A FRAMEWORK FOR GENERALIZED SYLLOGISMS

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Ersin ÇİNE

July 2018
İZMİR

We approve the thesis of **Ersin ÇİNE**

Examining Committee Members:

Assoc. Prof. Dr. Y. Murat ERTEN
Department of Computer Engineering, İzmir Institute of Technology

---

Assoc. Prof. Dr. Tahsin ÖNER
Department of Mathematics, Ege University

Dr. Mustafa ÖZUYSAL
Department of Computer Engineering, İzmir Institute of Technology

10 July 2018

Assoc. Prof. Dr. Y. Murat ERTEN
Supervisor, Department of Computer Engineering
İzmir Institute of Technology

Assoc. Prof. Dr. Y. Murat ERTEN
Head of the Department of
Computer Engineering

Prof. Dr. Aysun SOFUOĞLU
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGMENTS

I would like to thank ...

... my supervisor, Assoc. Prof. Dr. Y. Murat ERTEN for trusting my abilities and respecting my decisions. Despite the fact that his research interests do not exactly match with the subject of this thesis work, his wisdom and experience encouraged and guided me a lot.

... my initial supervisor, Dr. Bora İ. KUMOVA: he has been exceptionally inspiring for me until he leaves the university. I had the chance to take several interesting classes from him. I am grateful to him for his studies on syllogisms without which this thesis would have a completely different title.

... Assoc. Prof. Dr. Tahsin ÖNER and Asst. Prof. Dr. Mustafa ÖZUYSAL for their participation as examining committee members.

... my mother, Emine BAYRAM: her love and support are great blessings. She has listened my presentation rehearsals with gleaming eyes without knowing English.

... my sister, Elvan ÇİNE: despite our contrasting characters, she has always been a great sister. She had been very easygoing to me in the process of writing this thesis.

ABSTRACT

A FRAMEWORK FOR GENERALIZED SYLLOGISMS

Reasoning is an indispensable action both for natural intelligence and for artificial intelligence. In automated reasoning, relatively expressive logics are used to define and derive complex facts about the real world. Many facts cannot be expressed in inexpressive logics such as syllogistic logic and thus, those logics are naturally ignored for automated reasoning. Despite their inexpressiveness, logics with intuitive propositions can provide two advantages: favorable complexity properties for reasoning tasks, and better correspondence with the natural language statements. Syllogistic logic, the first known formal logic in history, is so intuitive that it is often studied by cognitive scientists in order to understand and model human reasoning. The problem is that its syntax and semantics do not allow for representing much knowledge. Hence, propositions of syllogistic logic should be generalized with useful extensions without sacrificing the advantageous properties much. The aim of this thesis work is to bridge the gap between syllogistic reasoning and automated reasoning via designing the underlying logic and framework for performing time-efficient fully-automated deduction over generalized syllogistic propositions. In this thesis work, we define a practical family of generalized syllogistic logics, reveal the theoretical properties of those logics and their relationships with some other logics, and specify comprehensive frameworks with alternative representations and methods.

ÖZET

GENELLEŞTİRİLMİŞ TASIMLAR İÇİN BİR ÇATI

Akıl yürütme, hem doğal zeka hem de yapay zeka için vazgeçilmez bir iş. Otomatik akıl yürütmede, ifade gücü nispeten yüksek olan mantıklar gerçek dünya hakkındaki karmaşık gerçekleri tanımlamak ve türetmek için kullanılır. Birçok gerçek, tasımsal mantık gibi ifade gücü zayıf mantıklarda ifade edilemez ve bu yüzden doğal olarak bu mantıklar otomatik akıl yürütme amaçlı kullanılmaz. İfade güçlerinin zayıflıklarına rağmen önermeleri sezgisel olan mantıklar iki konuda avantaj sağlayabilir: zaman karmaşıklığı düşük olan akıl yürütme ve doğal dildeki ifadelerle daha yüksek bir uyuşma oranı. Tarihte bilinen ilk biçimsel mantık olan tasımsal mantık o kadar sezgiseldir ki bilişsel bilimciler insanın akıl yürütme şeklini anlamak ve modellemek için sıklıkla bu mantığa başvurur. Burada problem şu ki, bu mantığın sözdizim ve anlambilimi, bilgi temsili için pek yeterli değildir. Bu yüzden tasımsal mantığın önermeleri, avantajlı yanlarından olabildiğince az feragat ederek kullanışlı ek özelliklerle donatılarak genelleştirilmelidir. Bu tez çalışmasının amacı, genelleştirilmiş tasımsal önermeler üzerinde düşük zaman karmaşıklı ve tamamen otomatikleştirilmiş tümdengelim uygulamasına temel oluşturacak olan mantığı ve çatıyı tasarlamak suretiyle tasımsal akıl yürütme ve otomatik akıl yürütme arasındaki uçurumu kapatmaktır. Bu tez çalışmasında, pratik bir genelleştirilmiş tasımsal mantık ailesi tanımlayıp, bu mantıkların teorik özelliklerini ve diğer bazı mantıklarla olan ilişkilerini irdeledik ve alternatif temsil ve yöntemler içeren kapsamlı çatılar tanımladık.

To my cherished mother and beloved sister:
without their amazing companionship and countless sacrifices over the years,
I wouldn't be who I am.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

LIST OF ABBREVIATIONS

SL . . . . . Syllogistic Logic
DL . . . . . Description Logic
FOL . . . . . First Order Logic
CPS . . . . . Categorical Polysyllogism
`TraSyl` . . . . . Traditional Syllogistic Logic
`PolSyl` . . . . . CPSs with Atomic Terms
`NegSyl` . . . . . CPSs with Possibly Negated Atomic Terms
`ComSyl` . . . . . CPSs with Complex Terms
$\texttt{ComSyl}^{+}$ . . . . . `ComSyl` with Individuals

CHAPTER 1

INTRODUCTION

1.1. Motivation

Reasoning, the process of drawing conclusions, is not only critical for natural intelligence but also a core topic for artificial intelligence. Figure 1.1 illustrates the other foremost components of artificial intelligence, along with reasoning.

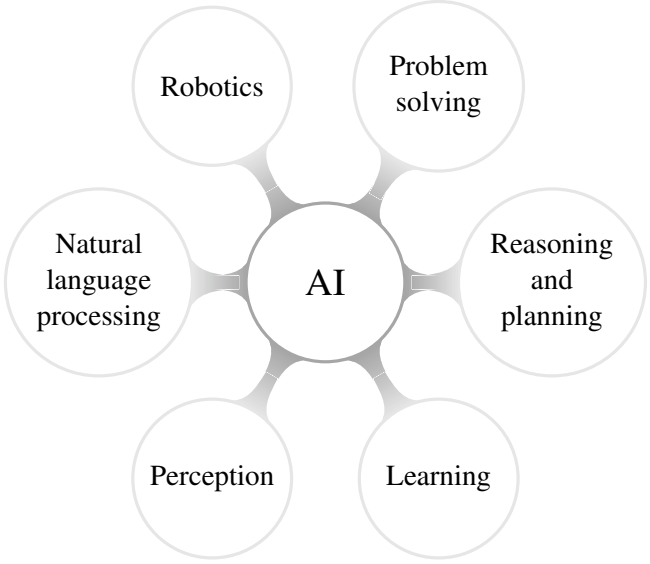


Figure 1.1. Primary components of artificial intelligence

Logic, the science of correct reasoning, is at the heart of all scientific disciplines. Figure 1.2 illustrates this fact with a sample of sciences.

Since the development of modern logics in 19th century, the first known formal study of logic, Aristotle's syllogism, has been considered obsolete by many, due to its limited expressivity.

Nonetheless, syllogistic reasoning is still an active research topic in cognitive science for the purpose of understanding and modeling human reasoning. The reason is that reflexive reasoning capabilities of humans are limited: humans have cognitive biases and

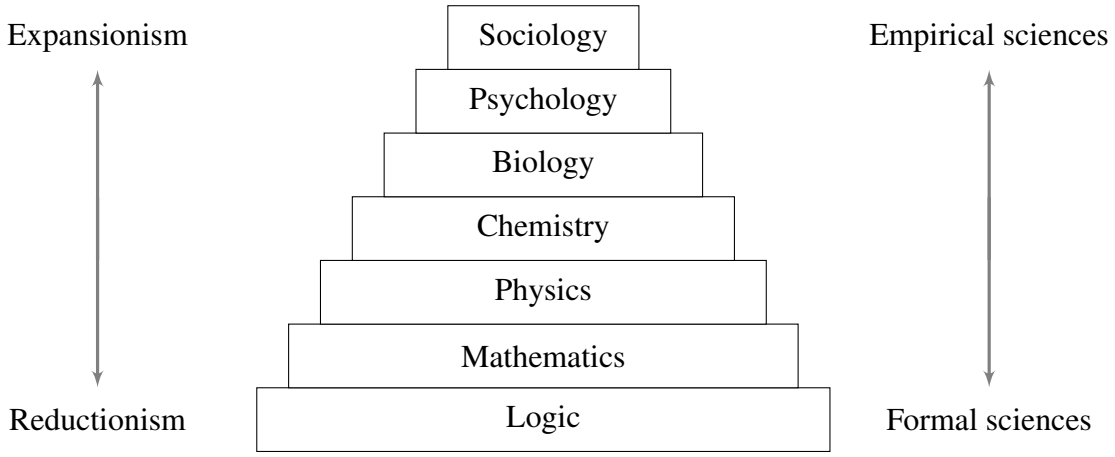


Figure 1.2. Hierarchy of selected scientific disciplines

tend to fall into logical fallacies easily.

At the same time, modern times witness a revival of the syllogistic logic: the current studies on syllogisms include various extensions, detailed analyzes, reasoning algorithms and more.

From computer science point of view, the main problems of more expressive logics are that they suffer from algorithmic complexity and they are unnatural to humans.

The essential parts of automated reasoning, reasoning which is done by computers without any human guidance, are the known facts called premises and the algorithms which lead to new facts called conclusions. The problems of complexity and unnaturalness inconvenience both of these essential parts:

- The unnaturalness of those logics prevents us from developing programs that learn facts automatically by exploiting the big (and cheap) data of natural language and even makes it exhausting to define handcrafted facts.
- The complexity of those logics makes it infeasible to conclude results in a reasonable time.

A useful family of generalized syllogistic logics can be a solution to overcome both of these difficulties: syllogisms correspond to simple sentences in natural languages and promise algorithms with lower time complexity.

1.2. Thesis Definition

Informative definition: This thesis defines a system which derives new facts (called conclusions) from known facts (called premises). Figure 1.3 shows how the overall system with a very simple example can be seen from a high-level perspective.

