4, we can observe that there is a significant variation between the ratio of the SNC signature that is primitive and that of the explicitly stated definitions for some of the ontologies (specifically CHMO, RXNO, and MHCRO). This variation can be mainly attributed to superfluous information extracted by the SNC algorithm. Furthermore, it is worth noting that a significant number of concepts have explicitly stated definitions that are entirely within the primitive signature for ontologies such as (CHMO, RXNO, ACESO, MHCRO), which suggests that for these ontologies, it is more likely for the extracted SNCs to have superfluous information than the extracted definitions (as illustrated in Example 6.3.1).

Similarly, in Table 6.5, we can see that most ontologies contain explicitly stated definitions that have very little of their signatures completely within the primitive signature. Furthermore, for most of the ontologies (with the exception of the MOP ontology), less than fifty percent of the signature of the explicitly stated definitions is primitive for most defined concepts, i.e., most of the explicitly stated definitions reference defined concepts. The pattern is similar for the extracted SNCs.

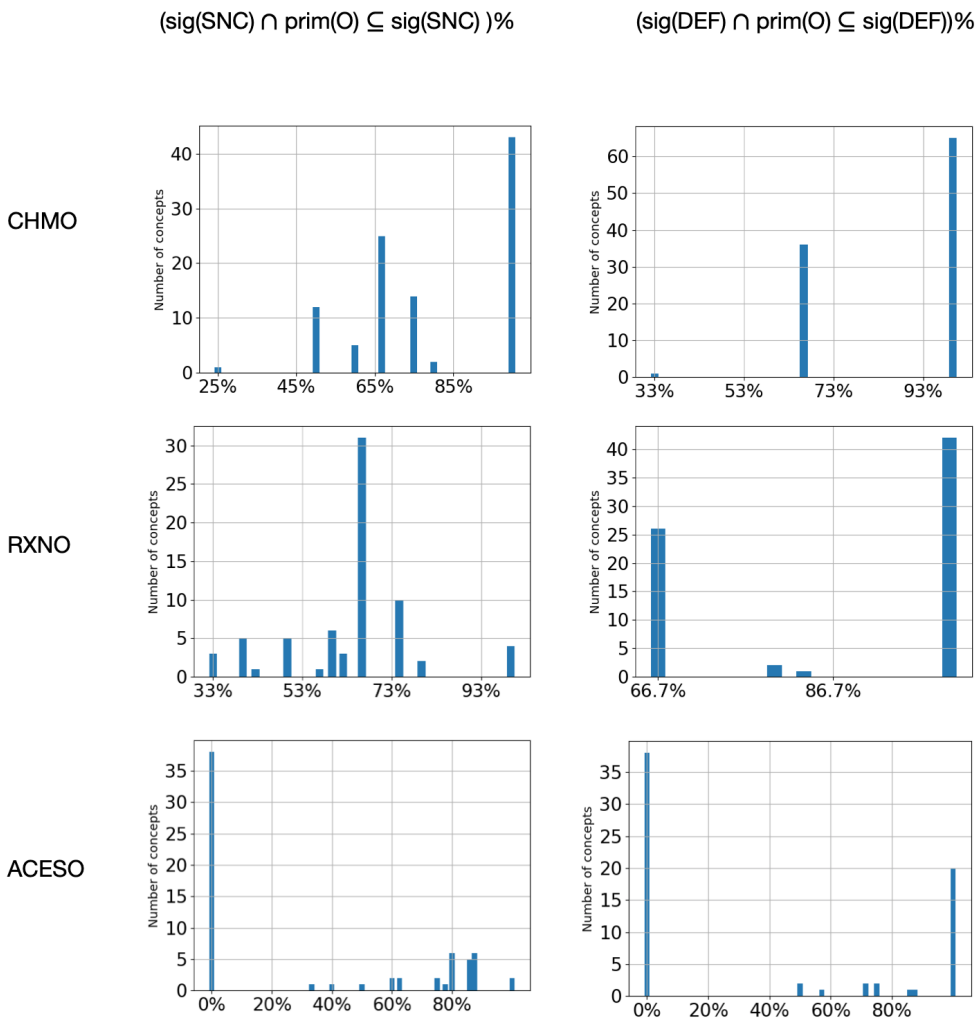

Figure 6.3: Percentage of signature that is primitive. The first column labelled "$(sig(SNC) \cap prim(O)) \subseteq sig(SNC))\%$ indicates that charts in this column indicate the percentage of the SNC signature that was primitive. The second column labelled "$(sig(DEF) \cap prim(O)) \subseteq sig(DEF))\%$ indicates that charts in this column indicate the number of explicitly defined concepts and the percentage of their explicitly stated definition signature that was primitive (where 'DEF' means 'explicitly stated definition'. The horizontal axis (x-axis) indicates the percentage of the signature of the strongest necessary condition (definition) that is primitive. The vertical axis (y-axis) is the number of concepts. For example, the second bar in the RXNO ontology for the first column can be read "for five concepts in the RXNO ontology 43% of their SNC signature was primitive".

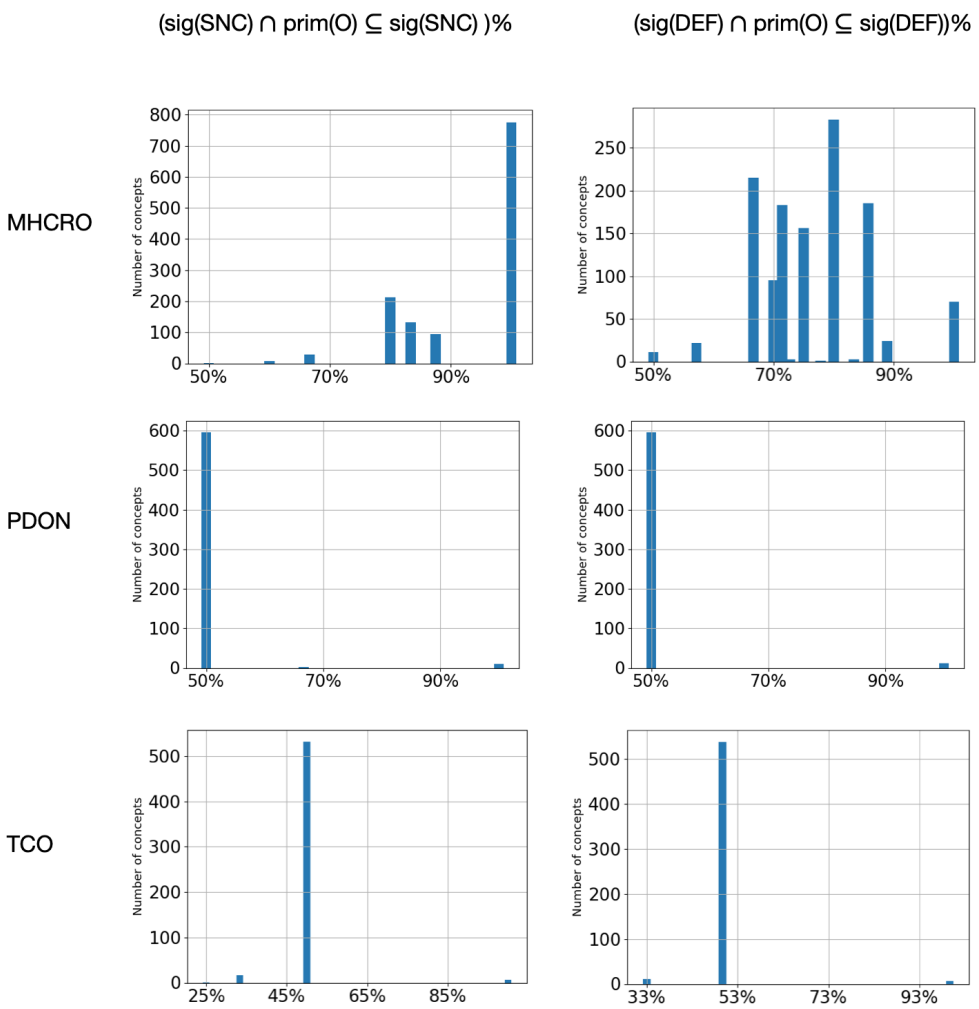

Figure 6.4: Percentage of signature that is primitive set 2.

Ontology	Defined #	>50% prim (def)	>50% prim (SNC)	100% prim (def)	100% prim (SNC)
MOP	31	31	31	**9**	**9**
BNO	14	-	-	-	-
ONTOMA	13	3	3	**0**	**0**
CCTOO	11	1	1	**0**	**0**
ONTOPARON_SOCIAL	7	7	7	**7**	**7**
HP_O	5	0	0	**0**	**0**
ISSVA	3	0	0	**0**	**0**
DLORO	2	0	0	**0**	**0**
GECKO	2	2	2	**2**	**0**
EO	1	-	-	-	-
MEDEON	1	1	1	**0**	**0**
XEO	1	1	1	**1**	**1**

Table 6.5: Percentage of definition and SNC signature that is primitive. The Column titled '> 50% prim (def)' indicates the number of explicitly defined concepts whose explicitly stated definitions had more than 50% of its signature within the primitive signature. The Column titled '50% prim (SNC)' indicates the number of explicitly defined concepts whose strongest necessary conditions extracted using Algorithm 5.1.1 had more than 50% of its signature within the primitive signature. The Column titled '100% prim (def)' indicates the number of explicitly defined concepts whose explicitly stated definitions had an entirely primitive signature. The Column titled '100% prim (def)' indicates the number of explicitly defined concepts whose strongest necessary conditions extracted using Algorithm 5.1.1 had a signature that was entirely primitive.

Extraction Sizes

In our final point of inquiry, we extract an SNC and the explicitly stated definitions of explicitly defined concepts with respect to the primitive signatures of the ontologies in the corpus. Let A be an explicitly defined concept name and ϕ an explicitly stated definition of A; we investigate the difference in signature sizes between $SNC(A;O;\Sigma_P)$ and $SNC(\phi;O;\Sigma_P)$ where Σ_P is the primitive signature of the ontology O of interest. The motivation is illustrated in Example 6.3.1, which suggests that $SNC(\phi;O;\Sigma_P)$ may contain less superfluous concepts than $SNC(A;O;\Sigma_P)$; the aim of this report is to investigate how often this occurs in the corpus. We only report results on successful extractions. An extraction is deemed successful if the resulting concept expression is equivalent to the defined concept with respect to the ontology and contains no fixpoint operators or definer symbols in its signature. The experiment pipeline was as follows (with a three hour timeout [4]):

1. **for** every explicitly defined concept $A \in sig(O)$ **do:**
   (a) Extract an explicitly stated definition φ of A from O with the smallest signature size.
   (b) Generate a fresh individual i and add $\varphi(i)$ to O.
   (c) Generate a fresh individual i' and add $A(i')$ to O.
2. Extract the uniform interpolant $\mathcal{V}$ of O with respect to the primitive signature.
3. Let C denote the conjunction of every concept D such that $D(i) \in \mathcal{V}$, we have that C is a strongest necessary condition of the definition φ of the corresponding explicitly defined concept.
4. Let C' denote the conjunction of every concept D' such that $D'(i') \in \mathcal{V}$, we have that C' is a strongest necessary condition of the corresponding explicitly defined concept A with respect to the primitive signature.
5. Compare the signature sizes of C and C':
   (a) Compute the ratio of $sig(C')$ that intersects with $sig(C)$.
   (b) Compute the ratio of $sig(C)$ that intersects with $sig(C')$.

[4] This is because we observed that letting LETHE run for any longer yields minimal to no advantages.

Figure 6.5 illustrates the instances in which the signature size of the extracted SNC is greater than, equal to, or less than the signature size of the extracted definition. As can be seen, in the majority of the cases, we have either the signature sizes are equal or that of the extracted SNC was larger. There is a significant number of instances for which the extracted signature size of the SNC is greater than that of the definition. Table 6.6 illustrates the actual signature sizes in terms of quartiles; again here we can observe the magnitude of signature difference between the extracted definitions and the extracted SNC, the difference is significant in most ontologies apart from a few (such as ONTOPARON_SOCIAL, GECKO, and CHMO). Table 6.8 shows the percentage increase in signature size for the extracted definitions. The percentage increase for most cases is quite significant and indicates that extracting definitions with respect to the primitive signature comes with a significant cost in the expression of the extracted definition in terms of symbols used. As expected, the results indicate that extracting the SNC with respect to the primitive signature will contain superfluous information.

Results are shown for successful extractions only both on the concept level and ontology level.

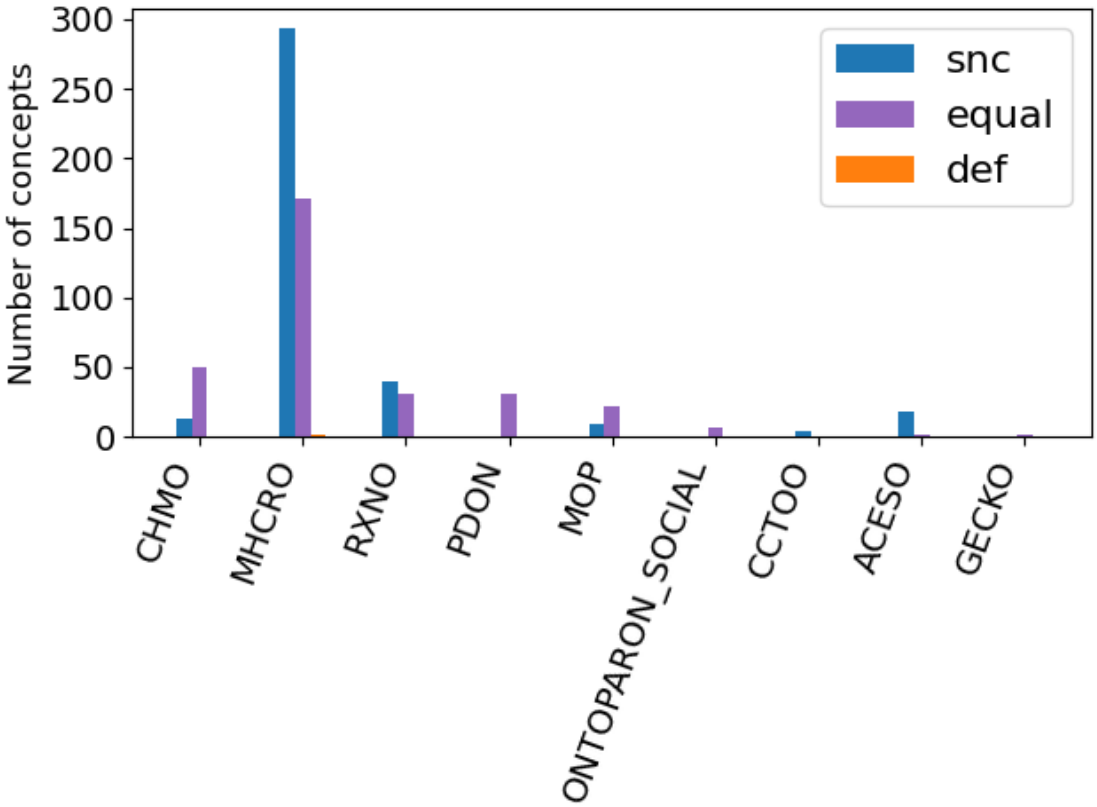

Figure 6.5: Compared signature sizes of extracted definitions. The purple bar indicates the number of concepts for which the signature size of the extracted concept name was equal to the signature size of the extracted definition. The blue bar indicates the number of concepts for which the signature size of the extracted concept name was greater than the signature size of the extracted definition. The orange bar indicates the number of concepts for which the signature size of the extracted definition name was greater than the signature size of the extracted concept name.

Ontology	Q1		Q2		Q3		Q4	
	SNC	DEF	SNC	DEF	SNC	DEF	SNC	DEF
CHMO	3.0	3.0	3.0	3.0	3.0 4.0	3.0	3.0	3.0
MHCRO	4.25	3.0	6.0	6.0	11.0	6.0	3.0	3.0
RXNO	28.0	3.0	34.0	3.0	37.0	28.0	3.0	2.0
PDON	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
MOP	4.0	3.0	17.0	4.0	17.0	17.0	3.0	3.0
ONTOPARON	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
CCTOO	5.75	2.0	6.5	2.0	7.25	2.5	5.3	2.0
ACESO	5.0	4.0	7.0	6.0	8.0	7.0	4.8	3.8
GECKO	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0

Table 6.6: Quantiles of signature sizes. The column titled 'Q1' denotes the first quantile, the column titled 'Q2' denotes the second quantile, while the column titled 'Q3' denotes the third quantile. The column titled 'SNC' denotes the quantile of signature size of the SNC extracted with respect to the primitive signature for the corresponding ontology. The column titled 'DEF' denotes the quantile of signature size of the SNC of the explicitly stated definition with respect to the primitive signature for the corresponding ontology.

Ontology	Time
MHCRO	3 hrs
PDON	3 hrs
CHMO	3 hrs
RXNO	20.5 mins
ACESO	3 hrs
MOP	28.8 secs
ONTOPARON_SOCIAL	4 secs
GECKO	1.7 secs

Table 6.7: Interpolation times

Ontology	Surface Size	Extracted Size	Increase %
GECKO	2.0	3.0	50.0
MOP	3.1	12.9	321.1
ONTOPARON	3.0	3.0	0.0
PDON	2.0	2.2	11.4
RXNO	2.9	28.7	881.2
CHMO	3.0	3.8	25.9
CCTOO	2.5	6.5	160.0
ACESO	2.9	6.3	117.0
MHCRO	5.9	8.4	42.8

Table 6.8: Percentage increase in signature size. The column titled 'Surface Size' indicates the average size of the definitions in the ontology with respect to the entire ontology signature. The column titled 'Extracted Size' indicates the average size of the explicit definitions extracted with respect to the primitive signature. The column titled 'Increase %' indicates the percentage increase.

6.4 Evaluation of Compiled Super Concept Extraction

As discussed in Section 5.2.1, assuming all symbols in the common signature are not interpreted the same way in the respective ontologies of the communicating agents, strongest necessary conditions and weakest sufficient conditions may not convey enough information about the concept intended to be shared. We discussed and implemented an algorithm (Algorithm 5.2.2) for extracting an approximation of compiled super concepts in Section 5.2.1. A shortcoming of Algorithm 5.2.2 is that it only extracts an approximation of the compiled superconcept. The approximate nature stems from the fact that there can be scenarios in which superfluous information may be included in the compiled superconcept (compiled subconcept). This is especially the case because Algorithm 5.2.2 is saturation-based.

Thus, an imperative point of concern (and focal point of examination in this section) is the feasibility of Algorithm 5.2.2 in terms of computation and the succinctness of its output. We evaluate the information returned by Algorithm 5.2.2 for extracting compiled superconcepts over some of the ontologies in the corpus. Due to the high computational cost of Algorithm 5.2.2, and, furthermore, the unavailability of an efficient role forgetting method, we were unable to perform the experiments on all the ontologies in the corpus.

As we have no means of validating whether a compiled superconcept has redundancies in its signature, we use syntactic module extraction as a baseline for comparison and evaluation.

Our motivation to utilise locality-based modules to evaluate the superfluous information that may be contained in compiled superconcepts extracted by Algorithm 5.2.2 is that syntactic bottom modules are analogous to compiled superconcepts in that they preserve axioms that entail all necessary conditions of a concept[5]. Because bottom modules are subsumer preserving, self-contained, and subsets of the original ontology, we expect that the signature of a compiled superconcept to be smaller than or equal to that of the bottom module for the seed signature corresponding with the signature of the concept. If the signature of the bottom module is smaller than the signature of the compiled superconcept, then this is an indication that the compiled superconcept contains some superfluous information. The aim of our evaluation in this section is thus to observe how frequently we can determine superfluous information in the compiled

[5]the same analogous nature applies to syntactic top modules and compiled subconcepts.

superconcepts extracted by Algorithm 5.2.2 when applied to ontologies in the experiment corpus. To simplify things, we extract compiled superconcepts with respect to the entire signature of the ontology (minus the concept name of interest) and compare it against the syntactic bottom module for the singleton set consisting of the concept name of interest. Given an ontology O, our experiment pipeline is as follows:

1. **for** each concept name A in O, **do:**
   (a) Extract $CSP(A; O; sig(O) \setminus \{A\})$.
   (b) Extract the bottom module for the seed signature $\{A\}$ denoted $\mathcal{M}_A^O$.
   (c) Compare and analyse the resulting signatures of $CSP(A; O; sig(O) \setminus \{A\})$ and $\mathcal{M}_A^O$.

The results of the experiment are presented in Figure 6.6, Figure 6.7, Table 6.9, and Table 6.10.

Figure 6.7 illustrates the successful extractions of both methods. For Algorithm 5.2.2 (which extracts the CSPs in the evaluation) an extraction is deemed successful if there is no timeout or exception thrown. For module extraction, an extraction is deemed successful if there is no timeout. Both algorithms had a timeout of five minutes per concept name to maintain feasibility of the process as most ontologies in the dataset had well over 100 concept names. The ontologies iterated over were restricted to {EO, EPIE, EPIP, XEO, ISSVA, HP_O, ONTOMA, HPIO, MEDEON, XEO, RXNO} primarily because Algorithm 5.2.2 timed out for most concepts in the remaining ontologies in the corpus. From Figure 6.7 we observe that Algorithm 5.2.2 failed to extract any CSPs for any concepts in the HPIO ontology, and only three concepts in the EO ontology. With the exception of the HP_O and RXNO ontologies, we observe an equal success rate for the rest of the ontologies.

Figure 6.6 displays the signature comparisons of the modules extracted and compiled superconcepts in which both extractions were successful; for each ontology, the blue bar illustrates the number of concepts for which the signature size of the compiled superconcept extracted by Algorithm 5.2.2 was smaller than the signature size of the bottom module of the same concept. Similarly, the green bar illustrates the number of concepts for which the signature size of the bottom module was smaller than the signature size of the compiled superconcept of the same concept extracted by Algorithm 5.2.2. The purple bar illustrates cases in which the signature sizes were equal.

Figure 6.6 highlights the fact that there are very few cases in which the compiled superconcept contains a smaller signature than the bottom module; therefore, it is clear

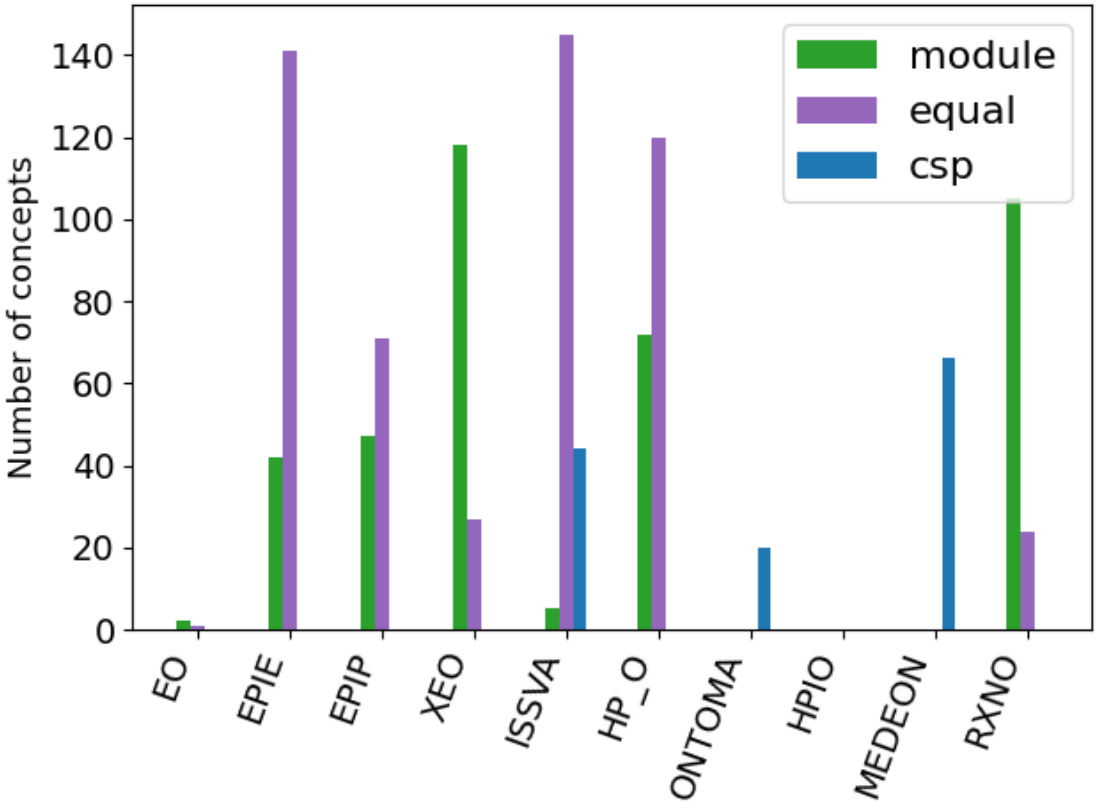

Figure 6.6: Signature comparisons of Bottom Module and Compiled superconcepts. The green bar illustrates the number of concepts for which the signature of the bottom module was smaller than the signature of the compiled super concept. The blue bar illustrates the number of concepts for which the signature of the compiled super concepts was smaller than the signature of the bottom module. The purple bar illustrates the number for which the signature of the bottom module was equal to the signature of the compiled super concept.

that the compiled superconcepts extracted using Algorithm 5.2.2 often contain superfluous information. The magnitude of the difference in the signature sizes of the extracted compiled superconcepts and bottom modules are illustrated in Table 6.9. With the exception of a few cases such as the RXNO, EPIP, and EO ontologies, we observe that the difference in signature sizes is almost always smaller than 10 symbols (most cases were around 2 or 3 symbols).