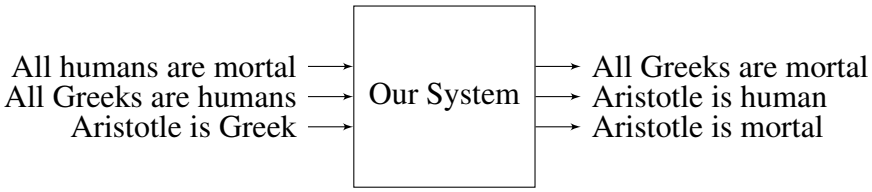


Figure 1.3. The overall system with an informative example

Note that this figure is over-simplified: despite what the visualization suggests, the conclusion candidates, or queries, are actually taken as inputs along with the premises, and the system decides whether it is possible to conclude them or not. Otherwise, there would be possibly infinite number of conclusions. The actual system can easily be adapted to this form by generating query proposals by enumerating interesting patterns with respect to any criteria. The overall efficiency would be arguable, though. The second simplification here is that the propositions, both premises and conclusions, are shown as English sentences. In fact, they are represented in a formal language.

Normative definition: The primary objective of this thesis work is to perform time-efficient fully-automated deduction over generalized syllogistic propositions. Figure 1.4 shows a run of the system with a realistic example.

The goals of this thesis are:

Goal 1: Formally define a practical family of generalized syllogistic logics with different levels of expressivity by combining, improving and extending recent works on syllogisms.

Goal 2: Investigate theoretical properties of the generalized syllogistic logics including expressivity and complexity, and reveal their relationships with the other logics.

Goal 3: Design and implement a comprehensive yet easy-to-use framework for knowledge representation.

Goal 4: Design and implement a high-performance framework for automated deduction.

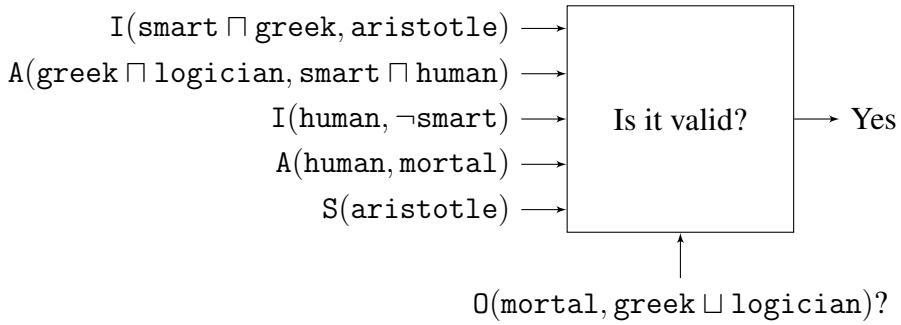


Figure 1.4. The overall system with a normative example

1.3. Related Work

The family of description logics (DLs) (Rudolph (2011)), (Krötzsch et al. (2012)), (Baader et al. (2017)) are perhaps the most successful example of knowledge representation formalisms. It includes many logics with different expressivities, which vary from the lightweight DLs such as DL-Lite (Calvanese et al. (2007)) or $\mathcal{EL}^{++}$ (Baader et al. (2008)) to the typical DLs such as $\mathcal{ALC}$ (Donini and Massacci (2000)) or $\mathcal{ALCO}$ (Grimm and Hitzler (2009)) to the more expressive DLs such as $\mathcal{SHOIQ}$ (Kazakov and Motik (2006)) or $\mathcal{SROIQ}$ (Horrocks et al. (2006)) to other extensions such as fuzzy (Moral et al. (2017)) or temporal (Lutz et al. (2008)) DLs. Their theoretical properties are well-studied. These logics have applications in a wide range of domain from semantic web to medical informatics to software engineering among many others (Baader et al. (2010)). DL reasoners include Racer, FaCT++, Pellet, JFact, HermiT, Konclude and many others.

On the other hand, there has been a long history of studies on syllogistic logics (SLs). Aristotle's SL, the first known formal logic study in history, is sometimes considered as the first formal study in artificial intelligence. It was state-of-the-art until the 19th century (Russell and Norvig (2009)). In the 19th century, modern logics superseded Aristotle's syllogisms (Frege (1879)), (Hammer (1998)).

However, modern logics, first-order logic (FOL) and its extensions, suffer from semi-decidability (i.e. in finite time, no algorithm can prove contradictions in FOL). Furthermore, homophonic theories should be preferred (Evans (1977)). However, FOL is unnatural to the humans: its syntax bears no resemblance to the syntax of the natural language sentences (Quine (1986)). A candidate for these homophonic theories is naturally, the chronologically first logics, SLs.

Today, we witness a revival of SLs. Extentions such as fuzzy quantification (Schwartz (2014)), (Pereira-Fariña et al. (2014)), (Murinová and Novák (2016)), indefinite terms (Pratt-Hartmann and Moss (2009)), (Moss (2011a)), (Alvarez and Correia (2012)), (Alvarez-Fontecilla (2016)), complex terms (Çine and Kumova (2017)), (Çine (2018)), non-categorical roles (Nishihara et al. (1990)), (Moss (2010)), (Moss (2011b)), intersecting adjectives (Moss (2009)), relative clauses (Pratt-Hartmann and Moss (2009)), logical connectives (van Rooij (2010)), (van Rooij (2012)) are added to SLs in order to acquire more expressive SLs.

CHAPTER 2

BACKGROUND

2.1. Automated Deduction and Knowledge Representation

Reasoning is the process of drawing conclusions from facts. Types of reasoning usually fall into two categories:

- Inductive reasoning, or induction is the process of risky generalizations of known facts. Scientific theories are developed thanks to inductions. Statistics is the main tool for this kind of reasoning. A strong induction means that if the premises are true, the conclusion is true with a high probability. An example of strong inductive arguments is: *a human DNA consists of 4 types of nucleotides, a worm DNA consists of 4 types of nucleotides, ..., therefore all animal DNAs consist of 4 types of nucleotides.*

- Deductive reasoning, or deduction is making safe conclusions out of known facts. Mathematical proofs are of this type. Logic is the main tool for this kind of reasoning. A valid deduction means that when the premises are true, it is impossible that the conclusion is false. A well-known example of valid deductive arguments is: *all humans are mortal, Socrates is human, therefore Socrates is mortal.*

Automated reasoning is reasoning which is done by computers without any human guidance.

Automated Deduction Automated deduction task has several forms such as consistency detection, entailment checking or query answering. An algorithmic solution to one of them is usually sufficient to develop algorithms for the others trivially.

A proof system is sound if everything that the system concludes is true given the premises, and is complete if everything that is true given the premises can be concluded by the system. The vast majority of studies on automated reasoning focuses on sound and complete deduction as opposed to approximate reasoning. In this work, we study sound and complete deduction.

The inflexible nature of computers bring the problem of knowledge representation into the scene: a solution to automated reasoning problem becomes practical only if the knowledge is stored structurally with consistent semantics.

Knowledge Representation In knowledge representation, there is a trade-off between expressive power and algorithmic complexity (Brachman and Levesque (1984)): when a formalism is equipped with new features in order to represent different kinds of knowledge, the algorithms for reasoning becomes more time-consuming. Because of this, there is a great variety of formalisms for knowledge representation.

The formalisms for knowledge representation include ontological approaches such as description logic and rule-based approaches such as Datalog.

2.2. Traditional Syllogistic Logic

There are four quantifiers in the traditional syllogistic logic (`TraSyl`): *all*, *no*, *some* and *some not*. They are abbreviated as *A*, *E*, *I* and *O*, respectively. A proposition consists of a quantifier and two terms called *subject* and *predicate*. The types of propositions are:

1. (universal affirmative) All S are P,
2. (universal negative) No S are P,
3. (particular affirmative) Some S are P,
4. (particular negative) Some S are not P.

A syllogism contains 2 premises which contain exactly one common term called the *middle term*. For example:

1. (major premise) All horses are animals.
2. (minor premise) Some animals are cute.

Here, the middle term is “animals”. The other term of the major premise, “horses”, is called the *major term*, and the other term of the minor premise, “cute”, is called the *minor term*.

Minor term must be the subject of the conclusion, and major term must be the predicate of the conclusion. A valid syllogism example is: *All humans are mortal. All engineers are humans. Therefore, all engineers are mortal.*

Note that the validity depends on the position of terms, not the meaning of terms. The following syllogism has the same structure, thus it is also valid: *All M are P. All S are M. Therefore, all S are P.*

A mood specifies the quantifiers of a syllogism using their abbreviations. For example the mood of the syllogism above is AAA. Note that there are only $4^3 = 64$ moods. The following syllogism is a valid example of EIO mood: *No P are M. Some M are S. Therefore, some S are not P.*

A figure specifies the position of the terms in a syllogism. Figure 2.1 illustrates all 4 figures of syllogisms.

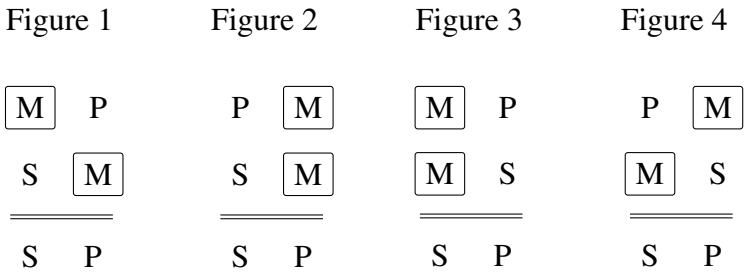


Figure 2.1. Syllogistic figures

The whole structure of a syllogism is called its form. A form consists of a mood and a figure. Therefore, there are $64 \times 4 = 256$ forms only. A form is a compact representation of a syllogism. For example, OIO-1 is an invalid syllogism that expands to the following argument: *Some M are not P. Some S are M. Therefore, some S are not P.*

A famous issue with `TraSyl` is the *existential import* problem: does *all S are P* imply *some S are P*? In other words, do we mean there is at least one S when we say *all S are P*? The answer depends on what we mean by *all*. Because natural language is ambiguous. Intuitively, the answer is "yes" (i.e. the set of X is not empty). However, in the modern predicate logic, the answer is "no" (i.e. the set of X may be empty).

`TraSyl` is the base for extensions in this thesis work.

2.3. Description Logics

In DLs, a knowledge base, or ontology, typically consists of a terminological part (TBox) for representing relationships between concepts and an assertional part (ABox) for representing membership of an individual to a concept or relationships between two individuals. Types of TBoxes are the empty TBox, acyclic TBox, cyclic TBox and general TBox.

We will only give definitions of the most relevant DLs, $\mathcal{ALC}$ and $\mathcal{ALCO}$ with general TBoxes.

Let $\mathtt{X}$ be a concept name, $\mathtt{r}$ a role, and $\mathtt{C}$ and $\mathtt{D}$ concepts. The production rule in $\mathcal{ALC}$ is:

$$\mathtt{C}, \mathtt{D} ::= \bot \mid \top \mid \mathtt{X} \mid \neg\mathtt{C} \mid \mathtt{C} \sqcap \mathtt{D} \mid \mathtt{C} \sqcup \mathtt{D} \mid \exists\mathtt{r}.\mathtt{C} \mid \forall\mathtt{r}.\mathtt{C}$$

Let $\mathtt{x}$ be an individual. $\mathcal{ALCO}$ introduces nominals:

$$\mathtt{C}, \mathtt{D} ::= \bot \mid \top \mid \mathtt{X} \mid \neg\mathtt{C} \mid \mathtt{C} \sqcap \mathtt{D} \mid \mathtt{C} \sqcup \mathtt{D} \mid \exists\mathtt{r}.\mathtt{C} \mid \forall\mathtt{r}.\mathtt{C} \mid \{\mathtt{x}\}$$

An ABox contains concept assertions which involve a concept and an individual (e.g. Aristotle is a human: $\mathtt{Human}(\mathtt{Aristotle})$) and role assertions which involve a role and two individuals (e.g. Aristotle is a student of Plato: $\mathtt{studentOf}(\mathtt{Aristotle}, \mathtt{Plato})$).

A general TBox contains general concept inclusions between two concepts (e.g. A game player is a human who plays some games: $\mathtt{GamePlayer} \sqsubseteq \mathtt{Human} \sqcap \exists\mathtt{play}.\mathtt{Game}$).

Semantics of $\mathcal{ALC}$ and $\mathcal{ALCO}$ are similar to those of SLs. Thus, we will omit them for saving space.

CHAPTER 3

A FAMILY OF GENERALIZED SYLLOGISTIC LOGICS

3.1. Introduction

We formally define a family of SLs with different levels of expressivity by generalizing the traditional categorical syllogisms in several dimensions. These dimensions include the number of premises, expressivity of terms, and types of propositions.

In increasing order of expressive power, the generalized SLs are:

1. $\texttt{PolSyl}$ (Categorical polysyllogisms (CPSs) with atomic terms),
2. $\texttt{NegSyl}$ (CPSs with possibly negated atomic terms),
3. $\texttt{ComSyl}$ (CPSs with any kind of complex terms),
4. $\texttt{ComSyl}^{+}$ ($\texttt{ComSyl}$ with individuals).

Arbitrary Number of Premises In traditional syllogisms, typically 2 premises are used. As in polysyllogisms, in generalized syllogisms, any number of premises can be used. For example, $\vDash \texttt{A}(\texttt{X}, \texttt{X})$ is a valid generalized syllogism with 0 premise (i.e. $\texttt{A}(\texttt{X}, \texttt{X})$ is a tautology), and $\texttt{A}(\texttt{Z}, \texttt{T})$, $\texttt{A}(\texttt{Y}, \texttt{Z})$, $\texttt{A}(\texttt{X}, \texttt{Y}) \vDash \texttt{A}(\texttt{X}, \texttt{T})$ is a valid generalized syllogism with 3 premises.

Complex Terms In traditional syllogisms, all terms are atomic. For example, an atomic term can be $\texttt{cars}$, $\texttt{motorcycles}$, $\texttt{red}\,(\texttt{things})$, or $\texttt{expensive}\,(\texttt{things})$. An atomic term can also be a complex-looking term such as $\texttt{red cars}$. However, it is not possible to conclude $\texttt{I}(\texttt{car}, \texttt{expensive})$ or $\texttt{I}(\texttt{car}, \texttt{red})$ from $\texttt{I}(\texttt{red car}, \texttt{expensive})$ if $\texttt{red car}$ is an atomic term.

In generalized syllogisms, new terms are produced from the atomic terms using a production rule. For example, a complex term can be $\texttt{expensive} \sqcap \texttt{motorcycle}$ ("expensive motorcycles"), $\neg\, \texttt{expensive} \sqcap \texttt{red} \sqcap \texttt{car}$ ("non-expensive red cars"), or $\texttt{car} \sqcup \texttt{motorcycle}$ ("cars and motorcycles"). It is possible to conclude $\texttt{I}(\texttt{car}, \texttt{expensive})$ and $\texttt{I}(\texttt{car}, \texttt{red})$ from $\texttt{I}(\texttt{red} \sqcap \texttt{car}, \texttt{expensive})$.

Complex terms implicitly cover indefinite terms (such as $\neg\texttt{cat}$ which means "non-cats", or everything that is not a cat), indirect syllogisms (i.e. subject and predicate terms of conclusion are switched as in $\texttt{A}(\texttt{Y},\texttt{Z})$, $\texttt{I}(\texttt{X},\texttt{Y}) \models \texttt{I}(\texttt{Z},\texttt{X})$), and non-standard figures (such as the figure in $\texttt{A}(\texttt{X},\texttt{Z})$, $\texttt{I}(\texttt{Y},\texttt{X}) \models \texttt{I}(\texttt{X},\texttt{Z})$ which contains the middle term in the conclusion).

Individuals Propositions about individuals do not fit into the standard notation for propositions of traditional syllogisms (e.g. "Socrates is human"). In generalized syllogisms, individuals are treated as singleton sets in order to prevent quantification problems.

Asserting individuality of a term can change the validity of a syllogism. For example $\texttt{I}(\texttt{Socrates},\texttt{human}) \not\models \texttt{A}(\texttt{Socrates},\texttt{human})$ and $\texttt{I}(\texttt{Socrates},\texttt{human}), \texttt{E}(\texttt{human},\texttt{car}) \not\models \texttt{E}(\texttt{Socrates},\texttt{car})$. However, if it is known that there is only one Socrates, we can conclude $\texttt{A}(\texttt{Socrates},\texttt{human})$ and $\texttt{E}(\texttt{Socrates},\texttt{car})$, i.e. $\texttt{A}(\texttt{Socrates},\texttt{human}), \texttt{S}(\texttt{Socrates}) \models \texttt{E}(\texttt{Socrates},\texttt{Car})$ and $\texttt{I}(\texttt{Socrates},\texttt{human}), \texttt{E}(\texttt{human},\texttt{car}), \texttt{S}(\texttt{Socrates}) \models \texttt{E}(\texttt{Socrates},\texttt{car})$.

3.2. Syntax

Atomic Term An atomic term is an indivisible label which stands for a set of individuals.

Atomic terms are often simple words such as a noun or an adjective in natural language. For example, cat (represents the set of cats in our domain), smart (represents the set of smart things in our domain) and Socrates (represents the singleton set that contains the person Socrates) are atomic terms.

Negated Atomic Term Let $\texttt{X}$ be an atomic term. A negated atomic term, or an indefinite term is $\neg\texttt{X}$.

For example, $\neg\texttt{cat}$ is a negated atomic term assuming cat is an atomic term.

Possibly-Negated Atomic Term A possibly-negated atomic term is either an atomic term or a negated atomic term.

Complex Term Let $\texttt{C}$ and $\texttt{D}$ be complex terms and $\texttt{X}$ be an atomic term. The production rule for complex terms is defined as:

$$\texttt{C},\texttt{D} ::= \texttt{X} \mid \top \mid \bot \mid \neg\texttt{C} \mid \texttt{C} \sqcap \texttt{D} \mid \texttt{C} \sqcup \texttt{D}$$

For example, $\neg\texttt{smart} \sqcap \texttt{cat}$ is a complex term assuming `smart` and `cat` are also complex terms.

Note that, complex terms inherently include possibly-negated atomic terms.

Quantified Proposition Let `C` and `D` be complex terms. A quantified proposition `P` is defined as:

$$\texttt{P} ::= \texttt{A}(\texttt{C}, \texttt{D}) \mid \texttt{E}(\texttt{C}, \texttt{D}) \mid \texttt{I}(\texttt{C}, \texttt{D}) \mid \texttt{I}(\texttt{C}, \texttt{D})$$

For example, $\texttt{I}(\neg\texttt{expensive} \sqcap (\texttt{car} \sqcup \texttt{motorcycle}), \texttt{red})$ is a quantified proposition assuming `expensive`, `car`, `motorcycle` and `red` are complex terms.

Individuality Proposition Let `C` be a complex term. An individuality proposition `P` is defined as:

$$\texttt{P} ::= \texttt{S}(\texttt{C})$$

For example, $\texttt{S}(\texttt{Socrates})$ is an individuality proposition assuming `Socrates` is a complex term.

Relational Proposition A relational proposition, or a relation is either a quantified proposition or an individuality proposition.

Our SLs are defined through syntactic restrictions on defined concepts. Table 3.1 shows definitions of these logics. Note that the symbols `X`, $\neg$`X` and `C` stand for atomic terms, negated atomic terms and complex terms, respectively. Additionally, propositions of type `S`, or individuality propositions may be allowed only as assertions, not as queries. The other types of propositions, or quantified propositions may be assertions or queries.

3.3. Semantics

Let `C` and `D` be complex terms. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set of individuals $\Delta^{\mathcal{I}}$ called the domain and an interpretation function $\cdot^{\mathcal{I}}$ that maps `C` to a set $\texttt{C}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ such that

1. $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$,

Table 3.1. Syntactic restrictions

Logic	Allowed Terms			Allowed Propositions				
	X	¬X	C	A	E	I	O	S
PolSyl	✓			✓	✓	✓	✓	
NegSyl	✓	✓		✓	✓	✓	✓	
ComSyl	✓	✓	✓	✓	✓	✓	✓	
ComSyl$^+$	✓	✓	✓	✓	✓	✓	✓	✓

2. $(\neg \mathtt{C})^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus \mathtt{C}^{\mathcal{I}}$,

3. $(\mathtt{C} \sqcap \mathtt{D})^{\mathcal{I}} = \mathtt{C}^{\mathcal{I}} \cap \mathtt{D}^{\mathcal{I}}$.

$\bot$ and $\mathtt{C} \sqcup \mathtt{D}$ can be seen as syntactic sugars which are synonyms of $\neg\top$ and $\neg(\neg\mathtt{C} \sqcap \neg\mathtt{D})$, respectively. Thus it is the case that $\bot^{\mathcal{I}} = \emptyset$ and $(\mathtt{C} \sqcup \mathtt{D})^{\mathcal{I}} = \mathtt{C}^{\mathcal{I}} \cup \mathtt{D}^{\mathcal{I}}$.