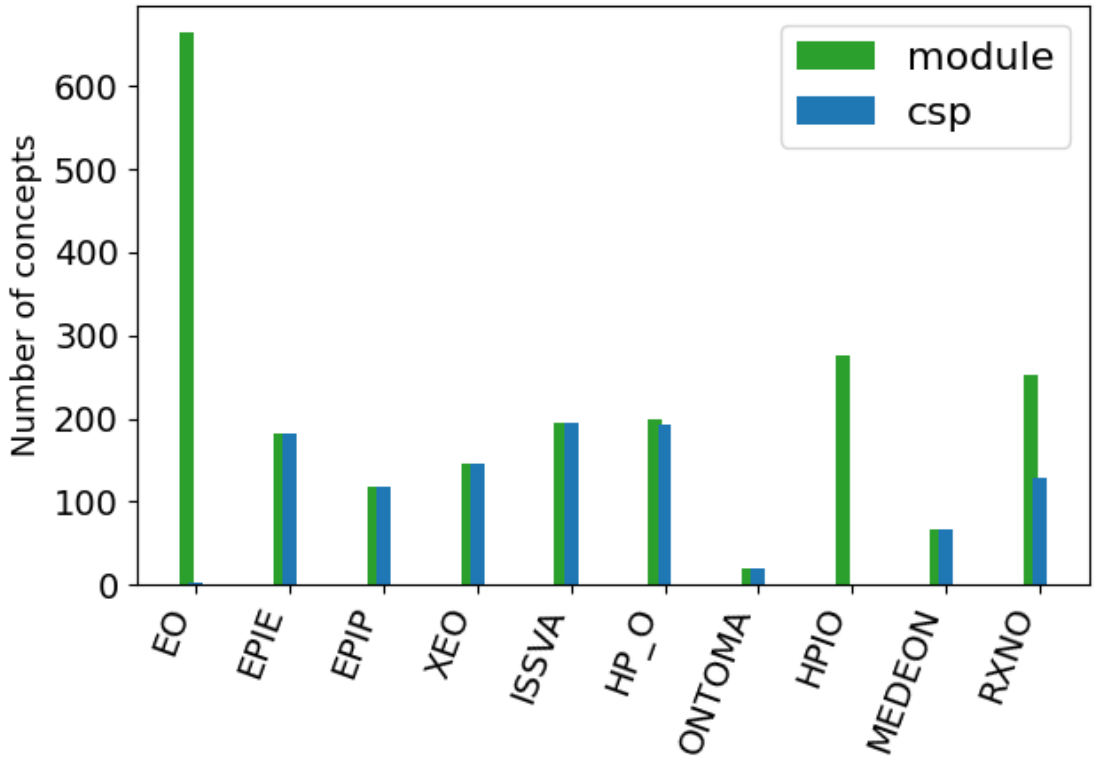

Figure 6.7: Successful Extractions.

Ontology	Q1		Q2		Q3		Q4	
	MOD	CSP	MOD	CSP	MOD	CSP	MOD	CSP
EO	7.0	6.0	7.0	11.0	8.0	164.5	10.0	318.0
EPIE	3.0	3.0	4.0	4.0	6.5	7.5	8.0	17.0
EPIP	2.0	2.0	3.0	3.0	4.0	39.0	5.0	41.0
XEO	13.0	14.0	16.0	17.0	18.0	19.0	18.0	19.0
ISSVA	3.0	3.0	9.0	7.0	12.0	10.0	14.0	14.0
HP_O	5.0	5.0	6.0	9.0	12.5	17.0	36.0	29.0
ONTOMA	42.75	2.0	43.0	2.0	44.0	3.0	45.0	4.0
HPIO	37.0	-	38.0	-	39.0	-	43.0	-
MEDEON	19.0	2.0	19.0	2.0	20.0	2.0	21.0	8.0
XEO	13.0	14.0	16.0	17.0	18.0	19.0	18.0	19.0
RXNO	3.0	12.0	4.0	12.0	6.0	13.0	15.0	16.0

Table 6.9: Quantiles of signature sizes. The column titled 'Q1' denotes the first quantile, the column titled 'Q2' denotes the second quantile, while the column titled 'Q3' denotes the third quantile. The column titled 'MOD' denotes the quantile of signature size of the bottom module for the corresponding ontology. The column titled 'CSP' denotes the quantile of signature size of the extracted compiled superconcept for the corresponding ontology.

Lastly, Table 6.10 illustrates the instances in which the signatures of the compiled superconcepts had symbols beyond the signatures of the bottom module, specifically for cases where the signature of the compiled superconcepts was smaller than the signature of the bottom modules i.e., $|sig(CSP(A;O;sig(O) \setminus \{A\})| \leq |sig(\mathcal{M}_A^O))|$ and $sig(CSP(A;O;sig(O) \setminus \{A\})) \not\subseteq sig(\mathcal{M}_A^O))$. We omit the case where $|sig(CSP(A;O;sig(O) \setminus \{A\})| > |sig(\mathcal{M}_A^O))|$ as in such cases, it follows that $sig(CSP(A;O;sig(O) \setminus \{A\})) \not\subseteq sig(\mathcal{M}_A^O))$.

	Contained		Total Count
Ontology	Count	percentage	
EO	1	**33.3%**	3
EPIE	140	**76.5%**	182
EPIP	70	**59.3%**	117
XEO	27	**18.6%**	145
ISSVA	188	**96.9%**	193
HP_O	120	**62.5%**	192
ONTOMA	20	**100.0%**	20
MEDEON	66	**100.0%**	66
RXNO	24	**18.6%**	129

Table 6.10: Instances in which the signature of the CSP had symbols outside of the signature of the bottom module. The *Contained* column indicates cases in which the signature size of the CSP was contained in the signature of the bottom module. The *Total* column indicates all cases.

Table 6.10 thus illustrates that even in cases where the compiled superconcept had a signature smaller than the extracted bottom module, there was possibly some superfluous information contained in the compiled superconcept.

The main source of complexity for Algorithm 5.2.2 is that it is based on saturation. It is worth noting that in our attempt to construct an algorithm to extract compiled superconcepts, several alternatives to saturation were explored, which include *unfolding* [BCM$^+$03], consequence-based reasoning [SKH11], brute force (extracting a strongest necessary condition of the concept of interest for every subset of the powerset of the signature of interest). In each case, we found that no advantage in terms of computation time existed over using saturation or that saturation would have had to be implemented in some way.

6.5 Contribution Summary

The contributions of this chapter are as follows:

- An empirical analysis of Algorithm 5.1.1 for extracting definitions from ontologies for concept names. Our results demonstrate that Algorithm 5.1.1 is able to extract definitions for defined concept names in most ontologies. Our results also demonstrate that there are often superfluous concepts in the definitions extracted by Algorithm 5.1.1.

- A characterisation of *primitive signatures* in the context of general TBoxes.

- An empirical analysis of definability and definitions in the context of primitive signatures. Our results demonstrate that in general TBoxes, it is often not the case that all implicitly or explicitly definable concept names are definable with respect to the primitive signature of the TBox.

- An empirical analysis of Algorithm 5.1.1 for extracting definitions in the context of and with respect to primitive signatures.

- An empirical analysis of Algorithm 5.2.2 for extracting compiled superconcepts (whilst using module-extraction as a benchmark). The results demonstrate that Algorithm 5.2.2 performs poorly even in small ontologies. Henceforth, we will only evaluate compiled superconcepts and compiled subconcepts from a theoretical perspective.

Chapter 7

Adoption within Agent Communication

In Chapter 6 we evaluated the knowledge extraction techniques that were developed in Chapter 5. All the algorithms developed and discussed in Chapter 5 are based on uniform interpolation. The implementation of uniform interpolation used was saturation based and thus all the algorithms developed in Chapter 5 intrinsically inherit disadvantages that come with saturation based methods such as expensive computation costs and inclusion of superfluous terms in the results. In Section 6.2 we empirically investigated the extent to which redundant terms are prevalent in definitions extracted using strongest necessary conditions extracted by Algorithm 5.1.1. In Section 6.3 we empirically investigated definability and definition extraction whilst taking primitive signatures (Definition 39) into consideration. In Section 6.4, we empirically evaluated the size of compiled superconcepts extracted using Algorithm 5.2.2.

The motivation to investigate the extraction of definitions in the first two sections of Chapter 6 was to evaluate Theorem 6 based on the assumption that given a restricted signature Σ possibly representing the common vocabulary between communicating agents, symbols in Σ had the same interpretation in the respective ontologies of the agents; in such cases, the definition α of a concept A with respect to Σ, such that $A \notin \Sigma$ forms an exact representation of A that can be conveyed to a participating communicating agent. The evaluation in Section 6.2 demonstrated that Algorithm 5.1.1 was successful in extracting definitions, but a significant amount of the definitions extracted had a significant amount of superfluous terms. Some of the results in Section 6.2 highlight the fact that using a reasoner such as Hermit [GHM$^+$14] to extract atomic

definitions may be preferable to Algorithm 5.1.1 for definition extraction tasks (specifically in cases where there are equivalent atomic concepts in the common signature for the concept of interest).

Although the knowledge extraction tools developed in Chapter 5 are specifically designed for the purposes of agent communication, we have not yet addressed how these tools can be adopted and implemented in any existing agent framework or even proposed guidelines on how they can be implemented or adopted; addressing this gap is the focal point of discussion and investigation of this chapter.

Thus, this chapter tries to answer **RQ4** "*How can the relevant knowledge extraction techniques be integrated into existing MAS frameworks or agent communication protocols?*". Specifically, we discuss and explore how strongest necessary conditions and weakest sufficient conditions can be adopted in the communication protocols proposed for the ANEMONE framework [vDBD+06] (Section 7.1). In Chapter 3, we discussed relevant agent communication frameworks that have some ontology-based components as part of their architecture. Of all the frameworks discussed, we find that the ANEMONE system is the most appropriate agent framework in which to evaluate the knowledge extraction tools developed in this thesis, primarily because the ANEMONE framework is designed specifically for the agent communication scenario that this thesis is concerned with: finding how an agent can convey an uncommon term with respect to its common vocabulary. Compared to other frameworks, ANEMONE places a high emphasis on communication of knowledge in ontologies, while other frameworks tend to focus on using ontologies to improve other aspects of agent-based systems such as planning and scheduling. Furthermore, other agent frameworks such as JASON and JADL & JIAC make the assumption that all knowledge is understood by the communicating agents and sometimes include architectural components to ensure that this assumption holds.

The communication protocols in the ANEMONE system are designed for a propositional form of description logics that does not allow for more expressive constructs such as value restrictions and disjunctions. However, the implicit shortcoming here is that the application of ANEMONE is limited in cases where agents have ontologies with concept descriptions that include allow for such constructs. We specify, discuss, and evaluate how the adoption of strongest necessary conditions and weakest sufficient conditions may enhance the limited application of ANEMONE to agents with ontologies as expressive as $\mathcal{ALC}$. Our results suggest that in most cases, the expressivity of the ontology and the existence of more expressive constructs in the ontology do

not significantly affect the performance of using a reasoner to extract descriptions of concepts, in fact, in most cases we find that using reasoners is equally as efficient as strongest necessary conditions and weakest sufficient conditions in conveying uncommon concepts.

7.1 ANEMONE

ANEMONE [vDBD+06] is a framework consisting of dynamic communication protocols that agents may use to convey concepts to each other depending on varying levels of their common vocabulary. The purpose of communication in ANEMONE is mainly to share assertional knowledge. Intuitively speaking, an agent AG_1 conveys a concept C to an agent AG_2 so that AG_2 can share instances of C under its knowledge.

The description logic of ANEMONE (which we denote $DL_{ANEMONE}$) is different from that of OWL ontologies. First of all, $DL_{ANEMONE}$ only defines semantics over a set of concept names N_C and a set of individuals N_I (i.e., no role symbols can exist in ANEMONE ontologies). Concepts only consist of concept names (i.e., no disjunction, negation, or roles) and relations among concepts are restricted to the constructs in Table 7.1.

Relation	Description	Syntax	Semantics
Subset	*transitive*, *anti-symmetric*	$C \sqsubset D$	$C^I \subset D^I$
Superset	*transitive*, *anti-symmetric*	$C \sqsupset D$	$D^I \subset C^I$
Equivalence	*transitive*, *symmetric*, *reflexive*	$C \equiv D$	$C^I = D^I$
Overlap	*symmetric*	$C \oplus D$	$C^I \not\subseteq D^I, D^I \not\subseteq C^I$ and $C^I \cap D^I \neq \emptyset$
Disjoint	*symmetric*	$C \perp D$	$C^I \cap D^I = \emptyset$

Table 7.1: ANEMOME description logic language constructs and semantics.

We find that the closest adaptation of the ontology semantics in ANEMONE to OWL semantics [AvH04] is $\mathcal{AL}$ without concept intersection, negation, universal restrictions, and limited existential quantification. We will henceforth assume the ANEMONE

description logic to be $\mathcal{AL}$ without concept intersection, negation, universal restrictions, and limited existential quantification.

It is worth noting that the $\oplus$ operator is inconsequential for retrieving and sharing instances of individual concept names; for example, adding $C \oplus D$ to the ontology $\{C(a), C(c), D(b), D(a)\}$ does not affect the query for instances of C.

ANEMONE has three design objectives: (i) Minimal and effective communication, (ii) Laziness, (iii) Decentralised communication. The minimal and effective objective is designed to ensure that communicated knowledge can be processed in a timely fashion. The laziness objective is designed to ensure that knowledge sharing only occurs on a *as-need* basis, i.e., knowledge should only be exchanged when strictly necessary. The decentralised objective is designed to ensure that there is no central control or location of the knowledge.

The laziness objective of ANEMONE is realised through the use of communication modes illustrated in Figure 7.1. There are three modes of communication: (i) normal communication protocol (ii) concept definition protocol (iii) concept explication protocol. All conversations start by assuming that there are no misunderstandings and occur in the normal communication protocol which is the uppermost layer. If any misunderstanding occurs, the agents switch to the concept definition protocol where they may attempt to resolve the misunderstandings by exchanging descriptions[1] of concepts that may have caused the misunderstanding (ideally in terms of concepts that are shared by both agents). If misunderstandings persist in the concept definition protocol, the agents switch to the concept explication protocol where agents convey the meaning of a concept by exchanging positive and negative examples of the misunderstood concept. The communication protocols help realise the *as-need* requirement of the laziness objective: agents only resort to using complex communication mechanisms when needed.

ANEMONE specifies two performatives to be used in communication, the $Inform$ and $ExactInform$ performatives. Let X be a concept an agent AG_1 is attempting to convey its communication partner AG_2. If X or an equivalent concept C is communicated to AG_2, the $ExactInform$ performative is used; however, if some other concept that is not equivalent to X is conveyed AG_2, then the $Inform$ performative is used. The use of these performatives dictates whether the conversation should switch from the normal communication protocol to the concept definition protocol: the use of the $Inform$ performative hints at some loss in communication and often suggests that the

[1] In [vDBD+06], the terms *define* and *definitions* are used instead; however, we use *describe* and *descriptions* to avoid confusion with the notion of definitions in $\mathcal{ALC}$ ontologies.

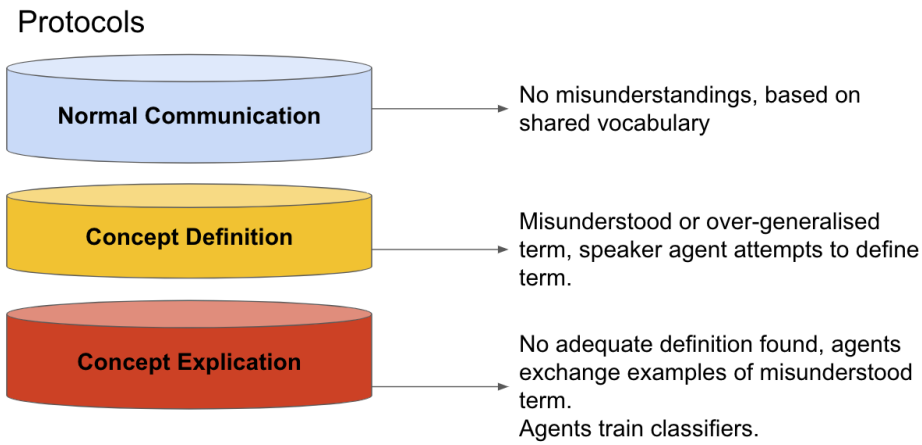

Figure 7.1: ANEMONE protocols.

agents have to switch to the concept definition protocol. In the concept definition protocol, agents exchange a finite set of axioms (which we refer to in this chapter as a *concept description*) that *describes*[2] an uncommon concept. When agents communicate in the concept definition protocol (attempting to resolve the meaning of a concept X), they switch to the concept explication protocol if the concept description extracted for X is perceived as inadequate by AG_2. A concept description is deemed inadequate if AG_2 can not infer the relation of the described and uncommon concept with every other concept in their ontology. Let O' denote a concept description recieved by AG_2 for an uncommon concept name A, if there is a concept name A' in $sig(O_2)$ (O_2 denotes AG_2's ontology) such that A can not be related to A' (under to $O' \cup O_2$) via any of the relations in Table 7.1, then O' is deemed inadequate. Intuitively speaking, the notion of adequacy in ANEMONE is motivated by expecting the hearer agent to be able to place an uncommon concept in an exact point in the class hierarchy of its ontology; however, the notion of adequacy is too strong for $\mathcal{ALC}$ ontologies as $\mathcal{ALC}$ is more expressive.

The concept explication protocol is underpinned by ensuring the ontologies of the communicating agents are *grounded*, meaning that the domain of discourse contains all the objects the agent may wish to speak about, e.g. the set of URLs on the internet. The use of grounded ontologies enables agents to realise the intended interpretation of concepts that are used in communication. Grounded knowledge bases may contain *classifiers* (intended to be realised using machine learning techniques) in addition to

[2]In [vDBD+06], the terms *define* and *definitions* are used instead; however, we use *describe* and *concept descriptions* to avoid confusion with the notion of definitions in $\mathcal{ALC}$ ontologies.

symbolic descriptions to ensure the agents can classify objects in the domain of discourse. In a more traditional sense, these classifiers can be considered as the sensors of the agent.

Terminological negotiation [SMVM16] and related approaches to concept explication can be adopted to realise the concept explication protocol, as such we focus our efforts on the Normal Communication and concept definition protocols which can benefit from known knowledge extraction techniques in DL.

7.2 Extracting Concept Descriptions in the ANEMONE system

The ANEMONE mechanism for conveying uncommon concepts aims to minimise loss during communication. In the normal communication protocol, agents must communicate only using common concept-names; if an agent wants to convey an uncommon concept name X, it has to find an equivalent common concept name X' and convey X' in place of X (thus ensuring that there is no loss in conveying X). If no such X' exists, then the agent must switch to the concept definition protocol where it attempts to extract a maximally precise description O of X and convey O. ANEMONE specifies that description for a concept X is a set of axioms relating X via subsumption $\sqsubset$, disjointness, and equivalence $\equiv$ to any concept names within the common vocabulary Σ.

Let X be concept name, Σ a signature and O an $\mathcal{AL}$ ontology without concept intersection, negation, universal restrictions, and limited existential quantification, such that $\Sigma \subseteq sig(O)$, $X \in sig(O)$, and $X \notin \Sigma$. A description for X following [vDBD+06] is defined as $\{X \sqsubset A \mid O \models X \sqsubset A \wedge A \in \Sigma\} \cup \{A \sqsubset X \mid O \models A \sqsubset X \wedge A \in \Sigma\} \cup \{A \equiv X \mid O \models A \equiv X \wedge A \in \Sigma\} \cup \{A \sqcap X \sqsubset \bot \mid O \models A \sqcap X \sqsubset \bot \wedge A \in \Sigma\}$. None of the ontologies in the corpus contain strict subsumptions ($\sqsubset$), as such in practice we use regular subsumptions ($\sqsubseteq$) to emulate the ANEMONE approach. We refer to descriptions extracted using this method as *ANEMONE descriptions*. For simplification and understanding, we streamline ANEMONE descriptions into two perspectives:

1. The Ability to extract subsumers of an unshared concept name.

2. The Ability to extract subsumees of an unshared concept name

7.2.1 The Ability to extract subsumers of an unshared concept name

An ANEMONE description contains information about the subsumers of an unshared concept name. Following the description above, the subsumers of an unshared concept name X with respect to a common signature Σ (such that $X \notin \Sigma$) may be emulated using the union of the following sets of axioms:

1. $\{X \equiv A \mid O \models X \equiv A \wedge A \in \Sigma\}$
2. $\{X \sqsubseteq A \mid O \models X \sqsubseteq A \wedge A \in \Sigma\}$
3. $\{X \sqsubseteq \neg A \mid O \models X \sqsubseteq \neg A \wedge A \in \Sigma\}$.

Let $A_{\equiv}$ denote the set of all concept names A in Σ such that $O \models A \equiv X$ (i.e., the set of As in the set of axioms in (1)). Let $A_{\sqsubseteq}$ denote the set of all concept names A in Σ such that $O \models X \sqsubseteq A$ (i.e., the set of As in the set of axioms in (2)). Let $A_{\bot}$ denote the set of all concept names A in Σ such that $O \models X \sqcap A \sqsubseteq \bot$ (i.e., the set of As in the set of axioms in (3)). We call the **ANEMONE-subsumer** of X with respect to Σ under O denoted *ANEMONE-subsumer*$(X; O; \Sigma)$:

$$\bigsqcap_{A \in A_{\equiv}} A \sqcap \bigsqcap_{A' \in A_{\sqsubseteq}} A' \sqcap \bigsqcap_{A'' \in A_{\bot}} \neg A''$$

7.2.2 The Ability to extract subsumees of an unshared concept name

In a dual fashion, an ANEMONE description contains information about the subsumees of an unshared concept name. Following the description above, the subsumees of an unshared concept name X with respect to a common signature Σ (such that $X \notin \Sigma$) is the union of the following sets of axioms:

1. $\{A \equiv X \mid O \models A \equiv X \wedge A \in \Sigma\}$
2. $\{A \sqsubseteq X \mid O \models A \sqsubseteq X \wedge A \in \Sigma\}$.