A non-existing term corresponds to an empty set and an existing term corresponds to a non-empty set in the interpretation.

Table 3.2 shows all possible types of propositions and their corresponding conditions. A proposition holds if and only if its corresponding condition is met.

Table 3.2. Semantics of propositions

Proposition	Condition	English
$\mathcal{I} \vDash \mathtt{A}(\mathtt{C}, \mathtt{D})$	$\mathtt{C}^{\mathcal{I}} \setminus \mathtt{D}^{\mathcal{I}} = \emptyset$	All C are D
$\mathcal{I} \vDash \mathtt{E}(\mathtt{C}, \mathtt{D})$	$\mathtt{C}^{\mathcal{I}} \cap \mathtt{D}^{\mathcal{I}} = \emptyset$	No C are D
$\mathcal{I} \vDash \mathtt{I}(\mathtt{C}, \mathtt{D})$	$\mathtt{C}^{\mathcal{I}} \cap \mathtt{D}^{\mathcal{I}} \neq \emptyset$	Some C are D
$\mathcal{I} \vDash \mathtt{O}(\mathtt{C}, \mathtt{D})$	$\mathtt{C}^{\mathcal{I}} \setminus \mathtt{D}^{\mathcal{I}} \neq \emptyset$	Some C are not D
$\mathcal{I} \vDash \mathtt{S}(\mathtt{C})$	$\lvert\mathtt{C}^{\mathcal{I}}\rvert = 1$	C is an individual

Syntax and semantics of syllogistic logics are intuitive. An example of valid syllogisms $\mathtt{I}(\mathtt{Turkish} \sqcap \mathtt{engineer}, \mathtt{smart} \sqcap \mathtt{person}), \mathtt{A}(\mathtt{person}, \mathtt{mammal}) \vDash \mathtt{I}(\mathtt{mammal}, \mathtt{engineer})$ corresponds to the following argument in English: *Some Turkish engineers are smart people. All people are mammals. Therefore, some mammals are engineers.*

3.4. Theoretical Properties

In this section, we will reveal theoretical properties of SLs and their relationships with the DLs.

Expressiveness Let $\mathtt{X}$ and $\mathtt{Y}$ be atomic terms. All 16 relational propositions in `NegSyl` constitute 8 groups of semantically equivalent propositions:

1. $\mathtt{A}(\mathtt{X},\mathtt{Y}) \equiv \mathtt{E}(\mathtt{X},\neg\mathtt{Y}) \equiv \nexists(\mathtt{X}\sqcap\neg\mathtt{Y})$
2. $\mathtt{E}(\mathtt{X},\mathtt{Y}) \equiv \mathtt{A}(\mathtt{X},\neg\mathtt{Y}) \equiv \nexists(\mathtt{X}\sqcap\mathtt{Y})$
3. $\mathtt{A}(\neg\mathtt{X},\mathtt{Y}) \equiv \mathtt{E}(\neg\mathtt{X},\neg\mathtt{Y}) \equiv \nexists(\neg\mathtt{X}\sqcap\neg\mathtt{Y})$
4. $\mathtt{E}(\neg\mathtt{X},\mathtt{Y}) \equiv \mathtt{A}(\neg\mathtt{X},\neg\mathtt{Y}) \equiv \nexists(\neg\mathtt{X}\sqcap\mathtt{Y})$
5. $\mathtt{I}(\mathtt{X},\mathtt{Y}) \equiv \mathtt{O}(\mathtt{X},\neg\mathtt{Y}) \equiv \exists(\mathtt{X}\sqcap\mathtt{Y})$
6. $\mathtt{O}(\mathtt{X},\mathtt{Y}) \equiv \mathtt{I}(\mathtt{X},\neg\mathtt{Y}) \equiv \exists(\mathtt{X}\sqcap\neg\mathtt{Y})$
7. $\mathtt{I}(\neg\mathtt{X},\mathtt{Y}) \equiv \mathtt{O}(\neg\mathtt{X},\neg\mathtt{Y}) \equiv \exists(\neg\mathtt{X}\sqcap\mathtt{Y})$
8. $\mathtt{O}(\neg\mathtt{X},\mathtt{Y}) \equiv \mathtt{I}(\neg\mathtt{X},\neg\mathtt{Y}) \equiv \exists(\neg\mathtt{X}\sqcap\neg\mathtt{Y})$

As a result of the correspondence above, using cardinality propositions, a term is an element of $\{\mathtt{X}\sqcap\mathtt{Y},\ \mathtt{X}\sqcap\neg\mathtt{Y},\ \neg\mathtt{X}\sqcap\mathtt{Y},\ \neg\mathtt{X}\sqcap\neg\mathtt{Y}\}$. Special terms $\mathtt{X}$, $\neg\mathtt{X}$ and $\bot$ are derived when the atomic terms are equal (i.e. $\mathtt{X}=\mathtt{Y}$). We will call these terms "regular" when the atomic terms are unequal (i.e. $\mathtt{X}\neq\mathtt{Y}$). Note that these regular terms represent all 4 regions in Venn diagram of two atomic terms.

Commutative property of intersection, unequal atomic terms in the term expressions, and 4 regions of two atomic terms reveal the true expressivity of `NegSyl`: it allows for asserting emptiness or non-emptiness of $2\mathtt{t}^2+1$ pairwise unequal sets: The bottom term $\bot$, $\mathtt{t}$ atomic terms, $\mathtt{t}$ negated atomic terms, and $4\times\frac{\mathtt{t}\times(\mathtt{t}-1)}{2}$ regular terms.

Expressiveness of `PolSyl` is calculated very similarly. It can express 3 regions out of 4 regions between two atomic terms: $\{\mathtt{X}\sqcap\mathtt{Y},\ \mathtt{X}\sqcap\neg\mathtt{Y},\ \neg\mathtt{X}\sqcap\mathtt{Y}\}$. $\mathtt{X}$ and $\bot$ can be derived when $\mathtt{X}\neq\mathtt{Y}$. In total, $\frac{3}{2}\mathtt{t}^2-\frac{1}{2}\mathtt{t}+1$ pairwise unequal sets can be represented.

$\mathtt{t}$ atomic terms constitute $2^{\mathtt{t}}$ regions in a Venn diagram. `ComSyl` and $\mathtt{ComSyl}^+$ are able to express all subsets of those regions: there are $2^{2^{\mathtt{t}}}$ pairwise unequal sets to represent.

Lemma 3.4.1 *The following propositions are all true:*

- `TraSyl` *is a fragment of any of* $\{\texttt{PolSyl}, \texttt{NegSyl}, \texttt{ComSyl}, \texttt{ComSyl}^+\}$.
- `PolSyl` *is a fragment of any of* $\{\texttt{NegSyl}, \texttt{ComSyl}, \texttt{ComSyl}^+\}$.
- `NegSyl` *is a fragment of any of* $\{\texttt{ComSyl}, \texttt{ComSyl}^+\}$.
- `ComSyl` *is a fragment of any of* $\{\texttt{ComSyl}^+\}$.

Proof A syntactic restriction on a logic defines a fragment of that logic. These logics are defined through syntactic restrictions in the previous sections. □

Lemma 3.4.2 `PolSyl`, `NegSyl`, `ComSyl` *and* $\texttt{ComSyl}^+$ *are monotonic.*

Proof The following facts prove the monotonicity:

1. A proposition in those logics is true only if it is true in all possible worlds. Thus, that proposition must be true in any subset of those possible worlds.
2. Algorithm 6 shows that each premise filters the possible worlds further. This means that, at any iteration over premises, the set of possible worlds is a subset of possible worlds of the previous iteration. □

Theorem 3.4.3 *Consistency detection both in* `PolSyl` *and in* `NegSyl` *is tractable.*

Proof Algorithm 2 along with algorithm 3 defines a sound and complete algorithm for consistency detection in `NegSyl` which will always terminate in polynomial time (PTIME, or P in short).

As discussed before, `PolSyl` is a fragment of `NegSyl`. As a result, consistency detection in `PolSyl` is also tractable. □

Theorem 3.4.4 `ComSyl` *and* $\texttt{ComSyl}^+$ *are fragments of* $\mathcal{ALC}$ *and* $\mathcal{ALCO}$ *with general TBoxes, respectively.*

Proof Transformation rules for `ComSyl` and $\texttt{ComSyl}^+$ to $\mathcal{ALC}$ and $\mathcal{ALCO}$ are defined in section 5.6. □

Corollary 3.4.5 `PolSyl`, `NegSyl`, `ComSyl` *and* $\texttt{ComSyl}^+$ *fragments of FOL.*

Proof $\texttt{ComSyl}^+$ is an extension of each of the other logics. Thus, it is sufficient to prove the proposition only for $\texttt{ComSyl}^+$.

It is known that $\mathcal{ALCO}$ is a fragment of FOL. We have shown that $\texttt{ComSyl}^+$ is a fragment of $\mathcal{ALCO}$. Thus, $\texttt{ComSyl}^+$ is a fragment of FOL.

A more direct proof the transformation rules defined in section 5.5. □

Corollary 3.4.6 *Consistency detection both in* $\texttt{ComSyl}$ *and in* $\texttt{ComSyl}^+$ *is in EXPTIME.*

Proof It is already stated that consistency detection in $\mathcal{ALC}$ and $\mathcal{ALCO}$ with respect to general TBoxes is EXPTIME-complete. We proved that $\texttt{ComSyl}$ and $\texttt{ComSyl}^+$ are fragments of those logics. Thus, they are in EXPTIME. □

Corollary 3.4.7 $\texttt{PolSyl}$, $\texttt{NegSyl}$, $\texttt{ComSyl}$ *and* $\texttt{ComSyl}^+$ *are decidable.*

Proof $\texttt{ComSyl}^+$ is an extension of each of the other logics. Thus, proving decidability of $\texttt{ComSyl}^+$ is sufficient to prove the whole statement.

Decidability of $\texttt{ComSyl}^+$ is proved twice:

1. It is already shown that $\texttt{ComSyl}^+$ is in EXPTIME, thus it is decidable.
2. Algorithm 6 finite procedure that proves the decidability of $\texttt{ComSyl}^+$. □

Corollary 3.4.8 $\texttt{ComSyl}$ *and* $\texttt{ComSyl}^+$ *are syntactic variants of* $\mathcal{ALC}$ *and* $\mathcal{ALCO}$*, respectively, when the only role which appears in the ontology is the categorical role be.*

Proof The only syntactic difference between $\texttt{ComSyl}$ terms and $\mathcal{ALC}$ concepts is that $\mathcal{ALC}$ allow us to define concepts via non-categorical roles. When the only role is the categorical role *be* (or, *beSubsetOf*) the semantics of $\exists r.C$ and $\forall r.C$ can be represented via propositions of type $\texttt{I}$ and $\texttt{A}$, respectively.