Let $A_{\equiv}$ denote the set of all concept names A in Σ such that $O \models A \equiv X$ (i.e., the set of As in the set of axioms in (a)). Let $A_{\sqsubseteq}$ denote the set of all concept names A in Σ such

that $O \models A \sqsubseteq X$ (i.e., the set of As in the set of axioms in (b)). We call the **ANEMONE-subsumee** of X with respect to Σ under O denoted *ANEMONE-subsumee*$(X;O;\Sigma)$:

$$\bigsqcup_{A \in A_{\equiv}} A \sqcup \bigsqcup_{A' \in A_{\sqsubseteq}} A'$$

.

Let O be an ontology, X a concept name, and Σ a subset signature of $sig(O)$ such that $X \in sig(O)$, and $X \notin \Sigma$. To construct an ANEMONE-description of X with respect to Σ under O using ANEMONE-subsumers and ANEMONE-subsumees, we use Algorithm 7.2.1. It is worth noting that all the processes in Algorithm 7.2.1 can be simply implemented using a reasoner such as Hermit [GHM+14].

Proposition 13. *Let O be an ontology and C, C' be two arbitrary concepts in O. If $O \models C \sqsubseteq C'$, then $O \models C \equiv C \sqcap C'$.*

Proof. Since $O \models C \sqsubseteq C'$, then (i) $O \models C \sqcap C' \sqsubseteq C$ (ii) $O \models C \sqsubseteq C \sqcap C'$. From (i) and (ii), we have $O \models C \equiv C \sqcap C'$. □

Proposition 14. *Let O be an ontology and C, C' be two arbitrary concepts in O. If $O \models C \sqsubseteq C'$, then $O \models C' \equiv C \sqcup C'$.*

Proof. Since $O \models C \sqsubseteq C'$, then (i) $O \models C' \sqsubseteq C' \sqcup C$ (ii) $O \models C' \sqcup C \sqsubseteq C'$. From (i) and (ii), we have $O \models C' \equiv C \sqcup C'$. □

From Propositions 13 and 14, it follows that if an uncommon concept name X has at least one equivalent concept name $A \in \Sigma$, then both the ANEMONE-subumer and ANEMONE-subsumee of X are definitions for X. Thus, either an ANEMONE-subumer or ANEMONE-subsumee can be used in place of an uncommon concept name X if they are both definitions of X when communicating in the normal communication protocol.

Streamlining ANEMONE descriptions into *subsumers* and *subsumees* also enables us to compare and contrast the ANEMONE approach for extracting descriptions to an approach using strongest necessary conditions, weakest sufficient conditions, compiled superconcepts, and compiled subconcepts. Observe that methods for extracting *subsumers* of concept names include: ANEMONE-subsumers, strongest necessary conditions, and compiled superconcepts. Similarly, methods for extracting *subsumees* of concept names include: ANEMONE-subsumees, weakest sufficient conditions, and compiled subconcepts. It is also worth noting that all methods can be potentially used

ALGORITHM 7.2.1

Input: An ontology O, a concept name X, a signature Σ, where $\Sigma \subseteq sig(O)$, $X \in sig(O)$, $X \notin \Sigma$.

Output: An ontology ϕ which is an ANEMONE description for X.

Step 1: Initialise ϕ to be $\emptyset$.

Step 2: Extract *ANEMONE-subsumer*$(X;O;\Sigma)$.

Step 3: Extract *ANEMONE-subsumee*$(X;O;\Sigma)$.

Step 4: **if** $O \models A \equiv$*ANEMONE-subsumer*$(X;O;\Sigma)$ **then:**

$\phi \leftarrow A \equiv$ *ANEMONE-subsumer*$(X;O;\Sigma)$

else:

$\phi \leftarrow A \sqsubseteq$ *ANEMONE-subsumer*$(X;O;\Sigma)$

Step 5: **if** $O \models A \equiv$*ANEMONE-subsumee*$(X;O;\Sigma)$ **then:**

$\phi \leftarrow A \equiv$ *ANEMONE-subsumee*$(X;O;\Sigma)$

else:

$\phi \leftarrow$ *ANEMONE-subsumee*$(X;O;\Sigma) \sqsubseteq A$

Step 6: return ϕ.

to extract definitions for concept names depending on the common signature of interest or expressivity of the ontology.

The extraction of subsumees is specifically relevant for the application of agents querying each other for instances of concepts. This is because for uncommon concepts that are not definable, an extracted subsumee is more suitable than an extracted subsumer as all instances of subsumees are also instances of the concept: if $O \models A \sqsubseteq B$, then all instances of A are instances of B. Instances of a subsumer may include instances that do not belong to the uncommon concept.

Example 7.2.1. The example is adapted from [vDBD+06]. In it, we have two agents AG_1 and AG_2 which are both personal news agents that classify news articles according to the ontologies provided in Table 7.2.

Ontology O_1 of agent AG_1	Ontology O_2 of agent AG_2
LawnTennis $\sqsubseteq$ BallAndRacquetGames, Wimbledon $\sqsubseteq$ LawnTennis, UKNews $\sqsubseteq$ RegionalNews, SoftwareAgents $\sqsubseteq$ ComputerScience, ComputerScience $\sqsubseteq$ ScienceNews, BallAndRacquetGames $\sqcap$ RegionalNews $\sqsubseteq \bot$, BallAndRacquetGames $\sqcap$ ScienceNews $\sqsubseteq \bot$, RegionalNews $\sqcap$ ScienceNews $\sqsubseteq \bot$	LawnTennis $\equiv$ Tennis Tennis $\sqsubseteq$ RacquetGames, RacquetGames $\sqsubseteq$ BallAndRacquetGames, BallAndRacquetGames $\sqsubseteq$ Sports, EuropeNews $\sqsubseteq$ RegionalNews, SoftwareAgents $\sqsubseteq$ ScienceNews, Sports $\sqcap$ RegionalNews $\sqsubseteq \bot$, Sports $\sqcap$ ScienceNews $\sqsubseteq \bot$, RegionalNews $\sqcap$ ScienceNews $\sqsubseteq \bot$

Table 7.2: Ontologies of the Agents in the ANEMONE paper. The first column contains the ontology O_1 for an agent AG_1, and the second column contains the ontology O_2 for an agent AG_2

We have that:

$$\Sigma = sig(O_1) \cap sig(O_2) =$$

$$\{LawnTennis, RegionalNews, SoftwareAgents, ScienceNews, BallAndRacquetGames\}$$

The ANEMONE-subsumer for *Wimbledon* with respect to Σ under O_1 is:

$$BallAndRacquetGames \sqcap LawnTennis \sqcap$$

$$\neg RegionalNews \sqcap \neg SoftwareAgents \sqcap \neg ScienceSubjects$$

There is no ANEMONE-subsumee for *Wimbledon* with respect to Σ under O_1. Furthermore, $O_1 \not\models Wimbledon \equiv ANEMONE\text{-}subsumer(Wimbledon; O_1; \Sigma)$. Thus the ANEMONE description for *Wimbledon* with respect to Σ under O_1 would be:

$$Wimbledon \sqsubseteq BallAndRacquetGames \sqcap LawnTennis \sqcap$$

$$\neg RegionalNews \sqcap \neg SoftwareAgents \sqcap \neg ScienceSubjects$$

Note that the ANEMONE-subsumber in this case is also an SNC for *Wimbledon* with respect to Σ under O.

In $\mathcal{ALC}$ ontologies, concepts may be defined using *rich* concept descriptions that may include constructs such as role restrictions and disjunctions (which exceed the expressivity of $DL_{ANEMONE}$). When communicating in the normal communication

protocol, agents must only use concepts in the common vocabulary or concepts equivalent to concepts not in the common vocabulary, provided such equivalent concepts are expressed in the common vocabulary. ANEMONE-subsumers and ANEMONE-subsumees are not complete methods for extracting definitions in $\mathcal{ALC}$, as they are restricted to definitions only involving concept names, and $\mathcal{ALC}$ allows for more complex constructs such as value restrictions. The requirements of communication in the normal communication protocol suggest that only using ANEMONE-subsumers and ANEMONE-subsumees in a system with agents that have $\mathcal{ALC}$ ontologies may unnecessarily lead to a loss in communication (because of the incompleteness of ANEMONE-subsumers and ANEMONE-subsumees for $\mathcal{ALC}$) and cause the agents switch to the concept definition or explication protocols, thus potentially increasing the cost of communication and possible errors in communication. This problem also extends to the concept definition protocol and to concepts without any equivalent concepts because the higher expressivity[3] of $\mathcal{ALC}$ implies that rich concept descriptions may exist in the agent's ontologies, thus ANEMONE descriptions for concepts in such ontologies may result in some information loss.

Example 7.2.2. Let AG_1 denote an agent with the following $\mathcal{EL}$ ontology O_1:

α_1: $A \equiv \exists r.C$

α_2: $A \sqsubseteq D$

Let AG_2 denote an agent with the following $\mathcal{EL}$ ontology O_2:

β_1: $D \equiv \exists r.C$

Let $\Sigma = sig(O_1) \cap sig(O_2) = \{r, C, D\}$. Let A be the concept name that AG_1 wishes to convey to AG_2 in the normal communication protocol. We have that *ANEMONE-subsumer*$(A; O_1; \Sigma) = D$ which is not equivalent to A and thus the agents must switch to the concept definition protocol where $A \sqsubseteq D$ is conveyed as the ANEMONE-description of A. However, observe that $O_1 \models A \equiv \exists r.C$ and thus $\exists r.C$ (which is a strongest necessary and weakest sufficient condition of A) can be conveyed in place of A by AG_1 to AG_2, and thus there is no loss in communication that triggers a switch to the concept definition protocol.

[3] *Higher* relative to $\mathcal{DL}_{ANEMONE}$.

Example 7.2.2 illustrates that ineffective communication may arise when there is information that can prevent descent into the lower communication layers but can not be extracted. To avoid such inefficiencies, we define notions of *completeness* of *subsumers* and *subsumees* of concept names that ensure that *subsumers* and *subsumees* are maximally precise with respect to the ontologies of the agents and do not result in unnecessary loss in communication.

7.3 Contextual Completeness for Subumers and Subsumees

Let O be an ontology, X a concept name, and Σ a subset of $sig(O)$. We define two notions of *contextual* completeness of subsumers and subsumees of X with respect to Σ under O. Here the word *contextual* is used to indicate that these notions of completeness are always with respect to the *context* of the ontology O.

Definition 42 (Contextual Completeness for Subsumers)**.** *Let O be a TBox, X a concept name and Σ a subset of $sig(O)$. Let C be a subsumer of X such that $O \models X \sqsubseteq C$. If X is definable under Σ, we say C is a* contextually complete subsumer *of X with respect to Σ if $O \models X \equiv C$ and $sig(C) \subseteq \Sigma$. If X is not definable under Σ, we say C is a* contextually complete subsumer *of X with respect to Σ if $sig(C) \subseteq \Sigma$ and for any other C' such that $sig(C') \subseteq \Sigma$ and $O \models X \sqsubseteq C'$, we have that $O \models C \sqsubseteq C'$.*

Let C and C' be two subsumers of X with respect to Σ such that $O \models X \sqsubseteq C$, $O \models X \sqsubseteq C'$, $sig(C) \subseteq \Sigma$, and $sig(C') \subseteq \Sigma$. We say C is more contextually complete than *C' (C' is* less contextually complete than *C) if $O \models C \sqsubseteq C'$ and $O \not\models C \equiv C'$.*

Let $\mathcal{L}$ be a description logic. A concept Ω is said to be contextually subsumer complete *for $\mathcal{L}$ if for any input $\mathcal{L}$-concept X, $\mathcal{L}$-TBox O, and signature Σ such that $\Sigma \subseteq sig(O)$, Ω returns a* contextually complete *subsumer of X under O.*

None of the concept conveying mechanisms discussed in this thesis are contextually subsumer complete for $\mathcal{ALC}$.

Definition 43 (Contextually Complete Subsumee)**.** *Let O be a TBox, X a concept name and Σ a subset of $sig(O)$. Let C be a subsumee of X such that $O \models C \sqsubseteq X$. If X is definable under Σ, we say C is a* contextually complete subsumee *of X with respect to Σ if $O \models X \equiv C$ and $sig(C) \subseteq \Sigma$. If X is not definable under Σ, we say C is a* contextually complete subsumee *of X with respect to Σ if $sig(C) \subseteq \Sigma$ and for any other C' such that $sig(C') \subseteq \Sigma$ and $O \models C' \sqsubseteq X$, we have that $O \models C' \sqsubseteq C$.*

Let C and C' be two subsumees of X with respect to Σ such that $O \models C \sqsubseteq X$, $O \models C' \sqsubseteq X$, $sig(C) \subseteq \Sigma$, and $sig(C') \subseteq \Sigma$. We say C is more contextually complete than *C' (C' is* less contextually complete than *C) if $O \models C' \sqsubseteq C$ and $O \not\models C \equiv C'$.*

Let $\mathcal{L}$ be a description logic. A a concept conveying mechanism Ω is said to be contextually subsumee complete *for $\mathcal{L}$ if for any input $\mathcal{L}$-concept X, $\mathcal{L}$-TBox O, and signature Σ such that $\Sigma \subseteq sig(O)$, Ω returns a* contextually complete *subsumee of X under O.*

None of the concept conveying mechanisms discussed in this thesis are contextually subsumee complete for $\mathcal{ALC}$.

The use of the word *definable* in Definitions 42 and 43 is meant in the sense of Definition 9.

Example 7.2.2 illustrates that ANEMONE-subsumers are not contextually subsumer complete for $\mathcal{EL}$ ontologies, and thus can not be contextually subsumee complete for DLs more expressive than $\mathcal{EL}$. A similar example can easily be constructed to show that ANEMONE-subsumees are not contextually subsumee complete for $\mathcal{EL}$ ontologies.

Contextual completeness allows us to evaluate how complete a subsumer or a subsumee of a concept name is with respect to a background ontology. If a communicating agent AG_1 has the ability to extract a contextually complete subsumer for an definable (under the common vocabulary) uncommon concept name X, then it is guaranteed to *never* unnecessarily switch from the normal communication protocol to the concept definition protocol. Similarly if X is not definable with respect to the common vocabulary and AG_1 has the ability to extract a subsumer and subsumee of X that are contextually complete, then it can extract a maximally precise concept description of X in the concept definition protocol, and thus not unnecessarily switch to the concept explication protocol.

Limitation of Contextual Completeness for Agent Communication

It is important to stress that *contextual* completeness is not sufficient for agent communication as *completeness* in this sense is only with respect to the ontology of the speaking agent. Let X be an uncommon concept name that an agent AG_1 is attempting to convey to another agent AG_2 with respect to Σ such that $\Sigma = sig(O_1) \cap sig(O_2)$ where O_1 and O_2 are the respective ontologies of AG_1 and AG_2. Regardless of whether X is defined or not, a contextually complete subsumer or subsumee of X with respect to Σ can still lead to communication errors. This is because AG_2 does not have access

to O_1, and X may have other relations to concepts that are expressible in Σ that are not captured by contextually complete subsumers or subsumees. Example 7.3.1 illustrates this.

Example 7.3.1. Let AG_1 denote an agent with the following ontology O_1:

α_1: $A \equiv \exists r.C$

α_2: $A \sqsubseteq D$

Let AG_2 denote an agent with the following ontology O_2:

β_1: $\exists r.C(i)$

β_2: $\neg D(i)$

Let $\Sigma = sig(O_1) \cap sig(O_2) = \{r, C, D\}$. Let A be the concept name that AG_1 wishes AG_2 to extract instances for. A contextually complete subsumer of A with respect to Σ under O is $\exists r.C$. If AG_1 conveys $\exists r.C$ in place of A, then AG_2 will return i as an instance that belongs to A, and AG_1 may erroneously add $A(i)$ to its knowledge. However, observe that this contradicts AG_1's knowledge: $O_2 \models (\exists r.C \sqcap \neg D)(i)$ while $O_1 \cup \{A(i)\} \models (\exists r.C \sqcap D)(i)$. A more contextually complete subsumer of A that could prevent this contradiction is $\exists r.C \sqcap D$.

This example highlights the fact that the lack of minimality of strongest necessary conditions as highlighted in Chapter 6 may not be so disadvantageous when it comes to standalone completeness, as a concept that may be deemed *superfluous* (in Chapter 6), is only so with respect to the ontology, but may be essential for another agent to know. In fact, given the ontology in Example 7.3.1, our implementation for extracting strongest necessary conditions will extract $\exists r.C \sqcap D$ as a strongest necessary condition of A with respect to the signature $\{r, C, D\}$.

7.4 Standalone Completeness for Subumers and Subsumees

Section 7.3 highlights the fact that when communicating agents do not have access to each other's ontologies, they must find a way to extract maximally precise subsumers and subsumees of uncommon concepts in a fashion that extracts all information about subsumers and subsumees from their respective ontologies. To capture this, we introduce *standalone* notions of completeness for subsumers and subsumees.

Definition 44 (Standalone Completeness for Subsumers)**.** *Let O be a TBox, X a concept name and Σ a subset of $sig(O)$. Let C be a subsumer of X such that $O \models X \sqsubseteq C$. We say C is a* standalone complete *subsumer of X with respect to Σ if C is a contextually complete subsumer of X with respect to Σ, and for any other C' such that $sig(C') \subseteq \Sigma$ and $O \models X \sqsubseteq C'$, we have that $\models C \sqsubseteq C'$.*

Let C and C' be two subsumers of X with respect to Σ such that $O \models X \sqsubseteq C$, $O \models X \sqsubseteq C'$, $sig(C) \subseteq \Sigma$, and $sig(C') \subseteq \Sigma$. We say C is more standalone complete than *C' (C' is* less standalone complete than *C) if $\models C \sqsubseteq C'$ and $\not\models C \equiv C'$.*

Let $\mathcal{L}$ be a description logic. A a concept conveying mechanism Ω is said to be standalone subsumer complete *for $\mathcal{L}$ if for any input $\mathcal{L}$-concept X, $\mathcal{L}$-TBox O, and signature Σ such that $\Sigma \subseteq sig(O)$, Ω returns a* standalone complete *subsumer of X under O.*

Example 7.4.1. Let O be the set of the following axioms:

1. $A \sqsubseteq \exists r.C$
2. $B \equiv \exists r.C$
3. $B \sqsubseteq D$

Let $\Sigma = sig(O) \setminus \{A\} = \{r, C, B, D\}$. Let A be the concept name to be conveyed.
A possible strongest necessary condition of A with respect to Σ under O is $\exists r.C$
The ANEMONE-subsumer of A with respect to Σ under O is $B \sqcap D$.
The compiled superconcept of A with respect to Σ is $\exists r.C \sqcap B \sqcap D$.
Therefore only $CSP(A; O; \Sigma)$ is standalone complete ($\models \exists r.C \sqcap B \sqcap D \sqsubseteq B \sqcap D$, $\models \exists r.C \sqcap B \sqcap D \sqsubseteq \exists r.C$). Furthermore $CSP(A; O; \Sigma)$ is more standalone complete than $SNC(A; O; \Sigma)$ and *ANEMONE-subsumer*$(A; O; \Sigma)$. $SNC(A; O; \Sigma)$ is more contextually complete than *ANEMONE-subsumer*$(A; O; \Sigma)$ but not more standalone complete.
ANEMONE-subsumer$(A; O; \Sigma)$ is not more standalone complete than $SNC(A; O; \Sigma)$. However, observe that *ANEMONE-subsumer*$(A; O; \Sigma) \sqcap SNC(A; O; \Sigma)$ is more standalone and contextually complete than *ANEMONE-subsumer*$(A; O; \Sigma)$ and $SNC(A; O; \Sigma)$, and in fact a compiled superconcept of A with respect to Σ.

Now consider Σ to be $\{B, D\}$.
A possible strongest necessary condition of A with respect to Σ under O is B
The ANEMONE-subsumer of A with respect to Σ under O is $B \sqcap D$.
The compiled superconcept of A with respect to Σ is $B \sqcap D$. Therefore only $CSP(A; O; \Sigma)$ and *ANEMONE-subsumer*$(A; O; \Sigma)$ are standalone complete.

Definition 45 (Standalone Complete Subsumee)**.** *Let O be a TBox, X a concept name and Σ a subset of $sig(O)$. Let C be a subsumee of X such that $O \models C \sqsubseteq X$. If X is definable under Σ, we say C is contextually is a* standalone complete *subsumee of X if C is a contextually complete subsumee of X with respect to Σ and C' such that $sig(C') \subseteq \Sigma$ and $O \models C' \sqsubseteq X$, we have that $\models C' \sqsubseteq C$.*

Let C and C' be two subsumees of X with respect to Σ such that $O \models C \sqsubseteq X$, $O \models C' \sqsubseteq X$, $sig(C) \subseteq \Sigma$, and $sig(C') \subseteq \Sigma$. We say C is more standalone complete than *C' (C' is* less standalone complete than *C) if $O \models C' \sqsubseteq C$ and $O \not\models C \equiv C'$.*

Let $\mathcal{L}$ be a description logic. A a concept conveying mechanism Ω is said to be standalone subsumee complete *for $\mathcal{L}$ if for any input $\mathcal{L}$-concept X, $\mathcal{L}$-TBox O, and signature Σ such that $\Sigma \subseteq sig(O)$, Ω returns a* standalone complete *subsumee of X under O.*

None of the concept conveying mechanisms discussed in this thesis are standalone complete for $\mathcal{ALC}$. However, we believe that there are classes of ontologies for which ANEMONE-subsumers and ANEMONE-subsumees are contextually subsumer complete and contextually subsumee complete.

Examples include:

1. $\mathcal{EL}$ ontologies without any existential quantification.

2. $\mathcal{AL}$ ontologies without any universal or existential quantifications.

Definition 46 (ANEMONE fragment)**.** *We say an ontology O falls under the ANEMONE fragment if it satisfies the following syntactic requirements:*

1. *Axioms involving concept names in O are* only *of the form $A \equiv C$ or $A \sqsubseteq C$ such that:*

   (a) *A is always an atomic concept.*

   (b) *C is always either an atomic concept, a negated atomic concept, or a conjunction $L_1 \sqcap L_2 \sqcap \ldots \sqcap L_N$ where L_i (such that $1 \leq i \leq N$) is either an atomic concept or a negated atomic concept.*

The key differences between $DL_{ANEMONE}$ and the ANEMONE fragment is that (1) $DL_{ANEMONE}$ involves the use of strict subsumptions ($\sqsubset$) while the ANEMONE fragment allows for subsumptions ($\sqsubseteq$). (2) $DL_{ANEMONE}$ includes the overlap ($\oplus$) operator while the ANEMONE fragment does not.

Conjecture 2. ANEMONE-subsumers and ANEMONE-subsumees are standalone subsumer complete and standalone subsumee complete (respectively) for ontologies O that fall under the ANEMONE fragment.

The reasoning behind Conjecture 2 is that for an ontology O that falls within the ANEMONE fragment, we believe that due to the syntactic restriction in Definition 46, all subsumers of any concept name $A \in sig(O)$ will be equivalent-to or a superconcept-of a subsumer C of A which has the form: $C_1 \sqcap \ldots \sqcap C_n$ where each C_i such that $1 \leq i \leq n$ is either an atomic concept or a negated atomic concept. Similarly, we believe that due to the syntactic restriction in Definition 46, all subsumees of any concept name $A \in sig(O)$ will be equivalent-to or a subconcept-of a subsumee C of A which has the form: $A_1 \sqcup \ldots \sqcup A_n$ where each A_i such that $1 \leq i \leq n$ is an atomic concept.

7.4.1 Relationship between CSPs, CSBs, SNCs, WSCs, ANEMONE subsumers, ANEMONE subsumees and Contextual/Standalone completeness

As demonstrated in Section 6.4, extraction of compiled superconcepts is often infeasible. ANEMONE-subsumers and ANEMONE-subsumees can be extracted easily using an off-the-shelf reasoner such as Hermit [GHM+14]. Reasoning tasks for concept names (subsumption checking, satisfiability checking, equivalence checking, and disjointness checking) all have been shown [BCM+03] to have worst case exponential complexity for $\mathcal{ALC}$ TBoxes. Observe that all of the reasoning tasks required to extract ANEMONE-subsumers and ANEMONE-subsumees fall under subsumption checking, satisfiability checking, equivalence checking, and disjointness checking, and thus we expect the extraction of ANEMONE-subsumers and ANEMONE-subsumees to have an exponentially complexity in the worst case. Results from Chapter 6 suggest that the extraction of strongest necessary conditions (and similarly weakest sufficient conditions) using Algorithm 5.1.1 (Algorithm 5.1.2 for WSCs) may be feasible in some cases; however, both Algorithm 5.1.1 and 5.1.2 have a double exponential complexity (Section 5.1.3).