The relationship between $\texttt{ComSyl}^+$ and $\mathcal{ALCO}$ is very similar to that of $\texttt{ComSyl}$ and $\mathcal{ALC}$. The only addition is the propositions of type $\texttt{S}$ which correspond to nominals in DLs. □

3.5. Existential Configuration

In traditional syllogisms, it is not clear whether $\texttt{A}(\texttt{X}, \texttt{Y})$ implies $\texttt{I}(\texttt{X}, \texttt{Y})$ or not. The implication depends on the personal interpretation of the quantifiers. This problem is

called "existential import" problem. Generalized syllogisms are similar to FOL in this sense: $\mathtt{A}(\mathtt{X},\mathtt{Y})$ does not imply $\mathtt{I}(\mathtt{X},\mathtt{Y})$. In other words, $\mathtt{X}$ may correspond to the empty set in the interpretation. However, most people (e.g. Aristotle) tend to think differently.

In order to make the generalized syllogisms more intuitive, we configure quantifiers with *existential indices*: an existential index is a subset of $\{\upsilon,\kappa\}$. Here, υ indicates the existence of the subject term and κ indicates the existence of the predicate term. For example, both $\mathtt{X}$ and $\mathtt{Y}$ may be non-existing terms even if $\mathtt{A}_{\emptyset}(\mathtt{X},\mathtt{Y}) \equiv \mathtt{A}(\mathtt{X},\mathtt{Y})$ holds, but $\mathtt{X}$ is an existing term if $\mathtt{A}_{\{\upsilon\}}(\mathtt{X},\mathtt{Y})$ holds. This means that, for example, $\mathtt{A}_{\emptyset}(\mathtt{X},\mathtt{Y})$ holds if and only if $\mathtt{X}^{\mathcal{I}} \setminus \mathtt{Y}^{\mathcal{I}} = \emptyset$ but $\mathtt{A}_{\{\upsilon\}}(\mathtt{X},\mathtt{Y})$ holds if and only if $\mathtt{X}^{\mathcal{I}} \setminus \mathtt{Y}^{\mathcal{I}} = \emptyset$ and $\mathtt{X}^{\mathcal{I}} \neq \emptyset$.

A proposition with a configured quantifier is only a syntactic sugar for the conjunction of a couple of quantified propositions. For example, $\mathtt{A}_{\{\upsilon\}}(\mathtt{X},\mathtt{Y}) \equiv \mathtt{A}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{I}(\mathtt{X},\mathtt{X})$.

Table 3.3 shows all configured quantifiers and their subjective classifications. Note that the quantifiers $\mathtt{I}$ and $\mathtt{O}$ cannot be configured without υ: both in modern interpretation and in traditional interpretation, they imply the existence of subject. The existential index $\{\kappa\}$ is seemingly not very useful. People probably mean $\{\upsilon\}$ or $\{\upsilon,\kappa\}$. There are equivalent configured quantifiers such as $\mathtt{A}_{\{\upsilon\}}$ and $\mathtt{A}_{\{\upsilon,\kappa\}}$. The reason is the fact that $\mathtt{A}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{I}(\mathtt{X},\mathtt{X}) \equiv \mathtt{A}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{I}(\mathtt{X},\mathtt{X}) \wedge \mathtt{I}(\mathtt{Y},\mathtt{Y})$. In other words, $\mathtt{A}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{I}(\mathtt{X},\mathtt{X})$ implies $\mathtt{I}(\mathtt{Y},\mathtt{Y})$.

Table 3.3. Configured quantifiers

Syntactic Sugar	**Equivalent Conjunction**	**Classification**
$\mathtt{A}_{\emptyset}(\mathtt{X},\mathtt{Y})$	$\mathtt{A}(\mathtt{X},\mathtt{Y})$	Modern
$\mathtt{A}_{\{\upsilon\}}(\mathtt{X},\mathtt{Y}) \equiv \mathtt{A}_{\{\upsilon,\kappa\}}(\mathtt{X},\mathtt{Y})$	$\mathtt{A}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{I}(\mathtt{X},\mathtt{X})$	Traditional
$\mathtt{A}_{\{\kappa\}}(\mathtt{X},\mathtt{Y})$	$\mathtt{A}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{I}(\mathtt{Y},\mathtt{Y})$	Bogus
$\mathtt{E}_{\emptyset}(\mathtt{X},\mathtt{Y})$	$\mathtt{E}(\mathtt{X},\mathtt{Y})$	Modern
$\mathtt{E}_{\{\upsilon\}}(\mathtt{X},\mathtt{Y})$	$\mathtt{E}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{I}(\mathtt{X},\mathtt{X})$	Alternative
$\mathtt{E}_{\{\kappa\}}(\mathtt{X},\mathtt{Y})$	$\mathtt{E}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{I}(\mathtt{Y},\mathtt{Y})$	Bogus
$\mathtt{E}_{\{\upsilon,\kappa\}}(\mathtt{X},\mathtt{Y})$	$\mathtt{E}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{I}(\mathtt{X},\mathtt{X}) \wedge \mathtt{I}(\mathtt{Y},\mathtt{Y})$	Traditional
$\mathtt{I}_{\emptyset}(\mathtt{X},\mathtt{Y})$, $\mathtt{I}_{\{\kappa\}}(\mathtt{X},\mathtt{Y})$	-	Impossible
$\mathtt{I}_{\{\upsilon\}}(\mathtt{X},\mathtt{Y}) \equiv \mathtt{I}_{\{\upsilon,\kappa\}}(\mathtt{X},\mathtt{Y})$	$\mathtt{I}(\mathtt{X},\mathtt{Y})$	Indispensable
$\mathtt{O}_{\emptyset}(\mathtt{X},\mathtt{Y})$, $\mathtt{O}_{\{\kappa\}}(\mathtt{X},\mathtt{Y})$	-	Impossible
$\mathtt{O}_{\{\upsilon\}}(\mathtt{X},\mathtt{Y})$	$\mathtt{O}(\mathtt{X},\mathtt{Y})$	Modern
$\mathtt{O}_{\{\upsilon,\kappa\}}(\mathtt{X},\mathtt{Y})$	$\mathtt{O}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{I}(\mathtt{Y},\mathtt{Y})$	Traditional

$\mathtt{A}_{\{\upsilon\}}(\mathtt{X}, \top)$ can be used to claim that $\mathtt{X}$ is an existing term and $\mathtt{A}_{\emptyset}(\mathtt{X}, \bot)$ can be used to claim that $\mathtt{X}$ is a non-existing term. $\mathtt{X}$ and $\mathtt{Y}$ are equal terms if and only if $\mathtt{A}_{\emptyset}((\mathtt{X} \sqcap \neg \mathtt{Y}) \sqcup (\mathtt{Y} \sqcap \neg \mathtt{X}), \bot)$ holds. $\mathtt{X}$ and $\mathtt{Y}$ are unequal terms if and only if $\mathtt{I}_{\{\upsilon\}}((\mathtt{X} \sqcap \neg \mathtt{Y}) \sqcup (\mathtt{Y} \sqcap \neg \mathtt{X}), \top)$ holds. $\mathtt{X}$ and $\mathtt{Y}$ are disjoint terms if and only if $\mathtt{E}_{\emptyset}(\mathtt{X}, \mathtt{Y})$ holds. $\mathtt{X}$ and $\mathtt{Y}$ are intersecting terms if and only if $\mathtt{I}_{\{\upsilon\}}(\mathtt{X}, \mathtt{Y})$ holds.

CHAPTER 4

REPRESENTATION FRAMEWORK

4.1. Introduction

A knowledge base, or an ontology, or in this case a syllogism, is a set of propositions. In this section, we define a framework for representing a syllogism and its parts: terms and propositions. Figure 4.1 illustrates the high-level components of the framework.

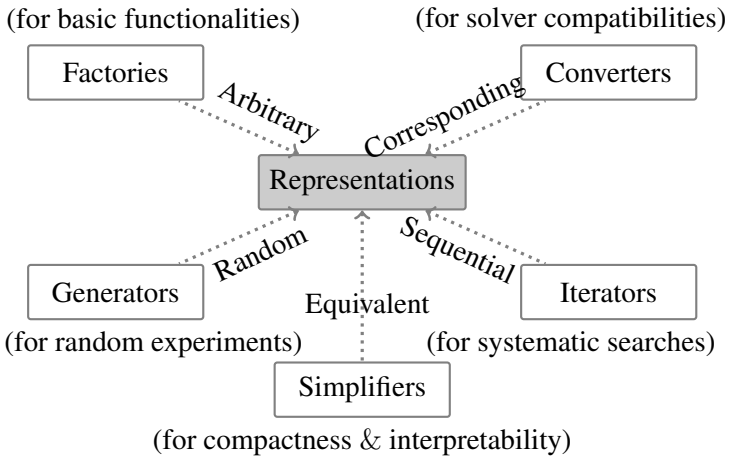


Figure 4.1. Components of the representation framework

Factories, generators, iterators and converters create arbitrary, random, sequential and equivalent representations of terms, propositions and syllogisms.

4.2. Representation of Terms

Terms are the most basic units of the representation framework. Figure 4.2 illustrates the alternative representations for the term $(\neg \mathtt{X}_2 \sqcup \mathtt{X}_3) \sqcap \mathtt{X}_1$ which corresponds to region set $\{\mathtt{1}, \mathtt{5}, \mathtt{7}\}$ when the number of atomic terms $\mathtt{t} = \mathtt{3}$.

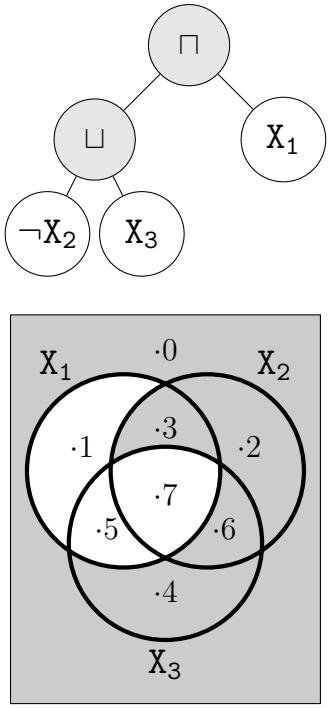


Figure 4.2. A term can be expressed as an expression tree or a region set

4.2.1. Terms as Expression Trees

Expression trees, a special class of binary trees, can be used to represent terms. In expression trees, each inner node contains an operator and each leaf contains an operand. Exceptionally, leaves internalize negations with the purpose of expression simplification. For example, $\neg \mathtt{X}_2 \sqcap \neg \mathtt{X}_4$ is stored instead of $\neg(\mathtt{X}_2 \sqcup \mathtt{X}_4)$. Figure 4.3 illustrates expression tree of the complex term $(\neg \mathtt{X}_2 \sqcup (\mathtt{X}_4 \sqcup \neg \mathtt{X}_0)) \sqcap \mathtt{X}_3$.