Example 7.4.1 highlights the fact that depending on the uncommon concept name and signature, ANEMONE subsumers are sometimes standalone complete, regardless of the expressivity of the ontology of interest. Furthermore, Example 7.4.1 highlights that the contextual completeness of SNCs can sometimes be used to make ANEMONE-subsumers more contextually complete (a dual example for subsumees can be easily

constructed). Let O be an ontology, X an uncommon concept name and Σ a subset of $sig(O)$, Example 7.4.1 highlights that if an $SNC(X;O;\Sigma)$ is more contextually complete than *ANEMONE-subsumer*$(X;O;\Sigma)$ then $SNC(X;O;\Sigma) \sqcap$ *ANEMONE-subsumer*$(X;O;\Sigma)$ is going to be more standalone complete than *ANEMONE-subsumer*$(X;O;\Sigma)$. However, if $O \models SNC(X;O;\Sigma) \equiv$ *ANEMONE-subsumer*$(X;O;\Sigma)$, then $SNC(X;O;\Sigma)$ only makes *ANEMONE-subsumer*$(X;O;\Sigma)$ more standalone complete if $\models SNC(X;O;\Sigma) \sqsubseteq$ *ANEMONE-subsumer*$(X;O;\Sigma)$ i.e., $SNC(X;O;\Sigma)$ is more standalone complete than *ANEMONE-subsumer*$(X;O;\Sigma)$.
If $O \models SNC(X;O;\Sigma) \equiv$ *ANEMONE-subsumer*$(X;O;\Sigma)$, and $\models$ *ANEMONE-subsumer*$(X;O;\Sigma) \sqsubseteq SNC(X;O;\Sigma)$, then $SNC(X;O;\Sigma)$ does not make *ANEMONE-subsumer*$(X;O;\Sigma)$ more standalone complete in any way, because it means that with respect to the ontology *and* the empty ontology, the ANEMONE-subsumer conveys as much or more than (in the case of $\models$ *ANEMONE-subsumer*$(X;O;\Sigma) \sqsubseteq SNC(X;O;\Sigma)$) the strongest necessary condition.

7.5 Empirical Evaluation

The following empirical evaluation aims to explore how frequently SNCs and WSCs are useful in making ANEMONE-subsumers and ANEMONE-subsumees of concept names occuring in the ontologies from Section 6.1 more standalone complete.

Because SNCs and WSCs can be used to extract subsumers and subsumees that may concepts that involve role restrictions, we expect SNCs and WSCs to make ANEMONE-subsumers and ANEMONE-subsumees more standalone complete when extracting for non-cyclic concept names.

We implement the extraction of ANEMONE-subsumers and ANEMONE-subsumees using Hermit [GHM+14]. We implement Algorithms 5.1.1 and 5.1.2 using the LETHE implementation[4]. We impose a twenty minute timeout on algorithms as in a practical communication latency is key for efficient communication.

For variability, we sample various subset signatures of each ontology in our dataset to emulate the common signature. Given an ontology O, we take seven *unique* uniform samples of $sig(O)$. The seven samples are of different sizes with respect to the signature of the ontology. The signature sizes range from 10% to 70% of the ontology's signature (using an interval of 10%). These signature sizes determine the portion of the ontology's signature that is randomly selected to serve as the common signature. Let

[4]LETHE can be accessed at `https://lat.inf.tu-dresden.de/~koopmann/LETHE/`.

Σ be a sample common signature for an ontology O. We assume the agent is interested in querying for instances of concept-names in $sig(O) \setminus \Sigma$ (which we call the *query sample*). We then record how frequently the extracted ANEMONE-subsumers and ANEMONE-subsumees are made more standalone complete by the extracted SNCs and WSCs of *all* the concept names in $sig(O) \setminus \Sigma$.

7.5.1 Evaluating Extraction of Subsumers

In this section, we evaluate the extraction of subsumers on the corpus from Section 6.1. For an ontology O, and signature Σ such that $\Sigma \subseteq sig(O)$, the experiment pipeline was as follows:

1. **for** every concept name $A \in sig(O) \setminus \Sigma$ **do:**
   (a) Extract $SNC(A;O;\Sigma)$ following Algorithm 5.1.1.
   (b) Extract $ANEMONE\text{-}subsumer(A;O;\Sigma)$ using Hermit.
2. **if** $O \models SNC(A;O;\Sigma) \sqsubseteq ANEMONE\text{-}subsumer(A;O;\Sigma)$ and $O \not\models SNC(A;O;\Sigma) \equiv ANEMONE\text{-}subsumer(A;O;\Sigma)$ **do:**
   (a) Record $SNC(A;O;\Sigma)$ as making $ANEMONE\text{-}subsumer(A;O;\Sigma)$ more standalone complete

   else if $O \models SNC(A;O;\Sigma) \equiv ANEMONE\text{-}subsumer(A;O;\Sigma)$ **do:**
   (a) Record $ANEMONE\text{-}subsumer(A;O;\Sigma)$ as not being made more standalone complete by $SNC(A;O;\Sigma)$

   else if $\models SNC(A;O;\Sigma) \sqsubseteq ANEMONE\text{-}subsumer(A;O;\Sigma)$ and $\not\models SNC(A;O;\Sigma) \equiv ANEMONE\text{-}subsumer(A;O;\Sigma)$ **do:**
   (a) Record $SNC(A;O;\Sigma)$ as making $ANEMONE\text{-}subsumer(A;O;\Sigma)$ more standalone complete

   else if $\not\models SNC(A;O;\Sigma) \sqsubseteq ANEMONE\text{-}subsumer(A;O;\Sigma)$ and $\not\models ANEMONE\text{-}subsumer(A;O;\Sigma) \sqsubseteq SNC(A;O;\Sigma)$
   and $\not\models SNC(A;O;\Sigma) \equiv ANEMONE\text{-}subsumer(A;O;\Sigma)$ **do:**
   (a) Record $SNC(A;O;\Sigma)$ as making $ANEMONE\text{-}subsumer(A;O;\Sigma)$ more standalone complete

else:

(a) Record *ANEMONE-subsumer*$(A;O;\Sigma)$ as not being made more standalone complete by $SNC(A;O;\Sigma)$

Results

The results are displayed in Figures 7.2, 7.3, 7.4, and 7.5. We only display results for instances in which both the extraction of the ANEMONE-subsumer and strongest necessary conditions were successful. An extraction is deemed successful if no timeout occurs and the signature of the extracted subsumer is a subset of the sample signature. Algorithm 5.1.1 always timed out in the MHCRO and TCO cases, as such we exclude those cases from the result.

The ontology BNO falls under the ANEMONE-fragment (we have verified this syntactically). As such it is not that surprising the ANEMONE subsumer is more standalone complete than the SNC in all instances. This is most likely because our implementation for extracting the strongest necessary condition may not factor in other superconcepts of an uncommon concept name. For example, given the ontology $O = \{A \sqsubseteq B, B \sqsubseteq C\}$, and signature $\Sigma = \{B, C\}$, the strongest necessary condition of A with respect to Σ extracted by Algorithm 5.1.1 is B, while the ANEMONE-subsumer is $B \sqcap C$. EO, MEDEON, ONTOMA, EPIE, and EPIP are ontologies that do not fall under the ANEMONE-fragment but are all cases in which the extracted strongest necessary condition could not make the ANEMONE-subsumer more standalone complete for any concepts. These cases further support the notion that even though ANEMONE-subsumers are not standalone or contextually subsumer complete for a description logic, they are not necessarily always less useful than strongest necessary conditions (which have the ability to extract concepts that involve role restrictions). Table 7.3 summarises the results across all signature samples observed for all the ontologies in the dataset. As we can see, the strongest necessary conditions extracted using Algorithm 5.1.1 makes the ANEMONE-subsumer more standalone complete under fifty percent of the time in cases except for XEO, CCTOO, and PDON.

Ontology ID	ANE Standalone Complete	SNC Enhanced
ACESO	96 % (72)	4 % (3)
BNO	100 % (288)	0 % (0)
PDON	0 % (1)	99 % (176)
CHMO	90 % (774)	9 % (78)
RXNO	50 % (740)	49 % (722)
HPIO	96 % (880)	3 % (28)
MOP	96 % (5126)	3 % (206)
EPIE	100 % (173)	0 % (0)
EPIP	100 % (129)	0 % (0)
ONTOMA	100 % (138)	0 % (0)
ONTOPARON_SOCIAL	68 % (3148)	31 % (1460)
HP_O	55 % (302)	44 % (241)
ISSVA	67 % (294)	32 % (140)
GECKO	95 % (351)	4 % (15)
DLORO	53 % (385)	46 % (338)
EO	100 % (613)	0 % (0)
MEDEON	100 % (95)	0 % (0)
XEO	33 % (141)	66 % (278)
CCTOO	4 % (142)	95 % (2732)

Table 7.3: Subsumer extraction Results aggregated over all signature samples . The column titled 'ANE Standalone Complete' indicates the percentage and number of concept names for which (1) the extracted strongest necessary condition could not make the ANEMONE-subsumer more standalone complete and (2) the ANEMONE subsumer was more standalone complete than the strongest necessary condition across all common signature samples. The column titled 'SNC Enhanced' indicates the percentage and number of concept names for which the extracted strongest necessary condition made the ANEMONE-subsumer more standalone complete.

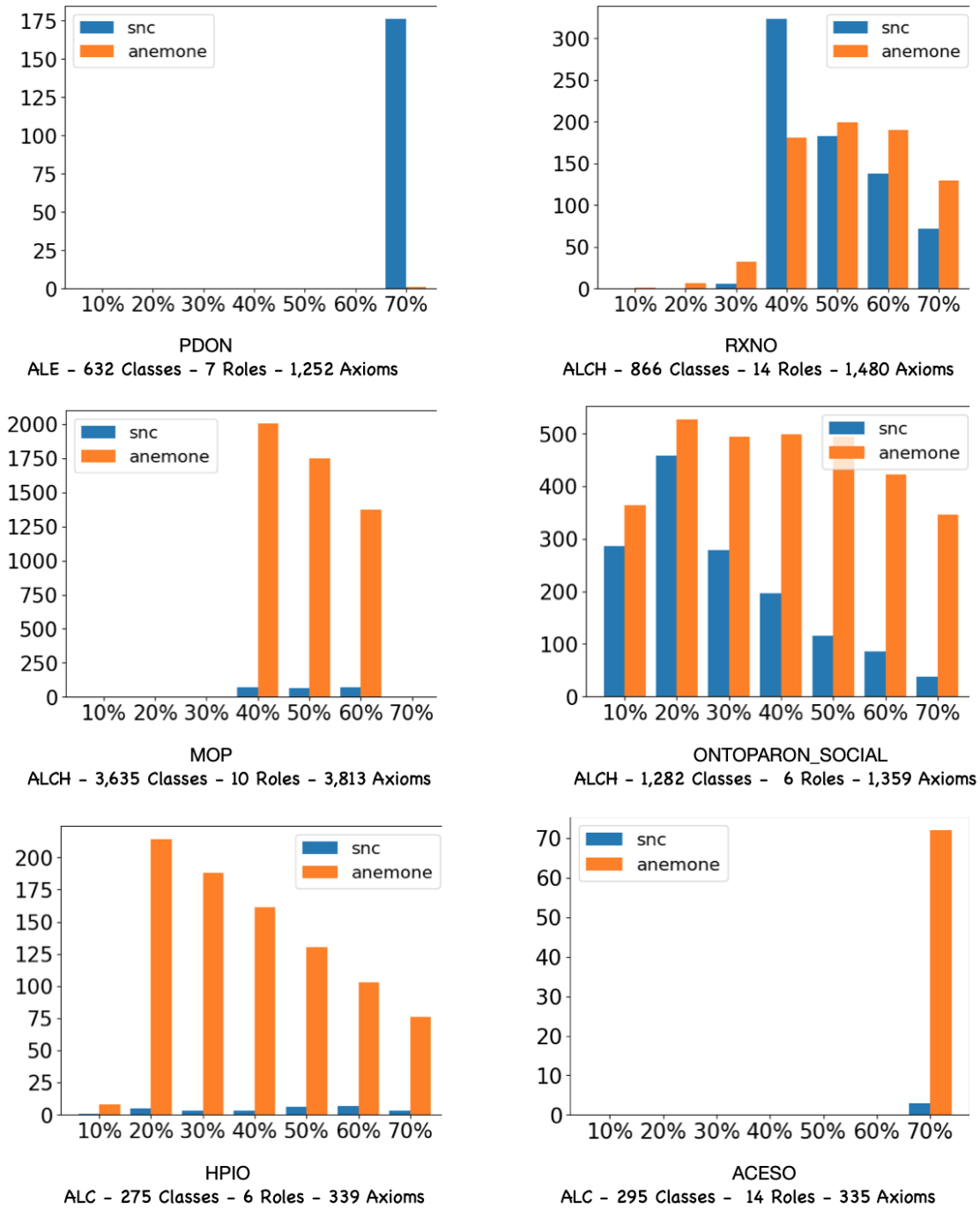

Figure 7.2: Results for subsumer extraction. The horizontal axis indicates the size of the sampled common signature as a percentage of the ontology's signature. The vertical axis indicates the number of concept names. The blue bars indicate the number of concepts for which the extracted strongest necessary condition made the extracted ANEMONE-subsumer more standalone complete. The orange bars indicate the number of concepts for which the ANEMONE-subsumer was not made more standalone complete by the extracted strongest necessary condition. The ticks without bars are cases in which either (1) the extracted SNC included a definer symbol (2) Algorithm 5.1.1 timed out when extracting an SNC.

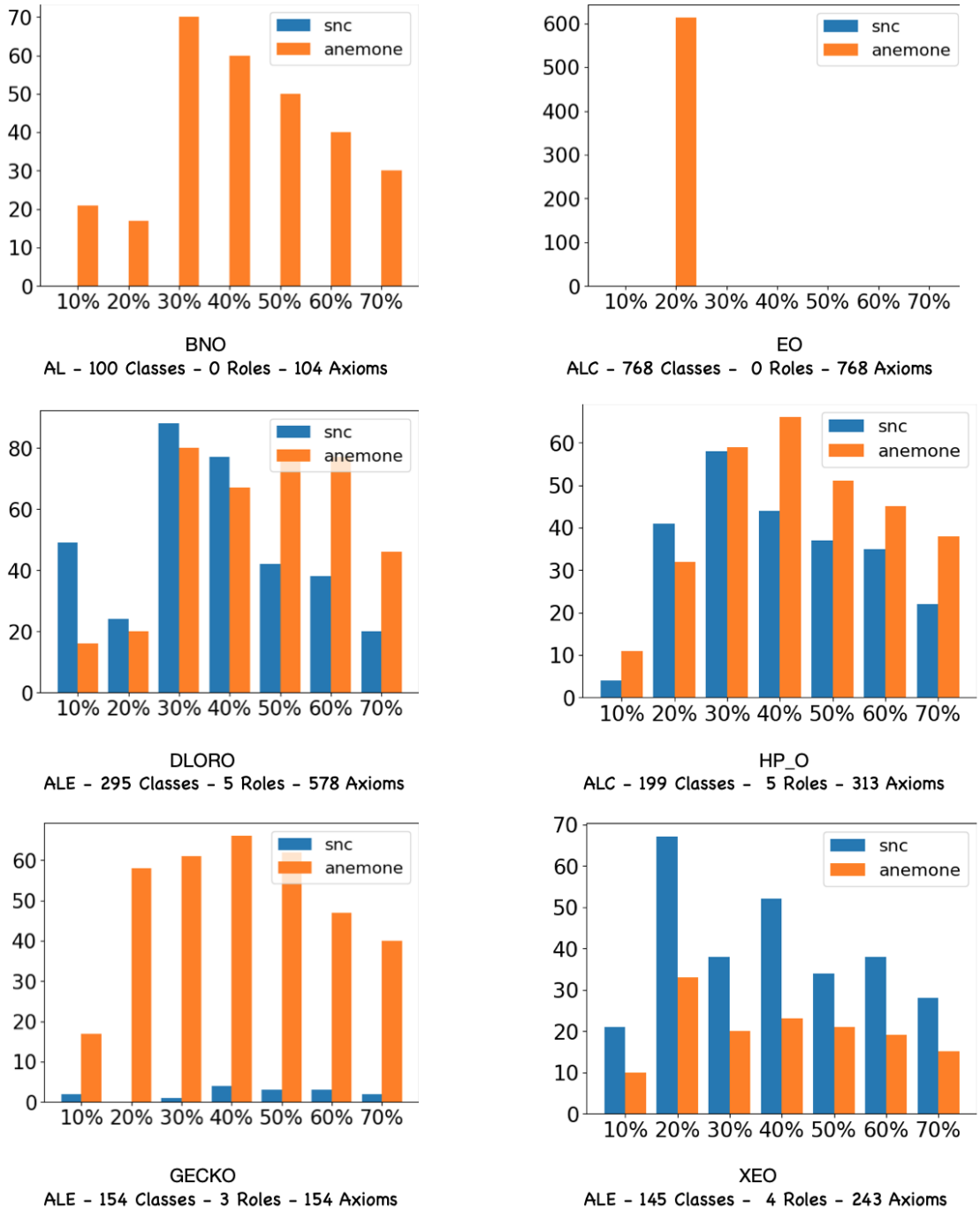

Figure 7.3: Results for subsumer extraction. See Figure 7.2 caption for details.

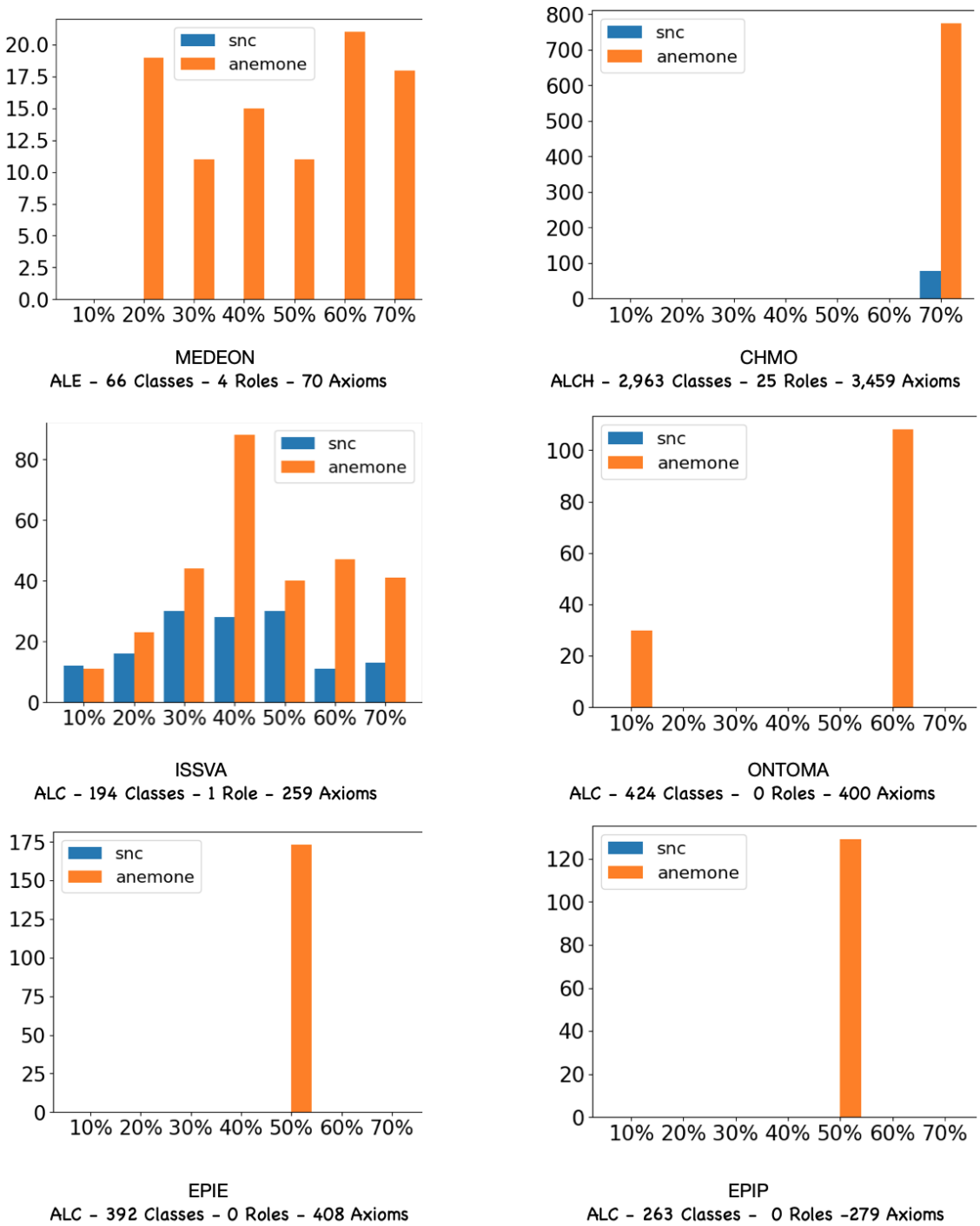

Figure 7.4: Results for subsumer extraction. See Figure 7.2 caption for details.

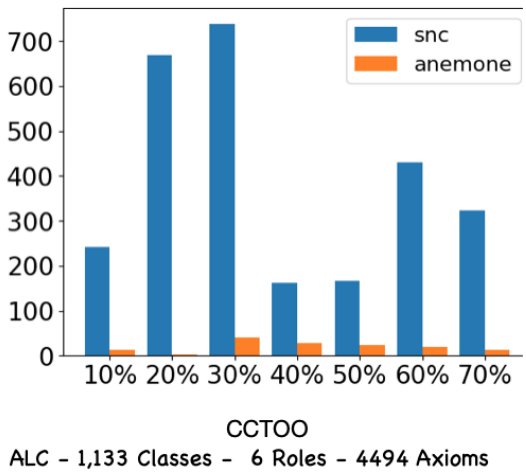

Figure 7.5: Results for subsumer extraction. See Figure 7.2 caption for details.

7.5.2 Evaluating Extraction of Subsumees

In this section we evaluate the extraction of subsumees on the corpus from Section 6.1. For an ontology O, and signature Σ such that $\Sigma \subseteq sig(O)$, the experiment pipeline was as follows:

1. **for** every concept name $A \in sig(O) \setminus \Sigma$ **do:**

   (a) Extract $WSC(A;O;\Sigma)$ following Algorithm 5.1.2.

   (b) Extract *ANEMONE-subsumee*$(A;O;\Sigma)$ using Hermit.