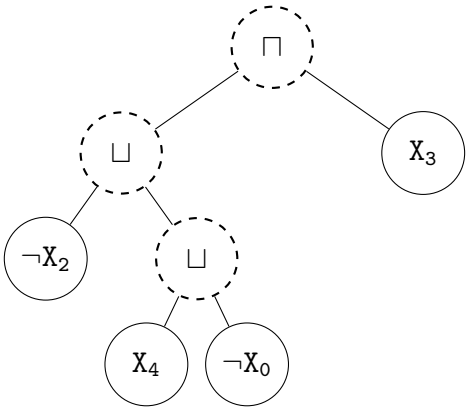


Figure 4.3. An expression tree

Advantages of expression trees over regions sets are that they are more compact in terms of space complexity than region sets and they enable lazy evaluation.

4.2.2. Terms as Region Sets

Algorithm 1 calculates an array of set of region numbers for a given number of atomic terms. For example, if we call the function as in `X = AtomicTerms(3)`, then $\mathtt{X}_1 = \{1, 3, 5, 7\}$, $\mathtt{X}_2 = \{2, 3, 6, 7\}$, $\mathtt{X}_3 = \{4, 5, 6, 7\}$, and $\mathtt{X.length} = 3$. Hereafter, $\mathtt{t}$ will represent the number of atomic terms.

Algorithm 1 Calculation of region sets

1: **function** ATOMICTERMS(t) ▷ t is number of atomic terms
2: **for** $i \leftarrow 1, t$ **do**
3: $X_i \leftarrow \{ \sum_{n \in N} 2^{n-1} \mid N \subseteq \{1, 2, ..., t\}, i \in N \}$
4: **return** X

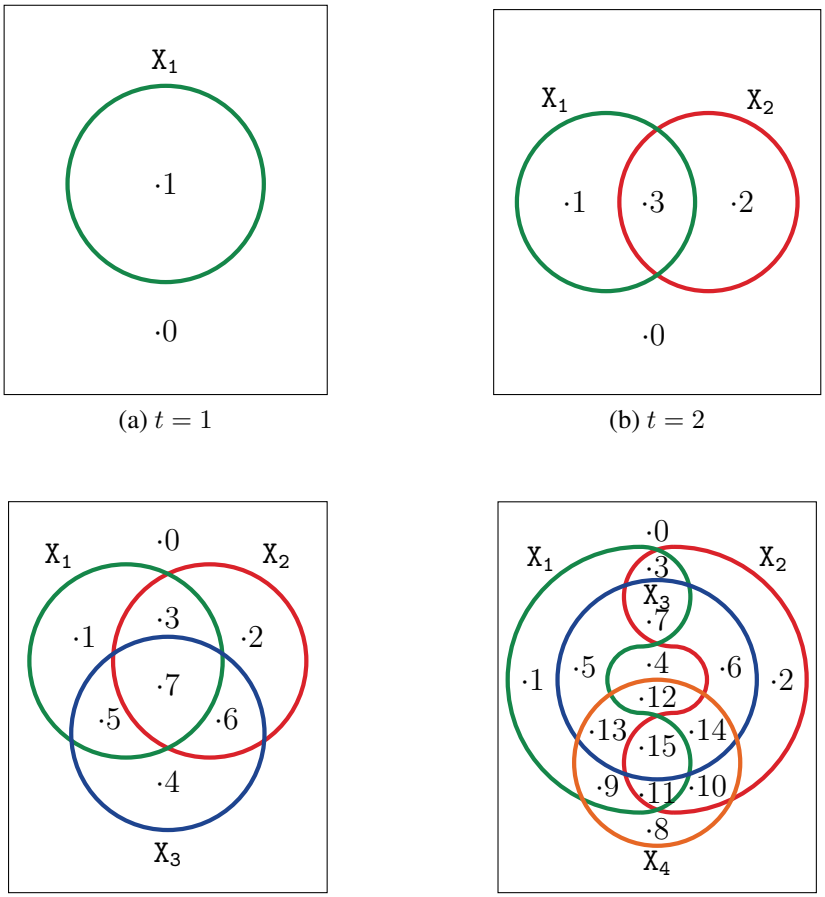
The intuition behind this function is simple: for every $\mathtt{X_i}$, there is a dedicated number $2^{\mathtt{i}-1}$. For example, $2^{3-1} = 4$ is dedicated to $\mathtt{X}_3$. Any region is numbered the sum of the dedicated numbers of those atomic terms that contains this region. For example, when $\mathtt{t} = 3$, $(\mathtt{X}_1 \cap \mathtt{X}_3) \setminus \mathtt{X}_2$ defines a single region that is contained by only $\mathtt{X}_1$ and $\mathtt{X}_3$. Therefore, that region will be numbered $2^{1-1} + 2^{3-1} = 1 + 4 = 5$.

Figure 4.4 shows a visualization of atomic terms and region numbers. It is useful to replace terms with meaningful identifiers (e.g. $\mathtt{human} = \mathtt{X}_1$).

Note that region sets require that t, the number of atomic terms, is pre-determined and assume that it is fixed. An advantage of region sets is that they are always in their simplest form. Along with that, when t is small, visualization of region sets is useful for research and education.

The pre-determined t allows for defining a special term as a syntactic sugar: the surplus term δ expresses everything in the domain except those which are in atomic terms. For example, $\delta = \neg\mathtt{X}_1 \sqcap \neg\mathtt{X}_2 \sqcap \neg\mathtt{X}_3$ when $t = 3$.

The bottom term $\bot$ defines a term that represents the empty set. $\bot$ is defined as a global constant: $\mathtt{VOID} = \{\ \}$. The top term $\top$ defines a term that represents all the domain. $\top$ is defined as a global variable: $\mathtt{UNIVERSE} = \{0, 1, ..., 2^{\mathtt{t}} - 1\}$. Value of `UNIVERSE` does not change unless the value of t is changed. If the value of $\mathtt{t}$ is changed, then the



(a) $t = 1$ (b) $t = 2$

(c) $t = 3$ (d) $t = 4$

Figure 4.4. Universes with different number of atomic terms

atomic terms should be also updated. The surplus term δ defines a term that represents everything in the domain other than the atomic terms. δ is defined as a global constant: $\texttt{EXTRAS} = \{\texttt{0}\}$.

Atomic terms and the surplus term are the building blocks of complex terms. The production of complex terms is done by the set operations such as intersection, union, difference, and complement. For example, if $\texttt{t} = 3$ then $\texttt{X}_1 \cup \overline{\texttt{X}_2} = \{0, 1, 3, 4, 5, 7\}$. It is true that $\texttt{UNIVERSE} = \bigcup_{i=1}^{t} \texttt{X}_i \cup \texttt{EXTRAS}$. An indefinite term is a basic example of complex terms that is produced by taking complement of an atomic term with respect to UNIVERSE. Each complex term is a subset of UNIVERSE. Therefore, there can be only $|\mathcal{P}(\texttt{UNIVERSE})| = 2^{2^t}$ different complex terms.

For all atomic terms $\texttt{X}_i$, $|\texttt{X}_i| = 2^{t-1}$. For all complex terms C, $0 \leq |\texttt{C}| \leq 2^t$. For example, $|\texttt{UNIVERSE}| = 2^t$, $|\texttt{VOID}| = 0$, $|\texttt{EXTRAS}| = 1$. Furthermore, $|\overline{\texttt{X}_i}| = 2^t - 1$,

$|\mathtt{X}_\mathtt{i} \cap \mathtt{X}_\mathtt{j}| = 2^{\mathtt{t}-2}$, $|\mathtt{X}_\mathtt{i} \cup \mathtt{X}_\mathtt{j}| = 2^\mathtt{t} - 2^{\mathtt{t}-2}$ where $\mathtt{X}_\mathtt{i}$ and $\mathtt{X}_\mathtt{j}$ are atomic terms and $\mathtt{X}_\mathtt{i} \neq \mathtt{X}_\mathtt{j}$.

4.2.3. Conversion Between Term Representations

It is trivial to convert an expression tree to a region set after calculating the atomic terms using algorithm 1. The calculation of complex terms is the standard expression tree evaluation.

Conversion from region sets to expression trees can be done via Quine-McCluskey algorithm (McCluskey (1956)) or Petrick's method (Petrick (1956)). They are similar to Karnaugh mapping but algorithmically more efficient.

4.3. Representation of Propositions

We have defined 5 types of propositions: $\mathtt{A}, \mathtt{E}, \mathtt{I}, \mathtt{O}$, and $\mathtt{S}$. These propositions are called *relational propositions*, or *relations*. In essence, those propositions express three types of *cardinality propositions*, or *cardinalities*: empty ($\nexists$), non-empty ($\exists$), or singleton ($\exists!$) sets.

Negations of these propositions other than those of type $\mathtt{S}$ ($\exists!$) exist within this logic: their negations can be used in need.

4.3.1. Propositions as Relational Propositions

Our initial definition of syllogistic propositions focuses on the relations between two terms named subject and predicate. These relational propositions are either a quantified proposition (i.e. one of $\{\mathtt{A}, \mathtt{E}, \mathtt{I}, \mathtt{O}\}$) or an individuality proposition (i.e. $\mathtt{S}$). Table 4.1 shows all possible types of relational propositions.

4.3.2. Propositions as Cardinality Propositions

Propositions can alternatively be represented using set-theoretic cardinality propositions. Table 4.2 shows all possible types of cardinality propositions.

Table 4.1. Relational propositions

Relation	English
$\mathtt{A}(\mathtt{C},\mathtt{D})$	All C are D
$\mathtt{E}(\mathtt{C},\mathtt{D})$	No C are D
$\mathtt{I}(\mathtt{C},\mathtt{D})$	Some C are D
$\mathtt{O}(\mathtt{C},\mathtt{D})$	Some C are not D
$\mathtt{S}(\mathtt{C})$	C is an individual

Table 4.2. Cardinality propositions

Cardinality	English
$\nexists(\mathtt{C})$	There are no C
$\exists(\mathtt{C})$	There are some C
$\exists!(\mathtt{C})$	There is only one C

Cardinality propositions capture the expressive power of relational propositions with fewer number of proposition types. This is an advantage for automated deduction. However, they may not be as intuitive as relational propositions (e.g. It is probably easier to think $\mathtt{A}(\mathtt{human},\mathtt{mortal})$, or "all humans are mortal" than of $\nexists(\mathtt{human} \sqcap \neg\mathtt{mortal})$, or "there is nothing which is human but not mortal").