2. **if** $O \models$ *ANEMONE-subsumee*$(A;O;\Sigma) \sqsubseteq WSC(A;O;\Sigma)$ and $O \not\models$ *ANEMONE-subsumee*$(A;O;\Sigma) \equiv WSC(A;O;\Sigma)$**do:**

   (a) Record $WSC(A;O;\Sigma)$ as making *ANEMONE-subsumee*$(A;O;\Sigma)$ more standalone completeX

   else if $O \models WSC(A;O;\Sigma) \equiv$ *ANEMONE-subsumee*$(A;O;\Sigma)$ **do:**

   (a) Record *ANEMONE-subsumee*$(A;O;\Sigma)$ as not being made more standalone complete by $WSC(A;O;\Sigma)$

   else if $\models$ *ANEMONE-subsumee*$(A;O;\Sigma) \sqsubseteq WSC(A;O;\Sigma)$ and $\not\models WSC(A;O;\Sigma) \equiv$ *ANEMONE-subsumee*$(A;O;\Sigma)$ **do:**

   (a) Record $WSC(A;O;\Sigma)$ as making *ANEMONE-subsumee*$(A;O;\Sigma)$ more standalone complete

   else if $\not\models$ *ANEMONE-subsumee*$(A;O;\Sigma) \sqsubseteq WSC(A;O;\Sigma)$ and $\not\models$ *ANEMONE-subsumee*$(A;O;\Sigma) \sqsubseteq WSC(A;O;\Sigma)$ and $\not\models WSC(A;O;\Sigma) \equiv$ *ANEMONE-subsumee*$(A;O;\Sigma)$ **do:**

   (a) Record $WSC(A;O;\Sigma)$ as making *ANEMONE-subsumee*$(A;O;\Sigma)$ more standalone complete

   else:

   (a) Record *ANEMONE-subsumee*$(A;O;\Sigma)$ as not being made more standalone complete by $WSC(A;O;\Sigma)$

Results

The results are displayed in Figures 7.6, 7.7, 7.8, and 7.9. Similar to the previous section, we only display results for instances in which both the extraction of the ANEMONE-subsumee and weakest sufficient conditions were successful. An extraction is deemed successful if no timeout occurs and the signature of extracted subsumee is a subset of the sample signature.

Unlike the case with subsumer extraction, here we observe that for most of the ontologies, the weakest sufficient conditions extracted using Algorithm 5.1.2 rarely makes the ANEMONE subsumees more standalone complete. The exception is the PDON case where the weakest sufficient conditions extracted using Algorithm 5.1.2 made the ANEMONE-subsumee more standalone complete 100% of the time in the 70% sample common signature size iteration. In other cases, either the extracted weakest sufficient condition does not make the extracted ANEMONE-subsumees more standalone complete at all, or only makes the ANEMONE-subsumees more standalone complete a minimal percentage of the time. In fact from Table 7.4 we observe that except for the PDON case, the extracted weakest sufficient condition only makes the ANEMONE-subsumees more standalone complete less than 10% of the time, and 0% of the time in most cases across all signature size samples observed.

Ontology ID	ANE Standalone Complete	WSC Enhanced
ACESO	97 % (41)	2 % (1)
BNO	100 % (119)	0 % (0)
PDON	0 % (0)	100 % (129)
CHMO	99 % (244)	0 % (1)
RXNO	89 % (302)	10 % (35)
HPIO	98 % (236)	1 % (3)
MOP	99 % (971)	0 % (4)
EPIE	100 % (36)	0 % (0)
EPIP	100 % (27)	0 % (0)
ONTOMA	100 % (81)	0 % (0)
ONTOPARON_SOCIAL	97 % (1089)	2 % (31)
HP_O	97 % (106)	2 % (3)
ISSVA	94 % (103)	5 % (6)
GECKO	99 % (128)	0 % (1)
DLORO	100 % (243)	0 % (0)
EO	100 % (35)	0 % (0)
MEDEON	100 % (41)	0 % (0)
XEO	90 % (108)	10 % (12)
CCTOO	98 % (246)	1 % (3)

Table 7.4: Subsumee extraction Results aggregated over all signature samples. The column titled 'ANE Standalone Complete' indicates the percentage and number of concept names for which (1) the extracted weakest sufficient condition could not make the ANEMONE-subsumee more standalone complete and (2)the ANEMONE subsumee was more standalone complete than the weakest sufficient condition across all common signature samples. The column titled 'WSC Enhanced' indicates the percentage and number of concept names for which the extracted weakest sufficient condition made the ANEMONE-subsumee more standalone complete.

Figure 7.6: Results for subsumee extraction. The horizontal axis indicates the size of the sampled common signature as a percentage of the ontology's signature. The vertical axis indicates the number of concept names. The green bars indicate the number of concepts for which the extracted weakest sufficient condition made the extracted ANEMONE-subsumer more standalone complete. The orange bars indicate the number of concepts for which the ANEMONE-subsumee was not made more standalone complete by the extracted weakest sufficient condition. The ticks without bars are cases in which either (1) the extracted WSC included a definer symbol (2) Algorithm 5.1.2 timed out when extracting an WSC.

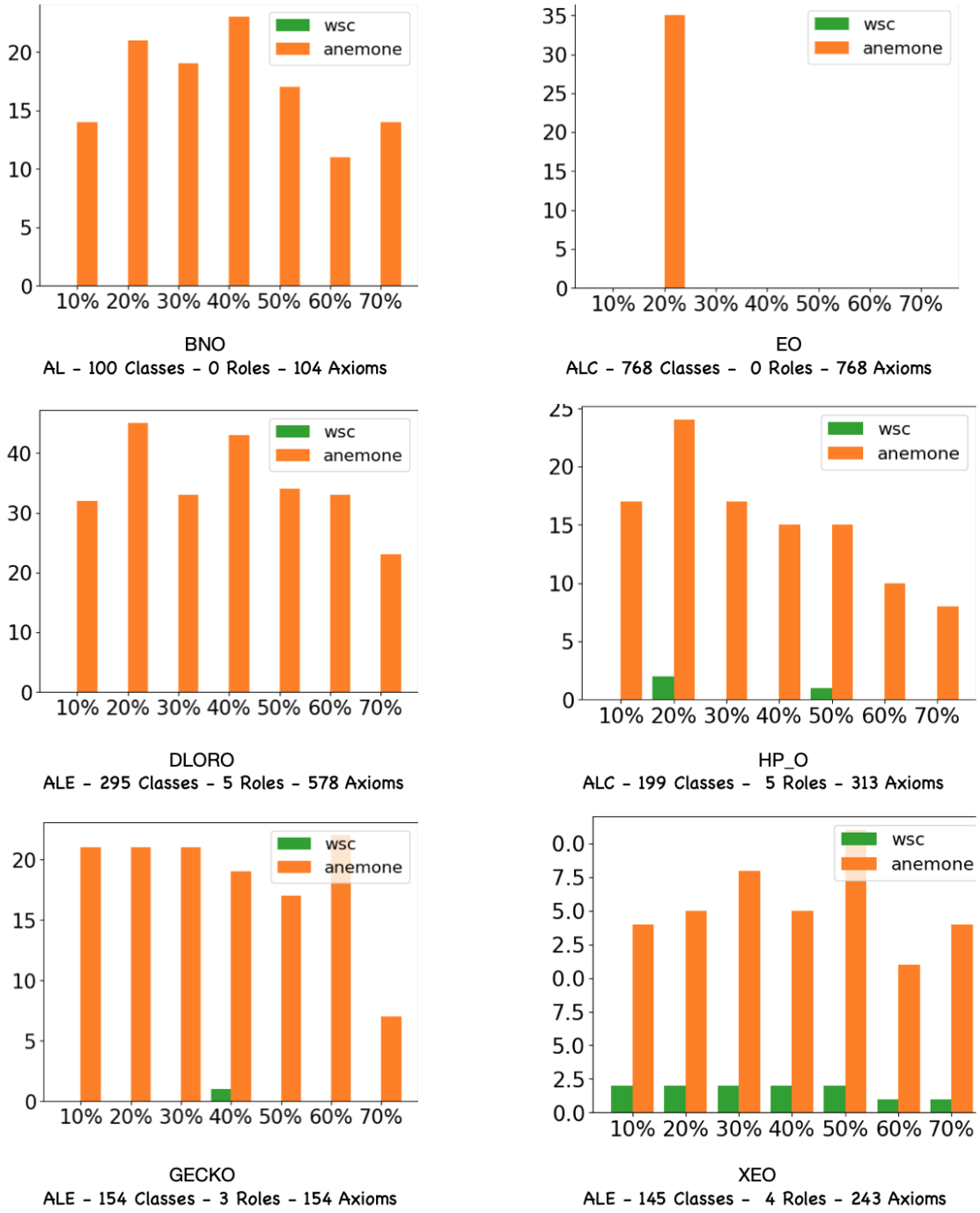

Figure 7.7: Results for subsumee extraction. See Figure 7.6 caption for details.

Figure 7.8: Results for subsumee extraction. See Figure 7.6 caption for details.

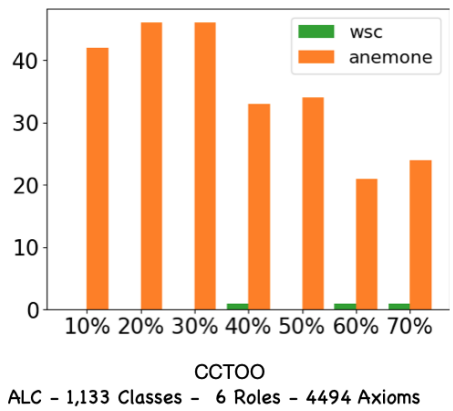

Figure 7.9: Results for subsumee extraction. See Figure 7.6 caption for details.

7.5.3 Discussion

ANEMONE-subsumers and ANEMONE-subsumees are the most practical and easily implemented concepts (out of those discussed up to this point in the thesis) for agent communication given that they may easily be implemented using an off-the-shelf reasoner such as Hermit. However, ANEMONE-subsumers and ANEMONE-subsumees do not guarantee contextual or standalone completeness for $\mathcal{ALC}$ ontologies without cycles. We have investigated how often our implementations for extracting strongest necessary and weakest sufficient conditions could be used to make ANEMONE-subsumers and ANEMONE-subsumees more standalone complete in practice. The results suggest that our implementation of Algorithm 5.1.1 for extracting strongest necessary conditions, when successful, seems to make ANEMONE-subsumers more standalone complete about half the time; the rest of the time, ANEMONE-subsumers are not made more standalone complete by the extracted strongest necessary conditions, and in one case (BNO) ANEMONE-subsumers are standalone complete for all concepts. However, it is worth noting that there are a significant number of cases in which Algorithm 5.1.1 times out. This is a reflection of the double exponential complexity associated with Algorithm 5.1.1.

Unlike the case with subsumers, we observe that our implementation of Algorithm 5.1.2 for extracting weakest sufficient conditions rarely makes ANEMONE-subsumees more standalone complete. This may just be a reflection of the fact that subsumees of concept names in the ontologies of the dataset are not expressive enough to warrant the use of weakest sufficient conditions.

Neither ANEMONE-subsumers nor ANEMONE-subsumees are standalone subsumer and subsumee complete for $\mathcal{ALC}$ for $\mathcal{ALC}$ ontologies without cycles, yet we observe that in a significant number of instances, they can not be made more standalone complete by strongest necessary or weakest sufficient conditions.

In theory, ANEMONE-subsumers and ANEMONE-subsumees are neither contextually nor standalone complete for $\mathcal{ALC}$ ontologies without cycles, but in practice, they can be feasibly and easily extracted.
In practice, Algorithms 5.1.1 and 5.1.2 can sometimes be feasibly extracted given enough of a timeout.
Because SNCs and WSCs may be used to extract concepts that involve role restrictions (which ANEMONE-subsumers and subsumees can not), we expected SNCs/WSCs to make ANEMONE-subsumers/ANEMONE-subsumees more standalone complete in

practice; however, our results demonstrate that the SNCs/WSCs extracted by Algorithms 5.1.1 and 5.1.2 rarely do so.

7.6 Contribution Summary

The contributions of this chapter are as follows:

- A discussion of the possible impact of strongest necessary conditions and weakest sufficient conditions on agent communication in the ANEMONE system.
- A characterisation of completeness for subsumers and subsumees. We initially characterise completeness with respect to an ontology (contextual completeness), however, highlight that such notions of completeness are not sufficient for agent communication, thus prompting the characterisation of completeness (standalone completeness) without the background ontology.
- An evaluation of how frequently our algorithms for extracting strongest necessary conditions could enhance the standalone completeness of ANEMONE-subsumers, and similarly how frequently the weakest sufficient conditions could enhance the standalone completeness ANEMONE-subsumees. Our results suggest that our algorithm for extracting strongest necessary conditions, when successful, seems to make ANEMONE-subsumers more standalone complete about half the time, whereas our algorithm for extracting weakest sufficient conditions rarely make ANEMONE-subsumees more standalone complete.

Chapter 8

Finite Entailment Sets and Ontology Alignment

Up to this point in thesis, we have assumed the pre-existence of a common vocabulary between communication partners. Ontology matching is a common way for agents to establish a common vocabulary. In this chapter we investigate the usefulness of knowledge extraction techniques in evaluating a common vocabulary established via ontology matching.

Thus, this chapter tries to answer **RQ7** "*Can knowledge extraction techniques be applied to establishing a common vocabulary?*". The goal of ontology matching is to build an ontology alignment which is a set of mappings or correspondences between symbols in two different ontologies. The integrity of an alignment determines the accuracy and quality of knowledge shared between communicating partners.

We investigate *entailment sets* that can be built using knowledge extraction techniques and how entailment sets can be used to evaluate ontology alignments.

In Section 8.1, we provide a background on entailment sets and discuss design decisions to be taken into consideration when constructing entailment sets. Section 8.2 introduces the entailment sets and how they can be constructed. Finally in Section 8.3, we discuss developed entailment sets and their applications to evaluating ontology alignments.

8.1 Entailment Sets

We call the (infinite) set of all axioms α such that $O \models \alpha$ the theory of O (also called the *deductive closure* of O). Therefore, an ontology is a finite set of axioms that represents

an infinite theory of knowledge. A *finite entailment set* is a subset of the deductive closure of O. Finite entailment sets may be extracted for several reasons such as finding the semantic and syntactic differences and similarities between a pair of ontologies [GPS12, GPS11], and generating metrics for ontologies [Bai13]. Practical examples of entailment sets include the class hierarchy of an ontology, i.e., a set of atomic subsumption and equivalence class axioms over the set of concept names in an ontology. Class hierarchies have several uses, including constructing alignments for ontologies (we expand on this in Section 3.3.2). Uniform interpolants also form a category of finite entailment sets, specifically those that capture the deductive closure of an ontology over a specified subset signature of the ontology. An imperative limitation of the class hierarchy is that it fails to capture entailments involving complex concepts that include roles in their signature. In this chapter, we are interested in finite entailment sets for the TBox of an ontology. Specifically, we are interested in entailment sets that can also capture some information about entailments that contain complex concepts, and investigate whether the knowledge extraction tools discussed and developed in Chapter 5 may be used to construct entailment sets that prove useful to evaluating ontology alignments.

Bail [Bai13] outlines several design decisions to be taken into account when constructing a finite entailment-set. We discuss design decisions outlined by Bail [Bai13] that are relevant to the entailment sets developed in this chapter.

Design Decisions for Entailment sets

1. **Tautologies:** Tautologies are axioms that are vacuously entailed by an ontology, i.e., axioms that do not follow from any explicitly stated axioms in an ontology. For example, the axioms $\bot \sqsubseteq A, A \sqsubseteq A, A \sqsubseteq \top, \top(a)$ can be inferred from any ontology. Including tautologies in an entailment set increases the size of the entailment set without capturing any knowledge from O. We do not include tautologies in the entailment sets discussed in this chapter.

2. **Equivalence to $\top$ or $\bot$:** Let A be a concept name such that an ontology O entails $A \equiv \bot$. The decision to add $A \equiv \bot$ (or $A \sqsubseteq \bot$) affects the number of axioms that have to be added to an entailment set. If $A \equiv \bot$ or $A \sqsubseteq \bot$ is excluded, then for every concept name A', such that $A' \in sig(O)$ and $O \not\models A' \equiv \bot$, the axiom $A \sqsubseteq A'$ may need to be added for every such A'. Similarly the dual case holds if $O \models A \equiv \top$.

3. **Axioms and expression Types:** Entailments involving class expressions that include value restrictions are sometimes needed for an application. To capture a finite set of such entailments, prime implicates and prime implicants [Bie08] have been suggested as a method to construct entailment sets involving class expressions that include roles [Bai13]. Strongest necessary conditions in negated normal form that do not contain any conjunctions ($\sqcap$) can be regarded as variants of implicates that factor in a background ontology, i.e., subsumption is considered with respect to a background ontology. The notion of *strength* (in the sense for strongest necessary conditions) loosely corresponds with the notion of *primeness* (in the sense of prime implicates). For the application of ontology alignment, the background ontology plays a significant role in the interpretation of class expressions. We therefore propose strongest necessary conditions as opposed to prime implicates in the construction of entailment sets for the evaluation of ontology alignments.

In the next sections, we investigate using SNC(s) and WSC(s) as alternatives to prime implicates to be used in constructing finite entailment sets that involve non-atomic concept expressions. We explore whether entailment sets constructed using SNC(s) and WSC(s) are useful in evaluating ontology alignments, and can thus be useful in the process of establishing a common vocabulary between agents.

8.2 Developed entailment sets

Unless otherwise stated, we only discuss entailment sets involving $\mathcal{ALC}$ ontologies or less expressive ontologies. Let O be an $\mathcal{ALC}$ ontology or less expressive ontology. We define three entailment sets that may be constructed over O using SNC(s), WSC(s), domain extraction, and range extraction, namely: (1) Ascending Entailment Set, (2) Descending Entailment Set, and (3) Relation Entailment Set.

8.2.1 Ascending Entailment Sets

The ascending entailment set is constructed by extracting SNCs for all concept names in the ontology and is described in Definition 47.

Definition 47 (Ascending Entailment Set)**.** *Let Σ be $sig(O)$ for an ontology O. The* ascending entailment set *of O denoted AES_O is the set of all axioms $A \oplus SNC(A; O; \Sigma \setminus$*

$\{A\}$) *such that* $A \in \Sigma$, $\oplus = \equiv$ *if* $O \models A \equiv SNC(A;O;\Sigma \setminus \{A\})$ *and* $\oplus = \sqsubseteq$ *if* $O \not\models A \equiv SNC(A;O;\Sigma \setminus \{A\})$.

Following Definition 47 the AES of an ontology is constructed using SNCs. Our

ALGORITHM 8.2.1

Input:

1. An $\mathcal{ALC}$ ontology O.
2. A signature Σ such that $\Sigma \subseteq sig(O)$.

Output: The ascending entailment set AES_O of O.

---

1. Initialise AES_O: $AES_O \leftarrow \emptyset$.
2. **for** each concept name $A \in \Sigma$ **do:**

   Extract $SNC(A;O;\Sigma \setminus \{A\})$.

   if $O \models A \equiv SNC(A;O;\Sigma \setminus \{A\})$ **do:**

   $AES_O \leftarrow AES_O \cup \{A \equiv SNC(A;O;\Sigma \setminus \{A\})\}$

   else:

   $AES_O \leftarrow AES_O \cup \{A \sqsubseteq SNC(A;O;\Sigma \setminus \{A\})\}$

   endfor
3. return AES_O.

method for constructing ascending entailment sets is specified in Algorithm 8.2.1. To illustrate the procedure of Algorithm 8.2.1, consider the ontology O consisting of the following axioms:

$$\{A \equiv B \sqcup C, C \sqsubseteq \neg B, D \sqsubseteq A\}$$

We have that:

- $SNC(A;O;\Sigma \setminus \{A\}) = B \sqcup C$.
- $SNC(B;O;\Sigma \setminus \{B\}) = A \sqcap \neg C$
- $SNC(C;O;\Sigma \setminus \{C\}) = A \sqcap \neg B$

- $SNC(D;O;\Sigma \setminus \{D\}) = A$

We have that A, B and C are equivalent to their respective strongest necessary conditions under O, thus, the ascending entailment set AES_O of O is $\{A \equiv B \sqcup C, B \equiv A \sqcap \neg C, C \equiv A \sqcap \neg B, D \sqsubseteq A\}$, and in this case, AES_O is indeed equivalent to O. In the context of $\mathcal{ALC}$ ontologies (or less expressive ontologies), ascending entailment sets can thus be regarded as summaries of ontologies that capture the subsumers of concept names. Ascending entailment sets can thus have applications to the same domains other entailment sets such as class hierarchies, including finding and detecting the semantic and syntactic differences between a pair of ontologies, ontology summarisation, and ontology alignment. In this chapter we are specifically interested in applications to ontology alignment, and in Section 8.3.1, we expand on how ascending entailment sets can be applied to ontology alignments.

8.2.2 Descending Entailment Sets

Descending entailment sets are the duals of AESs and are constructed by extracting the WSC for each concept name in an ontology and constructing the corresponding appropriate axiom.

Definition 48 (Descending Entailment Set)**.** *Let Σ be $sig(O)$ for an ontology O. The* descending entailment set *of O denoted DES_O is the set of all axioms $WSC(A;O;\Sigma \setminus \{A\}) \oplus A$ such that $A \in \Sigma$, $\oplus = \equiv$ if $O \models A \equiv WSC(A;O;\Sigma \setminus \{A\})$ and $\oplus = \sqsubseteq$ if $O \not\models A \equiv WSC(A;O;\Sigma \setminus \{A\})$.*

Our method for constructing descending entailment sets is specified in Algorithm 8.2.2. To illustrate the procedure of Algorithm 8.2.2, consider the ontology O consisting of the following axioms:

$$\{A \equiv B \sqcup C, C \sqsubseteq \neg B, D \sqsubseteq A\}$$

In O, there is no concept name that is equivalent to $\top$ or $\bot$, as such, $\Sigma = sig(O)$. Thus we have that:

- $WSC(A;O;\Sigma \setminus \{A\}) = B \sqcup C$.
- $WSC(B;O;\Sigma \setminus \{B\}) = A \sqcap \neg C$
- $WSC(C;O;\Sigma \setminus \{C\}) = A \sqcap \neg B$

ALGORITHM 8.2.2

Input:

1. An $\mathcal{ALC}$ ontology O.
2. A signature Σ such that $\Sigma \subseteq sig(O)$.

Output: The descending entailment set DES_O of O.

---

1. Initialise DES_O: $DES_O \leftarrow \emptyset$.
2. **for** each concept name $A \in \Sigma$ **do:**

   Extract $WSC(A; O; \Sigma \setminus \{A\})$.

   if $O \models A \equiv WSC(A; O; \Sigma \setminus \{A\})$ **do:**

   $DES_O \leftarrow DES_O \cup \{WSC(A; O; \Sigma \setminus \{A\}) \equiv A\}$

   else:

   $DES_O \leftarrow DES_O \cup \{WSC(A; O; \Sigma \setminus \{A\}) \sqsubseteq A\}$

   endfor
3. return DES_O.

- $WSC(D; O; \Sigma \setminus \{D\}) = \bot$ (*this is a tautology and is thus omitted*)

We have that A, B and C are equivalent to their respective weakest sufficient conditions under O, thus, the descending entailment set of O is $\{A \equiv B \sqcup C, B \equiv A \sqcap \neg C, C \equiv A \sqcap \neg B\}$, which is also equivalent to O. Descending entailment sets capture *specific* information about concept names that usually can not be captured by ascending entailment sets [1].

8.2.3 Relation Entailment Sets

A relation entailment set is a set of axioms that should contain information about the roles in an ontology. Since we focus on $\mathcal{ALC}$ (or less expressive) ontologies, the relation entailment set is induced by extracting the domain and range axioms of all the roles in an ontology. Let O be an ontology and Σ be $sig(O)$ without concept names

[1] The exception here is equivalent concepts.

that are equivalent to $\top$ or $\bot$ such that for each concept name $A' \in \Sigma$, we have that $O \not\models A' \sqsubseteq \bot$ and $O \not\models \top \sqsubseteq A'$. For each role $r \in sig(O)$, an axiom α representing the strongest domain of r with respect to Σ is added to the relation entailment set of O, similarly, an axiom β representing the strongest range of of r is added to the relation entailment set of O

Definition 49 (Relation Entailment Set)**.** *Let Σ be $sig(O)$ for an ontology O. The* relation entailment set *of O denoted RES_O is the set of all axioms of the form $\exists r.\top dom_s(r;O;\Sigma)$ and $\top \sqsubseteq \forall r.ran_s(r;O;\Sigma)$.*

ALGORITHM 8.2.3

Input:

1. An $\mathcal{ALC}$ ontology O.
2. A signature Σ such that $\Sigma \subseteq sig(O)$.

Output: The relation entailment set DES_O of O.