4.3.3. Conversion Between Proposition Representations

Table 4.3 shows correspondence of these proposition types.

A cardinality may theoretically correspond to multiple relations. For example, $\nexists(\mathtt{C}) = \mathtt{E}(\mathtt{C},\mathtt{C}) = \mathtt{A}(\mathtt{C},\bot)$. More generally, $\nexists(\mathtt{C}) = \mathtt{E}(\mathtt{F},\mathtt{G}) = \mathtt{A}(\mathtt{F},\neg\mathtt{G})$ such that $\mathtt{C} = \mathtt{F} \sqcap \mathtt{G}$.

Table 4.3. Correspondence between proposition representations

Relation	Corresponding Cardinality	English
$\mathtt{A}(\mathtt{C},\mathtt{D})$	$\nexists(\mathtt{C} \sqcap \neg\mathtt{D})$	All C are D
$\mathtt{E}(\mathtt{C},\mathtt{D})$	$\nexists(\mathtt{C} \sqcap \mathtt{D})$	No C are D
$\mathtt{I}(\mathtt{C},\mathtt{D})$	$\exists(\mathtt{C} \sqcap \mathtt{D})$	Some C are D
$\mathtt{O}(\mathtt{C},\mathtt{D})$	$\exists(\mathtt{C} \sqcap \neg\mathtt{D})$	Some C are not D
$\mathtt{S}(\mathtt{C})$	$\exists!(\mathtt{C})$	C is an individual

Cardinality	Corresponding Relation	English
$\nexists(\mathtt{C})$	$\mathtt{E}(\mathtt{C},\mathtt{C})$	There are no C
$\exists(\mathtt{C})$	$\mathtt{I}(\mathtt{C},\mathtt{C})$	There are some C
$\exists!(\mathtt{C})$	$\mathtt{S}(\mathtt{C})$	There is only one C

4.4. Representation of Syllogisms

Neither order of premises nor duplicate premises can change the validity of a syllogism. As a result, premises of a syllogism can be represented as a set of propositions.

Naturally, queries, the conclusion candidates, are used for entailment checking. Alternatively, negation of a query can be added to the set of premises and the consistency is checked.

4.4.1. Syllogisms as Validity of Arguments

The natural representation of a syllogism is a logical argument:

Validity of $((assertion_1 \wedge assertion_2 \wedge \cdots \wedge assertion_p) \Rightarrow query)$ is the question. Figure 4.5 illustrates this representation.

4.4.2. Syllogisms as Inconsistency of Propositions

An alternative representation of a syllogism is a set of propositions:

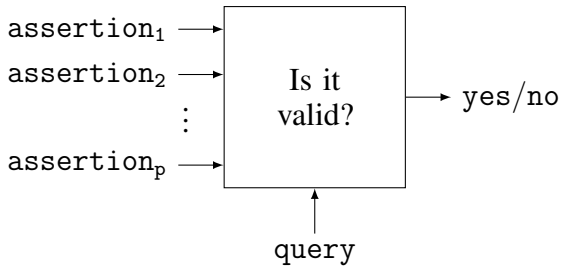


Figure 4.5. A syllogism as the validity of an argument

Inconsistency of $\{assertion_1, assertion_2, \ldots, assertion_p, \neg query\}$ is the question. Figure 4.6 illustrates this representation.

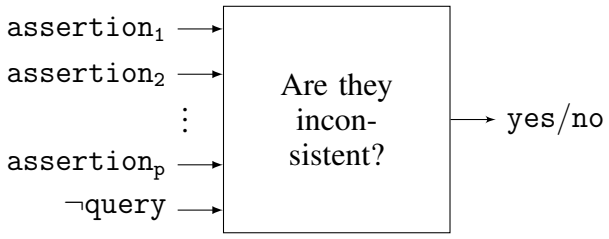


Figure 4.6. A syllogism as the inconsistency of propositions

Note that $\neg\exists(\mathtt{C}) = \nexists(\mathtt{C})$ and thus $\neg\nexists(\mathtt{C}) = \exists(\mathtt{C})$. Propositions of type $\exists!$ cannot be used as queries.

4.4.3. Conversion Between Syllogism Representations

Conversion between syllogism representations is trivial:

Proposition 4.4.1 *Let $a_1, ..., a_p$ and q be propositions. $((a_1 \wedge a_2 \wedge \cdots \wedge a_p) \Rightarrow q)$ is valid $\equiv \{a_1, a_2, \ldots, a_p, \neg q\}$ is inconsistent.*

CHAPTER 5

DEDUCTION FRAMEWORK

5.1. Introduction

In the previous chapter, two different forms of automated deduction were presented. Figure 5.1 summarizes these forms of deduction with examples. It was shown that the entailments checking and consistency detection are equivalent.

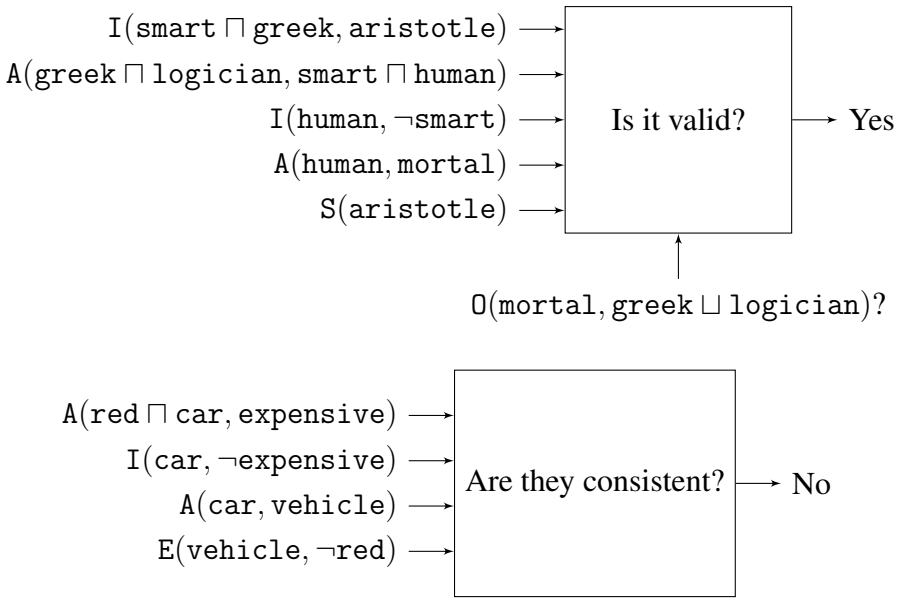


Figure 5.1. Automated deduction

In this chapter, we will define or adapt various methods for sound and complete automated deduction in generalized syllogisms.

5.2. Tractable Reasoning in Special Cases

NegSyl and PolSyl are tractable logics. Algorithm 2 is a polynomial time algorithm that is able to reason in NegSyl and thus in PolSyl.

Algorithm 2 Consistency detection in `NegSyl`

```
 1: function IsConsistent(kb)                          ▷ kb is knowledge base
 2:     // Checking consistency of current knowledge:
 3:     for all p ∈ kb.knowns do
 4:         if p.negation() ∈ kb.knowns then
 5:             kb.fillKnownsClearUnknowns()
 6:             return False
 7:     repeat
 8:         // Checking consistency of inferable knowledge:
 9:         for all p ∈ kb.unknowns do
10:             if immediatelyImplies(kb, p) then
11:                 kb.addToKnownsRemoveFromUnknowns(p)
12:                 if p.negation() ∈ kb.knowns then
13:                     kb.fillKnownsClearUnknowns()
14:                     return False
15:         // Checking domain non-emptiness:
16:         for all X ∈ kb.atomicTerms do
17:             if {∄(X), ∄(¬X)} ⊆ kb.knowns then
18:                 kb.fillKnownsClearUnknowns()
19:                 return False
20:     until convergence
21:     return True
```

Algorithm This algorithm consists of three stages: first, it searches for direct inconsistencies in the knowledge base. If both of a proposition and its negation appear in the knowledge base, then there is inconsistency. In the second stage, negations of immediately inferable propositions are searched. In the third stage, consistency with the non-empty domain assumption is checked. The second and the third stages are repeated until there is no change. The function $immediatelyImplies$ checks whether a given proposition is immediately inferable given a knowledge base. The function $implies$ checks the immediate inferences ignoring most of the propositions: it adds p to a copy of $kb.knowns$, then filters the propositions that contain irrelevant atomic terms, and then runs the exhaustive search in algorithm 5.

Soundness and Completeness Soundness of the algorithm relies on the soundness of the exhaustive search in algorithm 5. The most important reason of the low time complexity is the proposition elimination in the function $implies$: the algorithm ignores many combinations of propositions. However, the elimination of the propositions does not violate the completeness. Because polysyllogisms can be expressed as a chain of multiple syllogisms: when it is possible to construct a bridge between two terms, it is possible to construct that bridge using a single middle term.

Algorithm 3 Immediate inferences in NegSyl

```
1: function IMMEDIATELYIMPLIES(kb, p) ▷ kb is a knowledge base, p is a proposition
2:     XY = p.relevantAtomicTerms                                  ▷ 0 ≤ |XY| ≤ 2
3:     tautologies = {∄(⊥), ∃(⊤)}
4:     if p ∈ tautologies then
5:         return True
6:     for all Z ∈ kb.atomicTerms do
7:         XYZ = XY ∪ {Z}                                          ▷ 1 ≤ |XYZ| ≤ 3
8:         if implies(kb, p, XYZ) then
9:             return True
10:    return False
```

Tractability In the algorithm design, we take advantage of the monotonicity. The algorithm is guaranteed to converge in polynomial time: the number of all propositions is $\Theta(t^2)$ and in the worst-case, a single proposition will be inferred in each iteration. Numbers of iterations in the inner loops are also a polynomial. Algorithm 2 is a linear-time algorithm: there are $\Theta(t)$ iterations in each call and the innermost function call takes a constant time as the upper bound for the number of atomic terms is fixed to 3. These facts make the whole algorithm polynomial. Despite the fact that the complexity of the algorithm can be reduced, it is not an urgent need until practical applications of NegSyl is discovered.