---

1. Initialise RES_O: $RES_O \leftarrow \emptyset$.
2. **for** each role $r \in \Sigma$ **do:**

   Extract $dom_s(r;O;\Sigma \setminus \{r\})$.

   Extract $ran_s(r;O;\Sigma \setminus \{r\})$.

   $RES_O \leftarrow RES_O \cup \{\exists r.\top \sqsubseteq dom_s(r;O;\Sigma \setminus \{r\})\}$

   $RES_O \leftarrow RES_O \cup \{\top \sqsubseteq \forall r.ran_s(r;O;\Sigma \setminus \{r\})\}$

   endfor
3. return DES_O.

Our method for constructing descending entailment sets is specified in Algorithm 8.2.2. To illustrate the procedure of Algorithm 8.2.2, consider the ontology O consisting of the following axioms:

$$\{A \sqsubseteq B \sqcap C, C \sqsubseteq \neg B, D \sqsubseteq E, \exists r.\top \sqsubseteq D \sqcup A, \top \sqsubseteq \forall p.B\}$$

We have that $O \models A \sqsubseteq \bot$, therefore, $\Sigma = \{B, C, D, E, r, p\}$. Thus we have that:

- $dom_s(r;O;\Sigma \setminus \{r\}) = D$
- $ran_s(r;O;\Sigma \setminus \{r\}) = \top$ (*this is a tautology and is thus omitted*)
- $dom_s(p;O;\Sigma \setminus \{p\}) = \top$ (*this is a tautology and is thus omitted*)
- $ran_s(p;O;\Sigma \setminus \{p\}) = B$

The relation entailment set of O is $\{\top \sqsubseteq \forall p.B, \exists r.\top \sqsubseteq D\}$.

Similar to ascending and descending entailment sets, relation entailment sets factor in tautologies by placing similar constraints on Σ and ensuring that each extracted domain and range is not equivalent to $\top$ or $\bot$.

8.2.4 Impact of Defined Concepts and Primitive Signatures

If a suitable primitive signature can be identified for the ontology of interest, it may be more beneficial to use the primitive signature as the basis of extracting the ascending or descending entailment set as opposed to using the signature of the entire ontology. Let Σ_P denote a suitable primitive signature for O, and let Σ be the subset of $sig(O)$ that does not contain any concept names A such that $O \models \top \sqsubseteq A$ or $O \models A \sqsubseteq \bot$. When extracting the ascending or descending entailment sets for O, both the SNCs and WSCs for any concept name A in Σ should be extracted with respect to $\Sigma \cap \Sigma_P \setminus \{A\}$ in order to make sure the entailment set is as informative as possible. For example, let O be $\{A_1 \equiv A_2, A_2 \equiv A_3 \sqcup A_4, A_3 \equiv \exists r.C_1\}$; the primitive signature of O is $\{A_4, C_1, r\}$. The ascending entailment set AES_O for O without taking the primitive signature into consideration is $\{A_1 \equiv A_2, A_2 \equiv A_1, A_3 \equiv \exists r.C_1\}$ while the ascending entailment set AES'_O for O that takes the primitive signature into consideration is $\{A_1 \equiv \exists r.C_1 \sqcup A_4, A_2 \equiv \exists r.C_1 \sqcup A_4, A_3 \equiv \exists r.C_1\}$. Observe that AES_O fails to capture that A_1 and A_2 are equivalent to $\exists r.C_1 \sqcup A_4$, unlike AES'_O, which also captures that A_1 and A_2 are equivalent to $A_3 \sqcup A_4$.

In general, we posit that primitive signatures should be taken into consideration when designing entailment sets that involve complex concepts.

A related problem is the information that may be included in GCIs of the ontology. Thus any application built on ascending or descending entailment sets inherits the limitation that any entailments involving GCIs will most likely be missing. For example, let $O = \{A_1 \equiv \exists r.C_1, A_1 \sqsubseteq B_1, \exists r.B_2 \sqsubseteq \exists p.C_2\}$, O entails the following GCIs (1) $\exists r.C_1 \sqsubseteq B_1$ (2) $\exists r.B_2 \sqsubseteq \exists p.C_2$; a valid ascending entailment set for O is $\{A_1 \equiv \exists r.C_1 \sqcap B_1\}$ and a

valid descending entailment set for O is $\{A_1 \sqsubseteq B_1\}$, both of which do not capture the GCI $\exists r.B_2 \sqsubseteq \exists p.C_2$.

8.3 Application to Ontology Alignment

There are several methods to evaluate the accuracy of alignments [SJRG17b]. However, the majority of these methods rely on entailments sets such as the class hierarchy of input ontologies [SJRG17b]. As a result, the scope of evaluation is often limited to atomic concepts and simple concept expressions.

In this section, we discuss how the entailment sets developed in this chapter can be exploited to expand the scope of potentially invalid mappings that are detected in the process of evaluating the accuracy of computed alignments. Section 3.3.2 provides a background on existing alignment approaches. In Section 8.3.1, we present our proposal for using ascending and descending entailment sets to extend the evaluation of alignments.

8.3.1 A Proposal for extending detection of conservativity violations

Detecting and generating repairs to an alignment can be seen as a form of *debugging* the alignment. As discussed earlier (Section 3.3.2), some existing approaches employ the use of some structure (such as an index or a graph) based on the class hierarchy of the input ontologies to reflect the semantic structure of the input ontologies and detect conservativity violations. However, the conservativity violations detected are only focused on atomic subsumptions, meaning that diagnoses involving complex concepts can be potentially missed. To capture conservativity violations involving complex concepts, some alignment approaches include an interaction component that enable a human user (preferably a domain expert) to contribute to the diagnostic process. ContentMap [JRGHL09] for example, allows users to customise and evaluate entailments that may include atomic subsumptions, disjointness, atomic universal restrictions[2], atomic existential restrictions [3], and role hierarchies.

[2] I.e., restrictions of the form $\forall r.A$, where A is always an atomic concept.
[3] I.e., restrictions of the form $\exists r.A$, where A is always an atomic concept.

Computing Deductive Differences using Uniform Interpolation

Uniform interpolation has been demonstrated [ZAS+19] to be useful in computing an approximation of the deductive difference between two ontologies that involve non-atomic concepts in DLs as expressive as $\mathcal{ALC}$. We provide an adaptation of the deductive difference algorithm proposed by Zhao et al. [ZAS+19] (originally called *uniform interpolation difference*) in Algorithm 8.3.1. We briefly explain the functioning of Algorithm 8.3.1. Let O and O' be ontologies. The uniform interpolation difference between O and O' with respect to O can be derived by extracting the uniform interpolant $\mathcal{V}$ of O' for $\Sigma = sig(O) \cap sig(O')$. A *UI-witness* is any axiom $\alpha \in \mathcal{V}$ such that $O \not\models \alpha$. Thus, the set of all UI-witnesses can be considered as a finite representation of the deductive difference between O and O' with respect to O.

Because uniform interpolation is not restricted to atomic subsumptions, we have the advantage of capturing deductive differences that involve complex concepts.

ALGORITHM 8.3.1

Input: Two $\mathcal{ALC}$ ontologies O and O'.
Output: The set of all UI-witnesses between O and O' that provide a representation of the deductive difference between O and O' with respect to O denoted $UI-diff(O,O')$.

---

1. Initialise Σ: $\Sigma \leftarrow sig(O) \cap sig(O')$
2. Extract a uniform interpolant $\mathcal{V}$ of O' for Σ
3. Initialise $UI-diff(O,O')$: $UI-diff(O,O') \leftarrow \emptyset$
4. **for** each axiom $\alpha \in \mathcal{V}$ **do:**
   if $O \not\models \alpha$ **do:**
   $UI-diff(O,O') \leftarrow UI-diff(O,O') \cup \alpha$
   endif
   endfor
5. return $UI-diff(O,O')$.

We generalise the Definition of conservativity violations to include complex concept inclusions in Definition 50.

Definition 50 (General Conservativity Violation)**.** *Let O_1 and O_2 be ontologies mapped by a coherent alignment $\mathcal{A}$, let $O_i \in \{O_1, O_2\}$. We define two sets of violations of $O_1 \cup O_2 \cup \mathcal{A}$ with respect to O_i:*

- general subsumption violations*, denoted $subViol_G(O_i, O_1 \cup O_2 \cup \mathcal{A})$, as the set of axioms $A \sqsubseteq C$ satisfying: (i) $UI-diff(O_i, O_1 \cup O_2 \cup \mathcal{A}) \models A \sqsubseteq C$*[4]*, (ii) $O_i \not\models C \sqsubseteq A$, and (iii) there is no D such that $O_i \not\models D \equiv \top$, $O_i \not\models D \equiv \bot$, $O_i \models D \sqsubseteq A$, and $O_i \models D \sqsubseteq C$, where A and D are atomic, $A \in sig(O_i)$, C is possibly complex, $sig(C) \subseteq sig(O_i) \setminus \{A\}$ and $D \neq A$.*

- general equivalence violations*, denoted as $eqViol_G(O_i, O_1 \cup O_2 \cup \mathcal{A})$, as the set of $A \equiv C$ axioms satisfying (i) $O_1 \cup O_2 \cup \mathcal{A} \models A \equiv C$, (ii) $UI-diff(O_i, O_1 \cup O_2 \cup \mathcal{A}) \models A \sqsubseteq C$ or $UI-diff(O_i, O_1 \cup O_2 \cup \mathcal{A}) \models C \sqsubseteq A$, where A is atomic, $A \in sig(O_i)$, C is possibly complex, and $sig(C) \subseteq sig(O_i) \setminus \{A\}$.*

Condition (ii) and (iii) of *general subsumption violations* in Definition 50 are motivated by the *assumption of disjointness*[Sch05] (similar to the case in Definition 31). We *assume* that if there is a concept name D that is included in both A and C, then A and C intersect on some interpretation and that D is a witness concept of this intersection. If there is no witness to support that A and C intersect on some interpretation, then we assume that A and C are disjoint, per the assumption of disjointness [Sch05]. Condition (iii) of *general equivalence violations* in Definition 50, highlights that a novel subsumption ($C \sqsubseteq A$ or $A \sqsubseteq C$) has been induced by the alignment which is not allowed (recall from Section 3.3.2 that conservativity violations in an alignment indicate that the alignment introduces new semantic relationships in the vocabulary of one of the input ontologies).

The main difference between Definition 50 and Definition 31 is that Definition 50 also includes non-atomic concepts for general subsumption and equivalence violations. Definition 50 includes subsumptions of the form $A \sqsubseteq C$ where A is atomic, and C may be a compound concept with roles in its signature when evaluating subsumptions for conservativity violations. Similarly, Definition 50 includes equivalences of the form $A \equiv C$ where A is atomic, and C may be a compound concept with roles in its signature when evaluating equivalences for conservativity violations unlike Definition 31 which

[4]Observe that this means $O_i \not\models A \sqsubseteq C$ following the definition of deductive difference.

strictly focuses on atomic concepts. Thus the set of conservativity violations captured by Definition 31 can be considered a subset of the convservativity violations captured by Definition 50.

The deductive difference computed by Algorithm 8.3.1 is only one of the components we use to detect the general conservativity violations specified in Definition 50. This is because some extra processing is required to evaluate each concept for conservativity violations.

In the next Section, we describe how the deductive difference computed using Algorithm 8.3.1 can be used in conjunction with ascending and descending entailment sets to detect general conservativity violations.

Detecting General Conservativity Violations using Uniform Interpolation and Entailment sets

In order to detect general conservativity violations, we use the deductive difference which can be computed using Algorithm 8.3.1. However, some extra processing is required to evaluate each concept for general subsumption and equivalence violations. Let $O_1 = \{A_1 \sqsubseteq B_1 \sqcup C_1, A_2 \sqsubseteq \neg D_1, A_3 \sqsubseteq \exists r.C_2\}$ and $O_2 = \{A'_1 \sqsubseteq \exists r'.B'_1, \exists r'.A'_2 \sqsubseteq \neg A'_1\}$ such that every symbol $S \in O$ is mapped to its dual symbol $S' \in O'$ O_1 and O_2. We have that $UI-diff(O_1, O_2) = \{A_1 \sqsubseteq \exists r.B_1, \exists r.A_2 \sqsubseteq \neg A_1\}$. However, $UI-diff(O_1, O_2)$ alone does not state what the general subsumption or equivalence violations are and whether there are any such violations.

General conservativity violations can be detected using a combination of Algorithm 8.3.1 (which outputs a set of UI-witnesses given two ontologies), ascending and descending entailment sets. We provide an algorithm for detecting general subsumption conservativity violations (as in Definition 50) in Algorithm 8.3.2. We briefly explain the functioning of Algorithm 8.3.2. Algorithm 8.3.2 checks for general subsumption violations by iterating over all subsumptions $A \sqsubseteq C \in AES_{UI-diff(O,\mathcal{U})}$ where $\mathcal{U}$ denotes $O \cup O' \cup \mathcal{A}$. For each subsumption $A \sqsubseteq C$, the requirements of Definition 50 are validated as follows: if A does not have a subsumed concept name D with respect to O then we assume that both A and C do not have a common subsumed-concept with respect to O, and thus consider $A \sqsubseteq C$ to be a general subsumption violation. Observe that because ascending entailment sets are used in Algorithm 8.3.2, the checks made for subsumers are minimal: for every axiom $A \sqsubseteq C \in AES_{UI-diff(O,\mathcal{U})}$, C is a strongest necessary condition of A for a signature $\Sigma \setminus \{A\}$ such that $\Sigma \subseteq UI-diff(O, \mathcal{U})$ and for all concept names $A' \in \Sigma$, $UI-diff(O, \mathcal{U}) \not\models A' \sqsubseteq \bot$ and $UI-diff(O, \mathcal{U}) \not\models \top \sqsubseteq A'$

ALGORITHM 8.3.2

Input:

1. Two $\mathcal{ALC}$(or less expressive) ontologies O and O' mapped via $\mathcal{A}$.
2. The UI difference $UI-diff(O,\mathcal{U})$ of O and $\mathcal{U}$.
3. The ascending entailment set AES_O of O.
4. The descending entailment set DES_O of O.
5. The ascending entailment set $AES_{UI-diff(O,\mathcal{U})}$ of $UI-diff(O,\mathcal{U})$.

Output: $subViol_G(O,\mathcal{U})$.
Where $\mathcal{U}$ denotes $O \cup O' \cup \mathcal{A}$.

---

1. $subViol_G(O,\mathcal{U}) \leftarrow \emptyset$
2. $\Sigma \leftarrow sig(O) \cap sig(\mathcal{A})$
3. **for** each axiom of the form $A \sqsubseteq C \in AES_{UI-diff(O,\mathcal{U})}$ **do:**
   if $O \not\models C \sqsubseteq A$ **do:**
   if there exists no axiom of the form $D \sqsubseteq A$ in DES_O **do:**
   $subViol_G(O,\mathcal{U}) \leftarrow subViol_G(O,\mathcal{U}) \cup \{A \sqsubseteq C\}$.
   endif
   if there exists some axiom $D \sqsubseteq A$ in DES_O **do:**
   if there exists no concept-name B such that $O \models B \sqsubseteq A$ and $O \models B \sqsubseteq C$ **do:**
   $subViol_G(O,\mathcal{U}) \leftarrow subViol_G(O,\mathcal{U}) \cup \{A \sqsubseteq C\}$.
   endif
   endif
   endfor
4. return $subViol_G(O,\mathcal{U})$.

thus for every necessary condition C' A may have with respect to Σ we can be confident that $UI-diff(O,\mathcal{U}) \models C \sqsubseteq C'$. As an example, consider the ontology O consisting of the following axioms:

$$O = \{A_1 \sqsubseteq A_2, A_3 \sqsubseteq \exists p.B_1, B_3 \sqsubseteq A_5, B_2 \sqsubseteq A_4\}$$

and the ontology O' consisting of the following axioms:

$$O' = \{A'_6 \sqsubseteq \neg A'_5, A'_3 \sqsubseteq B'_2, A'_3 \sqsubseteq A'_4 \sqcap A'_6, A'_5 \sqsubseteq \exists p'.A'_1\}$$

such that every symbol $S \in O$ is mapped to a dual symbol $S' \in O'$ and we have that $UI-diff(O,\mathcal{U})$ consists of the following axioms:

$$\{A_3 \sqsubseteq A_4 \sqcap \neg A_5, A_3 \sqsubseteq B_2, A_5 \sqsubseteq \exists p.A_1\}$$

Observe that for A_3 the following subsumers exist A_4, B_2, $A_4 \sqcap B_2$ $A_4 \sqcap \neg A_5$, $A_4 \sqcap \exists p.A_1$, and more. However, the strongest necessary condition is $A_4 \sqcap \neg A_5 \sqcap B_2$ and thus only one evaluation has to be made. Similarly we could naively assume that the only subsumer of A_5 is $\exists p.A_1$; however, A_5 also has the subsumers $\neg A_3$ and $\neg A_4$, which will all follow from the SNC of A_5. In a larger ontology, there may be even more subsumers to evaluate depending on the concept name.

We briefly explain the functioning of Algorithm 8.3.3. Algorithm 8.3.3 checks for general equivalence violations by iterating over all axioms $A \equiv C \in AES_{O,\mathcal{U}}$. For each axiom $A \equiv C$, the requirements of Definition 50 are validated as follows: if there exists a concept D that is equivalent to A with respect to $UI-diff(O,\mathcal{U})$, then we check to see if $\mathcal{U} \models C \equiv D$ holds and in the case that it holds, then $C \equiv D$ is considered a general equivalence violation. Otherwise, we check if there exists a subsumer or subsumed-concept D of A with respect to $AES_{UI-diff(O,\mathcal{U})}$, and if so, if $\mathcal{U} \models C \equiv D$ holds, then $C \equiv D$ is considered a general equivalence violation.

As an example, consider the ontology O consisting of the axiom:

$$O = \{A_1 \sqsubseteq \exists r.C_1\}$$

and the ontology O' consisting of the axiom:

$$O' = \{A'_1 \equiv \exists r'.C'_1\}$$

such that every symbol $S \in O$ is mapped to a dual symbol $S' \in O'$ and we have that

ALGORITHM 8.3.3

Input:

1. Two $\mathcal{ALC}$(or less expressive) ontologies O and O' mapped via $\mathcal{A}$.
2. The UI difference $UI-diff(O,\mathcal{U})$ of O and $\mathcal{U}$.
3. The ascending entailment set $AES_{\mathcal{U}}$ of O, $\mathcal{U}$.
4. The ascending entailment set $AES_{UI-diff(O,\mathcal{U})}$ of $UI-diff(O,\mathcal{U})$.
5. The descending entailment set $DES_{UI-diff(O,\mathcal{U})}$ of $UI-diff(O,\mathcal{U})$.

Output: $eqViol_G(O, O\cup O'\cup\mathcal{A})$.
Where $\mathcal{U}$ denotes $O\cup O'\cup\mathcal{A}$.

---

1. $eqViol_G(O,\mathcal{U}) \leftarrow \emptyset$
2. $\Sigma \leftarrow sig(O)\cap sig(\mathcal{A})$
3. **for** each axiom $A\equiv C\in AES_{\mathcal{U}}$ **do:**
   if there exists some axiom $A\equiv D$ in $AES_{UI-diff(O,\mathcal{U})}$ **do:**
   if $\mathcal{U}\models C\equiv D$ **do:**
   $eqViol_G(O,\mathcal{U}) \leftarrow eqViol_G(O,\mathcal{U})\cup\{A\equiv C\}$.
   endif
   elif there exists some axiom $A\sqsubseteq D$ in $AES_{UI-diff(O,\mathcal{U})}$ **or**
   there exists some axiom $D\sqsubseteq A\in DES_{UI-diff(O,\mathcal{U})}$ **do:**
   if $\mathcal{U}\models C\equiv D$ **do:**
   $eqViol_G(O,\mathcal{U}) \leftarrow eqViol_G(O,\mathcal{U})\cup\{A\equiv C\}$.
   endif
   endfor
4. return $eqViol_G(O,\mathcal{U})$.

$UI-diff(O,\mathcal{U})$ consists of the axiom:

$$A_1 \equiv \exists r.C_1$$

Thus, $A_1 \equiv \exists r.C_1$ is a general equivalence violation.

Impact of GCIs Algorithm 8.3.2 and Algorithm 8.3.3 are limited in their ability to detect general conservativity violations involving entailments relating to GCIs. As an example, consider the ontology O consisting of the following axioms:

$$O = \{A_1 \sqsubseteq \exists r.C, A_2 \sqsubseteq \exists p.\top, A_3 \sqsubseteq B_2\}$$

and the ontology O' consisting of the following axioms:

$$O' = \{A'_4 \equiv \exists r'.C', A'_3 \sqsubseteq \exists p'.B'_2\}$$

such that every symbol $S \in O$ is mapped to a dual symbol $S' \in O'$ and we have that $UI-diff(O,\mathcal{U})$ (where $\mathcal{U}$ denotes $O \cup O' \cup \mathcal{A}$) consists of the following axioms:

$$\{\exists r.C \sqsubseteq \exists p.B_2\}$$

$\exists r.C \sqsubseteq \exists p.B_2$ is a subsumption violation according to Definition 50 but a violation that can not be detected by Algorithm 8.3.3. However, since the deductive difference returned by Algorithm 8.3.1 is finite, the GCIs in the result may be used as a foundation to evaluate as potential general conservativity violations involving GCIs.

Experimental Evaluation

To evaluate Algorithms 8.3.2 and 8.3.3, we used an implementation of LogMap 2 [JRGZH12] publicly available at `http://krrwebtools.cs.ox.ac.uk/logmap/` to generate alignements. LogMap 2 automatically applies repairs to the alignments generated, thus our evaluation aims to report on any additional conservativity violations not detected by LogMap, specifically general conservativity violations.

To construct a corpus for evaluation, a set of pairs of ontologies to be aligned is required. In constructing a corpus, the ontologies in the Ontology Alignment Evaluation Initiative 2021 track [ANPAA+21] were considered. However, most of the ontologies in this corpus are either more expressive than $\mathcal{ALC}$ (such as the ontologies in the conference track), or too large for the implementation of LETHE used (such as the

ontologies in large biomedical ontologies track). Of all the tracks, we find the most compatible for evaluation is the anatomy track which only consists of the Adult Mouse Anatomy ontology and the NCI Thesaurus describing the human anatomy. This only gives us one pair of ontologies for evaluation.

The ontologies in the main corpus were all retrieved from the BioPortal ontology repository [WNS+11] which sometimes contains several versions of the same ontology. In order to augment the corpus, we used ontologies from the main corpus that had several versions available on the BioPortal ontology repository. Two separate versions of the same ontology provide an additional pair of ontologies for evaluation. For each ontology O in the main corpus, the following criteria were used to assess whether it would be used in the evaluation:

1. O had more than one version on the BioPortal repository.
2. The other version O' of O to be used from the BioPortal repository was not more expressive than $\mathcal{ALCH}$ (for $\mathcal{ALCH}$ ontologies, we deleted the role hierarchy).
3. The mappings generated by the LogMap tool could be parsed using the OWL-API.
4. Some evidence of a significant difference existed in the ontologies such as a significant difference in the number concept names, axioms, or roles.
5. The ontology could be classified using the Hermit reasoner [GHM+14] (under a time-out of one-hour).
6. The ontology contained no unsatisfiable concept names.
7. The number of concept names in O was less than 1,000. This is to reduce the risk of the LETHE implementation timing out whilst attempting to forget a role (as role-forgetting is compulsory for Algorithms 8.3.2 and 8.3.3) or throwing an exception.

The resulting dataset is provided in Table 8.1.

For each ontology, we extracted alignments using the LogMap tool. For each alignment, we only considered mappings that had a confidence value of greater than 0.5.

Ontology ID	DL	Class Count	Role Count	Axiom Count	Upload Date	Mapped Symbols
HUMAN	$\mathcal{ALC}$	3,304	2	11,544	13-02-2006	1350
MOUSE	$\mathcal{ALE}$	2,744	3	4,838	18-01-2007	1365
RXNO	$\mathcal{ALC}$	1074	14	1,735	16-12-2021	17
RXNO'	$\mathcal{ALCH}$	901	14	1,472	29-01-2021	15
ISSVA	$\mathcal{ALC}$	194	1	259	29-04-2021	195
ISSVA'	$\mathcal{ALCH}$	202	3	337	03-02-2021	201
XEO	$\mathcal{ALE}$	146	4	343	25-06-2014	21
XEO'	$\mathcal{ALE}$	145	4	243	03-04-2013	21

Table 8.1: Ontology alignment Corpus. HUMAN represents the NCI Thesaurus ontology describing the human anatomy from the anatomy track of the ontology alignment evaluation initiative. MOUSE represents the Adult Mouse Anatomy ontology from the anatomy track of the ontology alignment evaluation initiative. Upload date represents the upload date found in the corresponding ontology as an annotation (for the HUMAN and MOUSE case) or for the upload date as described on the BioPortal repository.

Ontology ID	SubViol	EqViol
HUMAN (O_1)	0	0
MOUSE (O_2)	0	0
RXNO (O_1)	0	11
RXNO' (O_2)	0	0
ISSVA (O_1)	0	3
ISSVA' (O_2)	0	0
XEO (O_1)	0	19
XEO' (O_2)	0	0

Table 8.2: Detected general violations

We ran our implementation of Algorithm 8.3.2 to detect general subsumption violations with respect to O and O'. The results are displayed in Table 8.2. Our implementation of Algorithm 8.3.2 only managed to detect general subsumption violations for two cases: (1) the HUMAN/MOUSE case (2) the ISSVA case, this can be observed in the columns titled 'SubViol (with respect to O_1' and 'SubViol (with respect to O_2)'. In the HUMAN/MOUSE case, there were no general subsumption or equivalence violations detected. For the ISSVA case, no general subsumption violations were found.

We also ran our implementation of Algorithm 8.3.3 to detect general equivalence violations with respect to O and O'. The results are displayed in Table 8.2. Our implementation of Algorithm 8.3.3 only managed to detect general equivalence violations

for three cases: (1) the RXNO case, (2) the ISSVA case, (3) the XEO case, this can be observed in the columns titled 'EqViol (with respect to O_1' and 'EqViol (with respect to O_2)'. There no general equivalence violations found for the HUMAN/MOUSE case possibly because the HUMAN and MOUSE ontologies have no defined concept names. In all cases, no general subsumption violations could be found.

While minimal these results suggest that our methods proposed for detecting general conservativity violations are feasible in some cases. In practice, we expect that the double exponential complexity associated with uniform interpolation and our algorithms for extracting strongest necessary and weakest sufficient conditions would be the main factor that may impede the feasibility of our methods.

8.4 Contribution Summary

The primary contribution of this Chapter are as follows:

- We have developed a characterisation of conservativity violations, namely *general conservativity violations* that aim to capture non-atomic/complex conservativity violations that may exist in ontology alignments.

- We have developed algorithms based on uniform interpolation, strongest necessary conditions, weakest sufficient conditions, strongest domains, and strongest ranges to detect general conservativity violations. We have minimally evaluated these algorithms on a small dataset and demonstrated that they do indeed capture general conservativity violations in real world ontologies.

Chapter 9

Conclusions and Summary

The primary aim of this thesis (**RQ1**) was to explore how knowledge extraction techniques could be applied to facilitate communication among agents that only have a subset of their respective vocabularies in common. We have studied existing knowledge extraction tools with existing theoretical specifications including uniform interpolation, module extraction, strongest necessary conditions, and weakest sufficient conditions, thus addressing **RQ1** "*What are the various knowledge extraction techniques in $\mathcal{ALC}$ and which of them are relevant to the problem of conveying words or expressions with respect to a restricted vocabulary?*". We are interested in *empirically* evaluating the knowledge extraction tools studied in this thesis. As such, we focus on knowledge extraction tools with existing implementations. Uniform interpolation and module extraction have implementations for description logics, but are however not suited for some agent communication tasks such as sharing concept descriptions. While strongest necessary conditions and weakest sufficient conditions have been demonstrated to be applicable for knowledge sharing among agents in propositional logic [Lin01], first order logic [DLS01], and second order logic, we find that studies in description logics are lacking. In Chapter 5, we fill the knowledge gap (i.e., the lack of any study of strongest necessary conditions and weakest sufficient conditions) by studying strongest necessary conditions and weakest sufficient conditions and creating adaptations in the description logic $\mathcal{ALC}$. Furthermore, we propose and implement algorithms for extracting strongest necessary conditions and weakest sufficient conditions. In Chapter 5 we propose and investigate novel knowledge extraction methods (and their computation) for agent communication, namely, compiled superconcepts, compiled subconcepts, strongest domain, and strongest range. In Chapter 5, we also explore the

application of strongest necessary and weakest sufficient conditions to extracting definitions for defined concept names in $\mathcal{ALC}$ ontologies thus partially exploring **RQ5** "*What is the most precise way to convey a word or expression with respect to a subset vocabulary of an ontology?*". As suggested in the introduction and background, when a concept-name A is definable with respect to a subset vocabulary, then a *definition* of that concept-name is the maximal description of A with respect to the ontology of interest. In Chapter 5 we demonstrate that strongest necessary conditions may be used to extract definitions of concept-names with respect to subset vocabularies. However, in the case when a concept-name is not definable, a maximal description of A with respect to a subset vocabulary and ontology O may be constructed using the strongest necessary and weakest sufficient conditions of A. Chapter 5 goes on to explore maximal descriptions of concepts that do not rely on background information from the ontology of interest; we define compiled superconcepts and compiled subconcepts to capture such notions and specify an algorithm designed to extract approximate forms of these notions.

Chapter 6 empirically evaluates the algorithms developed for extracting strongest necessary conditions and compiled superconcepts. Chapter 6 continues the exploration of **RQ5** and also explores **RQ2** "*How do the relevant knowledge extraction techniques perform in practice on real-world ontologies?*" by investigating how Algorithm 5.1.1 for extracting strongest necessary conditions performs in practice for extracting definitions. We find that in practice, our algorithm for extracting strongest necessary conditions successfully extracts definitions in about 99% of cases for concept names with respect to the entire signature of the ontology; however, our algorithms for extracting compiled superconcepts and compiled subconcepts rarely successfully terminate, and even when they do, output a result that may not be feasible for use, thus we deem our algorithms unusable due to their poor design which includes a saturation element. Chapter 6 also explores **RQ3** "*What are the factors that affect the performance of the relevant knowledge extraction techniques?*" and **RQ4** "*What are the considerations on the common or subset vocabulary that should be accounted for when conveying words or expressions?*" by evaluating the extraction of definitions (using strongest necessary conditions) and definability in the context of *primitive* signatures. Our exploration shows that the success of our developed definition extraction algorithm depends on whether the TBox of interest is *general* and if the concept-name of interest is cyclic. Our exploration also highlights that it is often times not the case that all definable concept names are definable with respect to the primitive signature of the input TBox, thus

this potential loss of definability should be taken into account if when considering the primitive signature as a common vocabulary.

Chapter 7 is an exploration of **RQ4** "*How can the relevant knowledge extraction techniques be integrated into existing MAS frameworks or agent communication protocols?*". In Chapter 7 we study ANEMONE and characterise the completeness of subsumers and subsumees of concept names for agent communication. We introduce two notions: contextual completeness, and standalone completeness. We empirically evaluate the usefulness of strongest necessary and weakest sufficient conditions in improving the standalone completeness of the methods proposed by the ANEMONE framework. Our results suggest that in practice, our algorithm for extracting strongest necessary conditions (Algorithm 5.1.1) may provide some enhancement in a few cases. However the dual case can not be stated for weakest sufficient conditions as we observe that in practice, our algorithm (Algorithm 5.1.2) for extracting weakest sufficient conditions rarely provides any enhancements.

Our study concludes in Chapter 8 where we investigate applications of strongest necessary conditions and weakest sufficient conditions to *ontology alignment*. We demonstrate that the knowledge extraction tools studied in this thesis, specifically uniform interpolation, strongest necessary conditions, weakest sufficient conditions, strongest domains and strongest ranges are useful for detecting conservativity violations in ontology alignments. The detection of conservativity violations helps in debugging by identifying potentially erroneous mappings that exist in alignments, thus laying the groundwork for improving the accuracy of alignments. Ontology alignments are often proposed and used as a means to establish a common vocabulary for communicating agents. As such, the integrity of an alignment affects the effectiveness of communication and Chapter 8 can be seen as an exploration of **RQ7** "*Can knowledge extraction techniques be applied to establishing a common vocabulary?*". Our results demonstrate that the knowledge extraction techniques explored have the potential to be applied to establishing a common vocabulary, but further research and empirical evaluation is required to investigate how significantly they contribute to existing methods.

9.1 Limitations of the Thesis

9.1.1 Emphasis on ALC

The overarching motivation for this thesis was to empirically investigate the application of uniform interpolation in the context of agent communication, specifically knowledge sharing applications. To this extent, there has been a focus on evaluating relevant knowledge extraction tools based on available existing implementations of uniform interpolation (such as LETHE). Our algorithms required a uniform interpolation implementation that could perform both TBox and ABox forgetting. At the time of writing the thesis, the available implementation of LETHE could only perform TBox and ABox forgetting on $\mathcal{ALC}$ ontologies. This naturally narrowed the scope of our investigation into knowledge extraction tools such as strongest necessary conditions and weakest sufficient conditions.

In an ideal case, i.e., a case where the available implementation of LETHE could handle TBox and ABox forgetting in ontologies as expressive as $\mathcal{SHROIQ}$, we would have been able to evaluate our algorithms for extracting strongest necessary and weakest sufficient conditions for more expressive ontologies. Nevertheless, a theoretical evaluation and study of the knowledge extraction tools investigated in more expressive description logics would still form a significant contribution; a relevant research question that may have been explored is: *Are strongest necessary conditions and weakest sufficient conditions always finitely representable?* A tangential question is whether all description logics have the uniform interpolation property. For example, we know [Koo15] that without the use of fixpoint operators or definer symbols, it is not possible to finitely represent uniform interpolants of ontologies with cycles.

9.1.2 Underexploration of representation limitations

Another point of concern is the *finite* representation of strongest necessary conditions and weakest sufficient conditions. A relevant research question that needs to be explored: In the case that strongest necessary conditions and weakest sufficient conditions are not finitely representable, what is the closest approximation that can be constructed? For example, we know that a finite SNC or WSC cannot be extracted for the concept A with respect to the ontology $A \equiv \exists r.A$ over the signature $\{r\}$.

9.1.3 Underexploration of alternative methods

Because of the expensive cost of uniform interpolation, another relevant research question is: What alternative options are there to extracting strongest necessary conditions and weakest sufficient conditions that are not related to uniform interpolation? As more practical methods may enable better empirical evaluation and analysis of results.

9.1.4 Emphasis on investigating sharing TBox knowledge

This thesis has focused on extracting descriptions of concepts and roles. However, a significant aspect of ontologies is the ABox, furthermore, most logic-based agent systems have implementations that use ABoxes to represent the beliefs of an agent. Theoretically speaking, uniform interpolation and reasoning should suffice to restrict an ABox to a specific signature; however, a relevant point of inquiry is the empirical evaluation of uniform interpolation using ontologies with existing ABoxes, and furthermore, dynamic ABoxes.

9.2 Application to General Aspects of Agent Communication

In this section, we outline how the knowledge extraction tools we have studied are worth taking into consideration when designing communication protocols for agent communication.

9.2.1 Factors to consider in the Design of Communication Protocols

Interpretation of the common signature

When designing communication protocols for agents, an agent engineer must be aware of how the interpretation of the common signature can affect the accuracy of the knowledge shared between agents.

If the agent engineer requires or assumes all symbols in the common signature to have the same interpretation (in the respective ontologies of the agents), then it is safe to assume that most knowledge extraction techniques convey enough and accurate

information about shared concepts, in fact, some more effectively (in terms of signature minimality) than others: for example a strongest necessary condition (weakest sufficient condition) of a concept conveys as much information as the compiled super concept (compiled subconcept). In such cases, the agent engineer can prioritise SNCs/WSCs over CSPs/CSBs in the agent communication protocols as they are more likely to be less in size than the CSPs/CSBs.

Given a pair of ontologies O_1 and O_2 for the respective communicating agents AG_1 and AG_2, in a less restrictive case, an agent engineer may require or assume that either (1) O_1 is a conservative extension of O_2 over the common signature, or (2) O_2 is a conservative extension of O_1 over the common signature. An example is a case in which one of the communicating agents only has a TBox and the other agent only has an ABox using some intersecting signature; it may be that the TBox agent aggregates terminological data from other TBox agents following some policy and the ABox agent collects instances using its sensors or classifiers or from other ABox agents following its own policy. Again, in such a case, it is safe to assume that most knowledge extraction techniques addressed in Chapter 5 convey enough and accurate information about concepts.

In the case that the agent engineer has no control, assumptions, or estimate about the interpretation of the symbols in the common signature, the agent engineer must take into account the communication objectives and constraints of the agents to design an effective communication protocol. As an example, there may be cases in which the consistency of the interpretation of the common signature is of little importance: consider two agents designed to share instances of the concept $HospitalPatient$ with the intention of aggregating as many instances of the concept, regardless of how it is defined or described in an ontology, $HospitalPatient$ may be defined (with respect to the common signature) as $Outpatient$ in one agent's ontology, and $\exists hasPrescription.Medicine$ in another agent's ontology. Here it does not matter as long as the communication process maximises the number of instances that are aggregated. In such cases, it may be the case that the agents seeking to aggregate instances have an internal mechanism that helps filter out genuine instances from irrelevant instances. An agent engineer may also not care about the consistency of the interpretation of the common signature in order to account for incomplete modeling in an ontology. As an example scenario, consider an agent engineer attempting to design a multi-agent system to aggregate as many instances of home addresses in the world: a home address is a concept that can be very unique to a territorial body, and it may be impossible to

create a definition that captures all the possible interpretation of the concept.

Without the guarantee of some notion of consistency over the common signature of the communicating agents, the consistency of the information shared about concepts is also not guaranteed. In such cases, an agent engineer may want to identify whether the knowledge of agents is compatible. A simple algorithm to accomplish this is to extract the uniform interpolants of both ontologies with respect to the common signature and use a reasoner to verify the consistency of the union of the extracted uniform interpolants, i.e., given two ontologies O_1 and O_2 representing the ontologies of communicating agents with a common vocabulary Σ, extract a uniform interpolant $\mathcal{V}_1$ with respect to Σ of O_1, and a uniform interpolant $\mathcal{V}_2$ with respect to Σ of O_2 and verify $\mathcal{V}_1 \cup \mathcal{V}_2$ is consistent.

Assuming knowledge of the communicating agents is compatible in some sense and that there is no guarantee that the symbols in the common signature have the same interpretation, a further point of consideration is the tool used to extract information. In a scenario where the agents wish to convey information about concepts, it may be necessary for the agents to prioritise using compiled superconcepts (if a mechanism exists to extract them) over strongest necessary conditions in order to ensure that information is maximally captured.

Time/Space constraints

Depending on the application scenario, there may be constraints on time (in terms of latency) and space (in terms of message size) between communicating agents.

In the presence of strict constraints on latency (i.e., constraints that require agents to respond quickly), an agent engineer should consider the potential worst computational complexity associated with knowledge extraction methods against other factors such as the accuracy of the information shared. For example, for scenarios with strict time constraints that may also impose strict constraints on the information conveyed (i.e., all conveyed content must be within the common signature), it may be useful to ensure that agents precompute respective uniform interpolants, SNCs/WSCs of their ontologies with respect to the common signature prior to communicating.

The agent engineer must balance their selection of tools (based on time constraints) against space constraints, taking into consideration the complexity of knowledge extraction tools. Large messages may increase the latency of communication and may be thus more difficult to understand. Therefore, care must be taken when choosing uniform interpolation (or any method based on uniform interpolation) as a knowledge

extraction device given the double exponential complexity associated with uniform interpolation. Similarly, care must be taken when choosing module extraction as the knowledge extraction device due to the possibility for current implementations to extract redundant information when extracting modules.

Expressivity A property of ontologies that can severely limit knowledge extraction techniques, is its expressivity. For example, to the best of our knowledge practical uniform interpolation methods for DLs that allow for nominals are lacking. As such, an agent engineer must be familiar with the limitations of the knowledge extraction tools of choice when implementing communication protocols.

It is also worth noting that knowledge extraction tools may produce results that are more expressive than the language of the ontology from which the knowledge is being extracted. This is likely to be the case when results cannot be represented within the original expressivity of the ontology from which the knowledge is being extracted. LETHE, for example, may use fixpoints to represent certain concept descriptions for results that are not finitely respresentable. Fixpoints are not a standard DL construct, and to the best of our knowledge there are no existing implementations of reasoners that facilitate reasoning with concept descriptions that include fixpoints. LETHE and FAME may also extend a less expressive ontology (such as $\mathcal{ALC}$) to $\mathcal{ALCOI}$ in order to represent some results.

Frequency and Duration of Communication

The estimated frequency of communication between communicating agents should influence the knowledge extraction tool to be used and the nature of knowledge extraction for communication scenarios that require knowledge extraction. For example, if communicating agents are envisaged to be communicating frequently, it may be worth pre-computing any artefact required for communication such as uniform interpolants or strongest necessary conditions, as this can help reduce the latency of communication.

Long term communication may require artefacts that the agents can reference as opposed to computing on the fly: it may be beneficial to construct a common ontology based on the common signature via uniform interpolation or module-extraction as opposed to repeatedly extracting strongest necessary conditions and weakest sufficient conditions.

Applications of conveyed information

A potential issue to consider is whether the knowledge extracted or conveyed needs to be readable or not. As demonstrated in Chapter 6, extracted superconcepts and subconcepts can often contain superfluous information; this, combined with the double exponential complexity associated with uniform interpolation suggests that there are cases in which knowledge extracted using uniform interpolation may be difficult to process by agents depending on the size (e.g., in cases where the signature of the compiled superconcepts exceed that of the bottom module as discussed in Section 6.4).

The application domain influences the nature of the knowledge extraction tool to be used. We envisage three general application patterns that would influence the knowledge extraction tools selected:

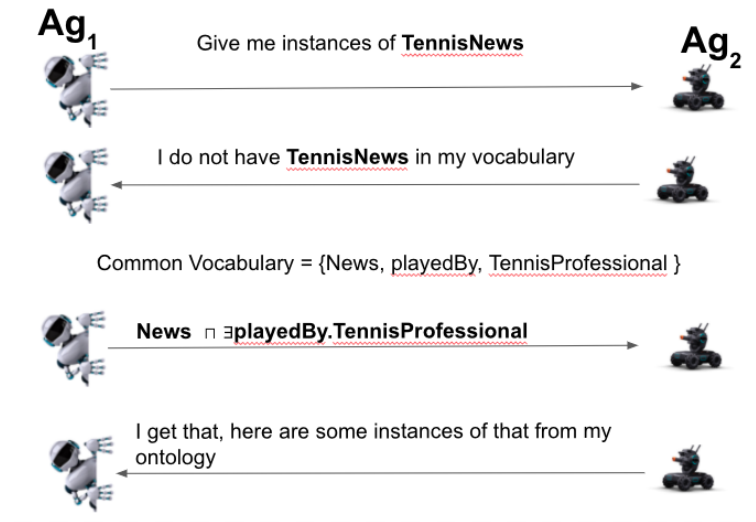

Figure 9.1: ANEMONE case.

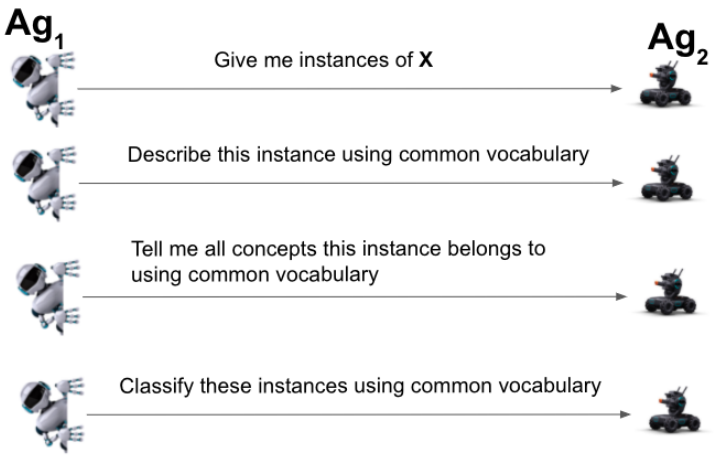

Figure 9.2: Other cases.

1. **Sharing instances of concepts:** As a motivating example, consider agents attempting to share instances of movie genres. If the agents do not have any built in classifiers or sensors to classify an individual to a class, it is safer to always request instances of the weakest sufficient condition of an uncommon movie genre X (with respect to the common vocabulary) in order to ensure that all the instances returned are indeed instances of the required movie genre. However, this comes with the disadvantage that a weakest sufficient condition may not exist or that the weakest sufficient condition only captures a minimal portion of the required movie genre (when in reality, the other agent has a lot more instances). Alternatively the agent may request instances of the strongest necessary condition of the uncommon movie genre X. However, the instances returned may capture other instances of other classes as the strongest necessary condition is a subsumer approximation of X. However, if the requesting agent has an inbuilt classifier, it can always filter instances of relevant movie genres from the returned set of instances. The agent engineer must asses the communication goal of the communication protocol in tandem with the agent's capabilities in order to maximise the efficiency of the knowledge exchanged between agents.

2. **Sharing information about individuals:** In applications where the domain of interpretation is shared among agents, agents may ask other agents to classify instances. As a motivating example, consider a scenario where agents are communicating to catalogue URLs on the internet (i.e., the domain of interpretation

is the set of all URLs on the internet), an agent may encounter a URL u and have no means to classify u thereby prompting them to ask another agent to classify u using only symbols in their common vocabulary. If the responding agent can classify u, it may simply most specific concept C of u and extract a strongest necessary condition of C with respect to the common vocabulary to convey to the requesting agent.

Another example is an agent requesting additional facts about an individual, for example, in the cataloguing scenario, the requesting agent may want to know all possible concepts the URL u belongs to (with respect to the common vocabulary). In such a scenario, the compiled super concept of C is better suited than strongest necessary conditions to convey the required information.

3. **Sharing TBox knowledge or TBox information about concepts:** In some cases, agents may only be interested in sharing TBox knowledge. As an example, consider an assembly line scenario in which the schematics of a device to be assembled are represented using only the TBox. Each agent in the MAS has a vocabulary associated with their assembly capabilities (each agent may be associated with a robot). For example, there may be a MAS consisting of an agent that specialises in electronics, an agent that specialises in chasis building, and an agent that specialises in testing and verifying. Such a MAS may be applied to assemble different products (cars, computers, ovens, etc) which have their respective schematics represented using a large TBox. Thus in order to capture the relevant aspects of the schematics, a uniform interpolant or a module may be extracted with respect to the vocabulary of each agent. In another case, agents may only be interested in the concept descriptions in their respective ontologies. For example agents may be only be communicating with other agents to augment or diagnose the knowledge in their ontologies. Depending on the level of knowledge required, it may be preferable to extract a uniform interpolant, a strongest necessary condition, or a weakest sufficient condition.

9.3 Further Applications

9.3.1 Ontology Strengthening

A potential application of strongest necessary conditions and weakest sufficient conditions not explored in this thesis, is exploring their applications to ontology engineering. Specifically, we are referring to cases where the ontology of interest is constructed using a description logic that has the Beth definability property. In such cases, an ontology can be potentially *strengthened* by extracting explicit definitions for implicitly defined concepts and proposing such definitions to an ontology engineer as an alternative description for implicitly defined concepts.

9.3.2 Constructing Ontology Alignments

In Chapter 8 we defined ascending, descending, and relation entailment sets as a basis for detecting general conservativity violations. However, a relevant point of inquiry is to investigate how ontology alignments may be constructed using ascending, descending, and relation entailment sets: the class hierarchy of an ontology is often used as the basis for constructing alignments in existing alignment tools such as LogMap and AML, thus the hypothesis here is that similar principles can be applied to construct ontology alignments on the basis of ascending, descending and relation entailment sets.

Bibliography

[ADMF13] Mohsen Afsharchi, Arman Didandeh, Nima Mirbakhsh, and Behrouz Far. Common understanding in a multi-agent system using ontology-guided learning. *Knowledge and Information Systems*, 36(1):83–120, Jul 2013.

[AJM+21] Alessandro Artale, Jean Christoph Jung, Andrea Mazzullo, Ana Ozaki, and Frank Wolter. Living without beth and craig: Definitions and interpolants in description logics with nominals and role inclusions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6193–6201, 2021.

[ANPAA+21] Mina Abd Nikooie Pour, Alsayed Algergawy, Florence Amardeilh, Reihaneh Amini, Omaima Fallatah, Daniel Faria, Irini Fundulaki, Ian Harrow, Sven Hertling, Pascal Hitzler, Martin Huschka, Liliana Ibanescu, Ernesto Jimenez-Ruiz, Naouel Karam, Amir Laadhar, Patrick Lambrix, Huanyu Li, Ying Li, Franck Michel, Engy Nasr, Heiko Paulheim, Catia Pesquita, Jan Portisch, Catherine Roussey, Tzanina Saveta, Pavel Shvaiko, Andrea Splendiani, Cassia Trojahn, Jana Vatascinova, Beyza Yaman, Ondrej Zamazal, and Lu Zhou. Results of the ontology alignment evaluation initiative 2021. In *Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 25, 2021 :*, number 3063 in CEUR Workshop Proceedings, pages 62–108. CEUR-WS.org, 2021.

[AP19] Anas Alseyat and Jae-Do Park. Multi-agent system using jade for distributed dc microgrid system control. In *2019 North American Power Symposium (NAPS)*, pages 1–5. IEEE, 2019.

[AvH04] Grigoris Antoniou and Frank van Harmelen. *Web Ontology Language: OWL*, pages 67–92. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[Baa03] Franz Baader. Computing the least common subsumer in the description logic el w.r.t. terminological cycles with descriptive semantics. In Bernhard Ganter, Aldo de Moor, and Wilfried Lex, editors, *Conceptual Structures for Knowledge Creation and Communication*, pages 117–130, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Bai13] Samantha Bail. *The justificatory structure of OWL ontologies*. The University of Manchester (United Kingdom), 2013.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, Daniele Nardi, et al. *The description logic handbook : theory, implementation, and applications*. Cambridge University Press, Cambridge, UK ;, 2003.

[BHW07] Rafael Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons, 2007.

[Bie07] Meghyn Bienvenu. Consequence finding in alc. In *Proceedings of the Twentieth International Workshop on Description Logics (DL2007)*, volume 250 of *CEUR Workshop Proceedings*, 2007.

[Bie08] Meghyn Bienvenu. Prime implicate normal form for alc concepts. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 1*, AAAI'08, page 412–417. AAAI Press, 2008.

[Bie09] Meghyn Bienvenu. Prime implicates and prime implicants: From propositional to modal logic. *Journal of Artificial Intelligence Research*, 36:71–128, 2009.

[BLCT16] Michael Benedikt, Julien Leblay, Balder ten Cate, and Efthymia Tsamoura. Generating plans from proofs: the interpolation-based approach to query reformulation. *Synthesis Lectures on Data Management*, 8(1):1–205, 2016.

[BLR+19] Elena Botoeva, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Query inseparability for $\mathcal{ALC}$ ontologies. *Artificial Intelligence*, 272:1–51, 2019.

[Boo09] George Boole. *An Investigation of the Laws of Thought: On Which Are Founded the Mathematical Theories of Logic and Probabilities*. Cambridge Library Collection - Mathematics. Cambridge University Press, 2009.

[BPR01] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with jade. In *Intelligent Agents VII Agent Theories Architectures and Languages*, Lecture Notes in Computer Science, pages 89–103, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[Bro86] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal of robotics and automation*, 2(1):14–23, 1986.

[BT02] Sidney Bailin and Walt Truszkowski. Ontology negotiation: How agents can really get to know each other. In *Workshop on Radical Agent Concepts*, pages 320–334. Springer, 2002.

[CCMV06] Balder ten Cate, Willem Conradie, Maarten Marx, and Yde Venema. Definitorially complete description logics. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'06, page 79–89. AAAI Press, 2006.

[dKLW05] Mark d'Inverno, David Kinny, Michael Luck, and Michael Wooldridge. A formal specification of dmars. In *Intelligent Agents IV Agent Theories, Architectures, and Languages*, Lecture Notes in Computer Science, pages 155–176, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[DLS01] Patrick Doherty, Witold Łukaszewicz, and Andrzej Szałas. Computing strongest necessary and weakest sufficient conditions of first-order formulas. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'01, page 145–151, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[DLS07] Patrick Doherty, Witold Lukaszewicz, and Andrzej Szalas. *Knowledge representation techniques: a rough set approach*, volume 202. Springer, 2007.

[Don06] Kevin Donnelly. Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 121:279, 2006.

[DTPP09] Paul Doran, Valentina Tamma, Terry Payne, and Ignazio Palmisano. Dynamic selection of ontological alignments: A space reduction mechanism. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, page 2028–2033, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[DV13] Chiara Del Vescovo. *The modular structure of an ontology: Atomic Decomposition and its applications*. PhD thesis, University of Manchester, 2013.

[ECMC14] Soledad Escolar, Jesús Carretero, Maria-Cristina Marinescu, and Stefano Chessa. Estimating energy savings in smart street lighting by using an adaptive control system. *International Journal of Distributed Sensor Networks*, 10(5):971587, 2014.

[ES07] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag, Berlin, Heidelberg, 1. aufl. edition, 2007.

[FFMM94] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. Kqml as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management*, CIKM '94, page 456–463, New York, NY, USA, 1994. Association for Computing Machinery.

[Fit12] Melvin Fitting. *First-order logic and automated theorem proving*. Springer Science & Business Media, 2012.

[FMA92] Ingrand Felix, Georgeff Michael, and Rao Ayyagari. An architecture for real-time reasoning and system control. *IEEE expert*, 7(6):34–44, 1992.

[FPS+13] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel Cruz, and Francisco M Couto. The agreementmakerlight ontology matching system. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, Lecture Notes in Computer Science, pages 527–541, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Gab08] Dov Gabbay. *Second-order quantifier elimination : foundations, computational aspects and applications*. Studies in logic : Mathematical logic and foundations ; 12. College Publications, London, 2008.

[Gel18] David Geleta. *Minimal Definition Signatures: Computation and Application to Ontology Alignment*. PhD thesis, University of Liverpool, 2018.

[GF+92] Michael Genesereth, Richard E Fikes, et al. Knowledge interchange format-version 3.0: reference manual. 3, 1992.

[GHKS08] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular reuse of ontologies: Theory and practice. *J. Artif. Int. Res.*, 31(1):273–318, feb 2008.

[GHM+14] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit: An owl 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.

[Gia96] Giuseppe De Giacomo. Eliminating ”converse” from converse pdl. *Journal of Logic, Language, and Information*, 5(2):193–208, 1996.

[GLW06] Silvio Ghilardi, Carsten Lutz, and Frank Wolter. Did i damage my ontology? a case for conservative extensions in description logic. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, KR’06, page 187–197. AAAI Press, 2006.

[GO92] Dov Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second-order predicate logic. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, KR’92, page 425–435, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[GPS11] Rafael Gonçalves, Bijan Parsia, and Ulrike Sattler. Categorising logical differences between owl ontologies. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, page 1541–1546, New York, NY, USA, 2011. Association for Computing Machinery.

[GPS12] Rafael Gonçalves, Bijan Parsia, and Ulrike Sattler. Concept-based semantic difference in expressive description logics. In *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I*, ISWC'12, page 99–115, Berlin, Heidelberg, 2012. Springer-Verlag.

[GPSK06] Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Modularity and web ontologies. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'06, page 198–208. AAAI Press, 2006.

[GPT16] David Geleta, Terry Payne, and Valentina Tamma. An investigation of definability in ontology alignment. In *Knowledge Engineering and Knowledge Management*, volume 10024 of *Lecture Notes in Computer Science*, pages 255–271. Springer International Publishing, Cham, 2016.

[GPT17] David Geleta, Terry Payne, and Valentina Tamma. Minimal coverage for ontology signatures. In Mauro Dragoni, María Poveda-Villalón, and Ernesto Jimenez-Ruiz, editors, *OWL: Experiences and Directions – Reasoner Evaluation*, pages 128–140, Cham, 2017. Springer International Publishing.

[GZ95] Silvio Ghilardi and Marek Zawadowski. Undefinability of propositional quantifiers in the modal system s4. *Studia Logica*, 55(2):259–271, Jun 1995.

[HB11] Matthew Horridge and Sean Bechhofer. The owl api: A java api for owl ontologies. *Semant. Web*, 2(1):11–21, jan 2011.

[HCZS14] Frank Ibarra Hernandez, Carlos Alberto Canesin, Ramon Zamora, and Anurag K Srivastava. Active power management in multiple microgrids using a multi-agent system with jade. In *2014 11th IEEE/IAS*

International Conference on Industry Applications, pages 1–8. Ieee, 2014.

[HM+00] James Hendler, Deborah McGuinness, et al. The darpa agent markup language. *IEEE Intelligent systems*, 15(6):67–73, 2000.

[JRCG11] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer, 2011.

[JRGHB11] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ian Horrocks, and Rafael Berlanga. Logic-based assessment of the compatibility of umls ontology sources. *Journal of biomedical semantics*, 2 Suppl 1(Suppl 1):S2–S2, 2011.

[JRGHL09] Ernesto Jimenez-Ruiz, Bernardo Cuenca Grau, Ian Horrocks, and Rafael Berlanga Llavori. Logic-based ontology integration using contentmap. In *JISBD*, pages 316–319, 2009.

[JRGZH12] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Yujiao Zhou, and Ian Horrocks. Large-scale interactive ontology matching: Algorithms and implementation. In *ECAI*, volume 242, pages 444–449, 2012.

[KB08] Thomas Klapiscak and Rafael Bordini. Jasdl: A practical programming approach combining agent and semantic web technologies. In *International Workshop on Declarative Agent Languages and Technologies*, pages 91–110. Springer, 2008.

[KHA06] Thomas Konnerth, Benjamin Hirsch, and Sahin Albayrak. Jadl – an agent description language for smart agents. In Matteo Baldoni and Ulle Endriss, editors, *Declarative Agent Languages and Technologies IV*, pages 141–155, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[KLWW09] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. *Formal Properties of Modularisation*, pages 25–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Koo15] Patrick Koopmann. *Practical uniform interpolation for expressive description logics*. PhD thesis, The University of Manchester, 2015.

[KWW09] Boris Konev, Dirk Walther, and Frank Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *International Joint Conference on Automated Reasoning*, pages 830–835, 2009.

[LBBH06] Michal Laclavík, Zoltán Balogh, Marián Babík, and Ladislav Hluchý. Agentowl: Semantic knowledge model and agent architecture. *Computing and informatics (Bratislava, Slovakia)*, 25(5):421–439, 2006.

[LBT+07] Loredana Laera, Ian Blacoe, Valentina Tamma, Terry Payne, Jerôme Euzenat, and Trevor Bench-Capon. Argumentation over ontology correspondences in mas. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, 2007.

[LDGS11] Lin Padgham, David Scerri, Gaya Jayatilleke, and Sarah Hickmott. Integrating bdi reasoning into agent based modeling and simulation. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 345–356, 2011.

[Lin01] Fangzhen Lin. On strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, 128(1):143–159, 2001.

[LK13] Michel Ludwig and Boris Konev. Towards practical uniform interpolation and forgetting for alc tboxes. In *In Proceedings of DL 2013, volume 1014 of CEUR Workshop Proceedings*, 2013.

[LKK15] Marco Lützenberger, Thomas Konnerth, and Tobias Küster. Programming of multiagent applications with jiac. In *Industrial agents*, pages 381–398. Elsevier, 2015.

[LPL+14] JeeHang Lee, Julian Padget, Brian Logan, Daniela Dybalova, and Natasha Alechina. N-jason: Run-time norm compliance in agentspeak (l). In *International Workshop on Engineering Multi-Agent Systems*, pages 367–387. Springer, 2014.

[LPW10] Carsten Lutz, Robert Piro, and Frank Wolter. Enriching el-concepts with greatest fixpoints. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, page 41–46, NLD, 2010. IOS Press.

[LR94] Fangzhen Lin and Ray Reiter. Forget it. In *Working Notes of AAAI Fall Symposium on Relevance*, pages 154–159, 1994.

[LW11] Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[LWW07a] Carsten Lutz, Dirk Walther, and Frank Wolter. Conservative extensions in expressive description logics. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, page 453–458, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[LWW07b] Carsten Lutz, Dirk Walther, and Frank Wolter. Conservative extensions in expressive description logics. In *International Joint Conference on Automated Reasoning*, volume 7, pages 453–458, 2007.

[Mar07] Maarten Marx. Queries determined by views: Pack your views. In *Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, page 23–30, New York, NY, USA, 2007. Association for Computing Machinery.

[Mat16] Nicolas Alexander Matentzoglu. *Module-based classification of OWL ontologies*. PhD thesis, University of Manchester, 2016.

[McM03] Kenneth McMillan. Interpolation and sat-based model checking. In *International Conference on Computer Aided Verification*, pages 1–13. Springer, 2003.

[MFHS02] Deborah Mcguinness, Richard Fikes, James Hendler, and Lynn Stein. Daml+oil: an ontology language for the semantic web. *IEEE Intelligent Systems*, 17(5):72–80, 2002.

[MGH+09] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, Carsten Lutz, et al. Owl 2 web ontology language profiles. *W3C recommendation*, 27:61, 2009.

[MP94] Joerg Muller and Markus Pischel. Integrating agent interaction into

a planner-reactor architecture. In *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence, Seattle, WA, USA*, 1994.

[MS21] Renate Schmidt Mostafa Sakr. Semantic forgetting in expressive description logics. In B. Konev and G. Reger, editors, *Frontiers of Combining Systems (FroCoS 2021)*, volume 12941 of *Lecture Notes in Computer Science*, pages 118–136. Springer, 2021.

[MVBH05] Alvaro Moreira, Renata Vieira, Rafael Bordini, and Jomi Hübner. Agent-oriented programming with underlying ontological reasoning. In *International Workshop on Declarative Agent Languages and Technologies*, pages 155–170. Springer, 2005.

[Neb90a] Bernhard Nebel. *Terminological cycles*, pages 119–147. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990.

[Neb90b] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249, 1990.

[Nik11] Nadeschda Nikitina. Uniform interpolation in general el terminologies. In *24th International Workshop on Description Logics*, page 345, 2011.

[PK15a] Renate Schmidt Patrick Koopmann. LETHE: A saturation-based tool for non-classical reasoning. In *Proc. ORE-2015*, volume 1387 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[PK15b] Renate Schmidt Patrick Koopmann. LETHE: A saturation-based tool for non-classical reasoning. In *Proceedings of the 4th International Workshop on OWL Reasoner Evaluation (ORE-2015)*, volume 1387 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[PK15c] Renate Schmidt Patrick Koopmann. Saturation-based forgetting in the description logic $\mathcal{SIF}$. In D. Calvanese and B. Konev, editors, *Proceedings of the 28th International Workshop on Description Logics (DL-2015)*, volume 1350 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[PK15d] Renate Schmidt Patrick Koopmann. Uniform interpolation and forgetting for $\mathcal{ALC}$ ontologies with aboxes. In B. Bonet and S. Koenig, editors, *Proc. AAAI-2015*, pages 175–181. AAAI Press, 2015.

[PT14] Terry Payne and Valentina Tamma. Negotiating over ontological correspondences with asymmetric and incomplete knowledge. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 517–524, 2014.

[Ram92] Ramesh Patil, Richard Fikes, Peter Patel-Schneider, Donald McKay, Tim Finin, Thomas Gruber, and Robert Neches. The DARPA Knowledge Sharing Effort: Progress Report. In Bernhard Nebel, editor, *Proceedings of the Third International Conference on Principles Of Knowledge Representation And Reasoning*. Morgan Kaufman, August 1992.

[Rao96] Anand S. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language. In Walter Van de Velde and John W. Perram, editors, *Agents Breaking Away*, pages 42–55, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[RBHC19] Alessandro Ricci, Rafael Bordini, Jomi Hübner, and Rem Collier. Agentspeak(er): Enhanced encapsulation in agent plans. In *Engineering Multi-Agent Systems*, Lecture Notes in Computer Science, pages 34–51, Cham, 2019. Springer International Publishing.

[RG+95] Anand Rao, Michael Georgeff, et al. Bdi agents: from theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995.

[SCE15] Chaimae Saadi, Habiba Chaoui, and Hassan Erguig. Security analysis using ids based on mobile agents and data mining algorithms. *(IJCSIT) International Journal of Computer Science and Information Technologies*, 6(1):597–602, 2015.

[Sch05] Stefan Schlobach. Debugging and semantic clarification by pinpointing. In *European Semantic Web Conference*, pages 226–240. Springer, 2005.

[Sch12] Renate A Schmidt. The ackermann approach for modal logic, correspondence theory and second-order reduction. *Journal of Applied Logic*, 10(1):52–74, 2012.

[SG11] Diego Goncalves Silva and Joao Carlos Gluz. Agentspeak(pl): A new programming language for bdi agents with integrated bayesian network model. In *2011 International Conference on Information Science and Applications*, pages 1–7. IEEE, 2011.

[Sho93] Yoav Shoham. Agent-oriented programming. *Artificial intelligence*, 60(1):51–92, 1993.

[SJRG17a] Alessandro Solimando, Ernesto Jimenez-Ruiz, and Giovanna Guerrini. Minimizing conservativity violations in ontology alignments: Algorithms and evaluation. *Knowledge and Information Systems*, 51(3):775–819, 2017.

[SJRG17b] Alessandro Solimando, Ernesto Jiménez-Ruiz, and Giovanna Guerrini. Minimizing conservativity violations in ontology alignments: algorithms and evaluation. *Knowledge and Information Systems*, 51(3):775–819, 2017.

[SKH11] František Simančík, Yevgeny Kazakov, and Ian Horrocks. Consequence-based reasoning beyond horn ontologies. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, page 1093–1098. AAAI Press, 2011.

[SLZ04] Kaile Su, Guanfeng Lv, and Yan Zhang. Reasoning about knowledge by variable forgetting. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning*, KR'04, page 576–586. AAAI Press, 2004.

[SMVM16] Marlo Souza, Alvaro Moreira, Renata Vieira, and John-Jules Meyer. Integrating ontology negotiation and agent communication. In *Ontology Engineering*, Lecture Notes in Computer Science, pages 56–68, Cham, 2016. Springer International Publishing.

[SS69] John Searle and John Rogers Searle. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press, 1969.

[SSZ09] Ulrike Sattler, Thomas Schneider, and Michael Zakharyaschev. Which kind of module should i extract? *Description Logics*, 477:78, 2009.

[Stu03] Russell Stuart. *Artificial intelligence : a modern approach*. Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, N.J, 2nd ed. edition, 2003.

[Sza93] Andrzej Szalas. On the correspondence between modal and classical logic: An automated approach. *Journal of Logic and Computation*, 3(6):605–620, 1993.

[TCFS13] Balder Ten Cate, Enrico Franconi, and Inanç Seylan. Beth definability in expressive description logics. *Journal of Artificial Intelligence Research*, 48:347–414, 2013.

[TS19] David Toluhi and Renate Schmidt. Knowledge sharing among agents with ontological reasoning. In *Automated Reasoning Workshop 2019*, page 1, 2019.

[TS21] David Toluhi and Renate Schmidt. Knowledge extraction for multi-agent system communication. In *Logical Aspects in Multi-Agent Systems and Strategic Reasoning*, page 1, 2021.

[TSP22] David Toluhi, Renate Schmidt, and Bijan Parsia. Concept description and definition extraction for the anemone system. In Natasha Alechina, Matteo Baldoni, and Brian Logan, editors, *Engineering Multi-Agent Systems*, pages 352–372, Cham, 2022. Springer International Publishing.

[vB08] Johan van Benthem. The many faces of interpolation. *Synthese*, 164(3):451–460, 2008.

[vDBD+06] Jurriaan van Diggelen, Robbert-Jan Beun, Frank Dignum, Rogier van Eijk, and John-Jules Meyer. Anemone: An effective minimal ontology negotiation environment. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, page 899–906, New York, NY, USA, 2006. Association for Computing Machinery.

[VDBD+07a] Jurriaan Van Diggelen, Robbert-Jan Beun, FPM Dignum, Rogier Van Eijk, and John-Jules Meyer. Ontology negotiation: goals, requirements and implementation. *International Journal of Agent-Oriented Software Engineering*, 1(1):63–90, 2007.

[VDBD+07b] Jurriaan Van Diggelen, Robbert-Jan Beun, FPM Dignum, Rogier Van Eijk, and John-Jules Meyer. Ontology negotiation: goals, requirements and implementation. *International Journal of Agent-Oriented Software Engineering*, 1(1):63–90, 2007.

[VDRA11] Viviana Mascardi, Davide Ancona, Rafael Bordini, and Alessandro Ricci. Cool-agentspeak: Enhancing agentspeak-dl agents with plan exchange and ontology services. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 109–116. IEEE, 2011.

[VMWB07] Renata Vieira, Álvaro Moreira, Michael Wooldridge, and Rafael Bordini. On the formal semantics of speech-act based communication in an agent-oriented programming language. *Journal of Artificial Intelligence Research*, 29:221–267, 2007.

[Wan10] Peng Wang. Lily-lom: An efficient system for matching large ontologies with non-partitioned method. In *Proc. CEUR Workshop*, volume 658, pages 69–72. CEUR-WS.org, 2010.

[WDP19] Renate Schmidt Warren Del-Pinto. ABox abduction via forgetting in ALC. In P. Van Hentenryck and Z.-H. Zhou, editors, *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-2019)*, pages 2768–2775. AAAI Press, 2019.

[Wer12] Christoph Wernhard. Projection and scope-determined circumscription. *Journal of Symbolic Computation*, 47(9):1089–1108, 2012.

[Wer14] Christoph Wernhard. Second-order characterizations of definientia in formula classes. In *Knowledge Representation and Reasoning 14-03*, 2014.

[WNS+11] Patricia Whetzel, Natalya Noy, Nigam Shah, Paul Alexander, Csongor Nyulas, Tania Tudorache, and Mark Musen. Bioportal: enhanced functionality via new web services from the national center for biomedical

ontology to access and use ontologies in software applications. *Nucleic acids research*, 39(suppl_2):W541–W545, 2011.

[YZ15] Renate Schmidt Yizheng Zhao. Concept forgetting in alcoi-ontologies using an ackermann approach. In *The Semantic Web, 14th International Semantic Web Conference, ISWC 2015*, volume 9366 of *Lecture Notes in Computer Science*, pages 587–602. Springer, 2015.

[YZ18] Renate Schmidt Yizheng Zhao. FAME: An automated tool for semantic forgetting in expressive description logics. In *Proc. IJCAR 2018*, volume 10900 of *Lecture Notes in Artificial Intelligence*, pages 19–27. Springer, 2018.

[ZAS+19] Yizheng Zhao, Ghadah Alghamdi, Renate Schmidt, Hao Feng, Giorgos Stoilos, Damir Juric, and Mohammad Khodadadi. Tracking logical difference in large-scale ontologies: a forgetting-based approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3116–3124, 2019.

[ZWL+14] Yuanzhe Zhang, Xuepeng Wang, Siwei Lai, Shizhu He, Kang Liu, Jun Zhao, and Xueqiang Lv. Ontology matching with word embeddings. In Maosong Sun, Yang Liu, and Jun Zhao, editors, *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 34–45, Cham, 2014. Springer International Publishing.

[ZZ10] Yan Zhang and Yi Zhou. Forgetting revisited. In *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning*, KR'10, page 602–604. AAAI Press, 2010.