5.3. Exhaustive Search for Visualization

A world is a subset of UNIVERSE. It contains the numbers of all the existing regions in UNIVERSE. Figure 5.2 shows a visualization of a possible world $\mathtt{w} = \{0, 1, 4, 6, 7\}$ in a universe with 3 atomic terms. Shaded regions corresponds to the empty regions in the interpretation. w satisfies $\mathtt{S}(\mathtt{X}_1 \sqcap \mathtt{X}_2)$ but does not satisfy $\mathtt{S}(\mathtt{X}_1)$. In w, $\mathtt{I}_{\{\upsilon\}}(\neg\mathtt{X}_1, \mathtt{X}_2 \sqcap \mathtt{X}_3)$ holds but $\mathtt{I}_{\{\upsilon\}}(\mathtt{X}_3 \sqcap \neg\mathtt{X}_2, \mathtt{X}_1)$ does not hold.

Algorithm 4 decides whether a proposition is true or not in a world. All the quantifiers are only related to existence or non-existence of the regions. When the proposition is a quantified proposition, if a region exists, it does not matter how many corresponding elements there are. However, single existing region does not imply single corresponding element in the interpretation. As a result, any model can be represented by a world from syllogistic point of view if and only if the conclusion is a quantified proposition. Therefore, the first part of the algorithm is unsound. However, this will not be a problem as the

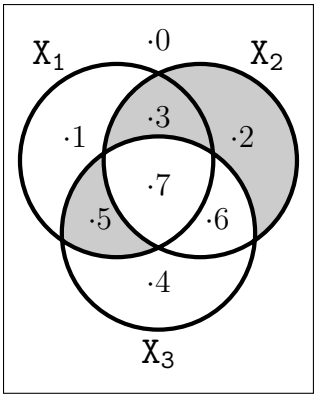


Figure 5.2. Visualization of a possible world

deductive machinery is designed accordingly. When the conclusion is a quantified proposition, the existing regions of the subject and the predicate are the intersections of those terms with the world. If there is an existential error, the proposition is false. Otherwise, the truth value is decided according to the quantifier of the proposition.

Algorithm 4 Does a given proposition hold in a given world?

```
 1: function HOLDS(p, w)                          ▷ p is a proposition, w is a world
 2:     if type(p) == S then                                  ▷ complete & unsound
 3:         return |w ∩ p.term| == 1
 4:     else                                                  ▷ complete & sound
 5:         subj ← w ∩ p.subject
 6:         pred ← w ∩ p.predicate
 7:         if p.subjectShouldExist and subj == ∅ then
 8:             return False
 9:         if p.predicateShouldExist and pred == ∅ then
10:             return False
11:         if p.quantifier == A then
12:             return subj \ pred == ∅
13:         if p.quantifier == E then
14:             return subj ∩ pred == ∅
15:         if p.quantifier == I then
16:             return subj == ∅ or subj ∩ pred ≠ ∅
17:         if p.quantifier == O then
18:             return subj == ∅ or subj \ pred ≠ ∅
```

Algorithm 5 decides whether a set of propositions is consistent. A set of propositions is consistent if and only if at least one possible world satisfies all the propositions.

Algorithm 6 decides whether a generalized syllogism is valid, i.e. conclusion necessarily follows from the premises. This algorithm takes a possibly empty set of premises and a conclusion. Each of the premises and the conclusion is a proposition.

Algorithm 5 Inconsistency detection & satisfiability checking

```
1: function IsConsistent(propositions)
2:     loop: for all world ⊆ UNIVERSE do
3:         for all proposition ∈ propositions do
4:             if not Holds(proposition, world) then
5:                 continue loop
6:         return True
7:     return False
```

Algorithm 6 Validity checking

```
1: function IsValid(premises, conclusion)
2:     if type(conclusion) == S then
3:         for all premise ∈ premises do
4:             if type(premise) ≠ S then
5:                 continue
6:             if Holds(A_∅(conclusion.term, premise.term), UNIVERSE) then
7:                 return IsValid(premises, I_{v}(conclusion.term, UNIVERSE))
8:         return not IsConsistent(premises)
9:     else
10:        loop: for all world ⊆ UNIVERSE do
11:            for all premise ∈ premises do
12:                if not Holds(premise, world) then
13:                    continue loop
14:            if not Holds(conclusion, world) then
15:                return False
16:        return True
```

When the query is of quantified proposition type, this procedure works for SLs as truth table works for propositional logic. When query is of individuality propositon type, this procedure searches for an upper bound 1 and then searches for a lower bound 1. If the search fails, as a last chance, it checks whether the assertions are inconsistent.

An advantage of these algorithms for satisfiability and validity checking is that they are very flexible in terms of responding to needs (i.e. it is possible to add more dimensions beyond our recommendation). For example, other types of constraints can easily be defined.

Another (and probably more important) advantage is that, with a little modification, this approach allows for visualizing valid and invalid syllogisms with compatible and contradicted worlds .

Soundness and Completeness The first part of the algorithm 6 searches an upper bound 1 for the number of corresponding elements in the interpretation of the regions of interest. If it finds an upper bound, it searches a lower bound 1, i.e. existence of the regions

of interest. Otherwise, the validity requires an inconsistency among the premises. The correctness of the second part of the algorithm 6 relies on the iteration of all possible worlds that works like a truth table for propositional logic. A syllogism is invalid if and only if there is at least one world that satisfies all the premises but does not satisfy the conclusion. In the corner case, if no world satisfies all the premises, any conclusion is valid because of the principle of explosion, i.e. from contradiction, anything follows.

Complexity The time complexity of checking validity or satisfiability for a single world is $\Theta(2^t \cdot p)$ where $\mathtt{t}$ is the number of atomic terms and $\mathtt{p}$ is the number of premises. This process is done for $2^{2^{\mathtt{t}}}$ worlds independently. Independence of many instances of the process allows us to run the program parallel and distributed. Nonetheless, the algorithms are seemingly not scalable in terms of the number of atomic terms as the number of possible worlds grows very fast. Despite theoretical time complexity, empirical results are promising:

1. invalid syllogisms are very dense in syllogisms that are generated uniformly at random,

2. a world contradicting with the conclusion in a consistent invalid syllogism is usually found at the very early stage of the search.

Implementation Let $\mathtt{t}$ be the number of atomic terms. Regions are numbered from 0 to $2^{\mathtt{t}} - 1$ inclusively. Any complex term is represented as sets of these region numbers. For efficiency, these sets are implemented as unsigned integers. A set $\{\mathtt{x}_1, \mathtt{x}_2, ..., \mathtt{x}_\mathtt{k}\}$ implemented as the result of the summation $\sum_{\mathtt{i}=1}^{\mathtt{k}} 2^{\mathtt{x}_\mathtt{i}}$ called a representative number. For example, $\{1, 3, 5, 7\}$ is implemented as 170, $\{0\}$ is implemented as 1, and $\{\}$ is implemented as 0. Using binary numeral system, it can be easily seen that this mapping is injective.

This approach has several consequences:

- Implementation usually becomes easier. For example, $\mathtt{for\ all\ world} \subseteq \mathtt{UNIVERSE}$ becomes $\mathtt{for}\ (\mathtt{world} = 0;\ \mathtt{world} <= \mathtt{UNIVERSE};\ ++\mathtt{world})$ and $\mathtt{VOID} = \emptyset$ becomes $\mathtt{VOID} = 0$.

- A region number $\mathtt{x} \in \{0, 1, ..., 2^{\mathtt{t}} - 1\}$ can be stored using $\mathtt{t}$ bits in a computer memory. Maximum number of regions in a complex term is $|\mathtt{UNIVERSE}| = 2^{\mathtt{t}}$.

Therefore, number of bits to store this term is roughly $\mathtt{t} \times 2^{\mathtt{t}}$. However, with the efficient implementation of region sets, the maximum number of bits to store a term is only $\lfloor \mathtt{log}_2 2^{2^{\mathtt{t}}-1} \rfloor + 1 = 2^{\mathtt{t}}$ as the universal term $\mathtt{UNIVERSE} = \{0, 1, ..., 2^{\mathtt{t}} - 1\}$ is implemented as $\sum_{\mathtt{x}=0}^{2^{\mathtt{t}}-1} 2^{\mathtt{x}} = 2^{2^{\mathtt{t}}} - 1$. Space complexity reduces in the worst case. The average case depends on probability distribution over the number of regions in a complex term.

- Efficient bitwise operations replace the set operations that require using a loop over the elements: `bitwise AND` replaces `intersection`, `bitwise OR` replaces `union`, and `bitwise NOT` replaces `complement`. For example, $\mathtt{X} \setminus \mathtt{Y}$ is implemented as $\mathtt{X} \,\&\, \sim \mathtt{Y}$. Nonetheless, when 2^t is greater than word size, multi-word arithmetic is required. Therefore, space complexity of a set and time complexity of a set operation are linear in terms of the number of regions (i.e. $\Theta(2^t)$ where t is the number of atomic terms, and therefore 2^t is the number of regions).

5.4. Syllogistic Constraint Satisfaction

A boolean constraint satisfaction problem (boolean CSP) can be defined by a tuple (V, C) such that

- V is the set of variables, each of which can take a boolean value, and
- C is the set of constraints, each of which consists of a pair: a scope of variables which appear in the constraint, and a relation which specify allowable values for those variables.

For instance:

$V = \{v_1, v_2, v_3\}$ and $C = \{c_1, c_2, c_3\}$ such that

$c_1 = (\ (v_1, v_2), \{(1, 1), (0, 0)\}\ ),$

$c_2 = (\ (v_1, v_2, v_3), \{(0, 0, 1), (0, 1, 0), (1, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 0), (1, 1, 1)\}\ ),$

$c_3 = (\ (v_1, v_3), \{(0, 0), (0, 1), (1, 1)\}\ )$

Another way of defining the same c_1, c_2 and c_3 is: $c_1 = ((v_1, v_2), v_1 = v_2)$, $c_2 = ((v_1, v_2, v_3), v_1 + v_2 + v_3 \geq 1)$ and $c_3 = ((v_1, v_3), v_1 \leq v_3)$. A solution to this boolean CSP is $\{v_1 = 0, v_2 = 0, v_3 = 1\}$.

Generalized syllogisms can be expressed as boolean CSPs: regions correspond to boolean variables (0 indicates the region is non-existing while 1 indicates the region is existing) and propositions correspond to constraints.

The most appropriate representations for this translation are terms as region sets, propositions as cardinality propositions, and syllogisms as inconsistency of propositions.

Another alternative type of CSP is the set CSPs. The only difference between set CSPs and boolean CSPs is that in set CSPs, each variable can take a set value instead of a boolean value.

5.4.1. MiniZinc

MiniZinc is a constraint modeling language (Nethercote et al. (2007)). Its contraint library allows for defining CSPs in a solver-independent way.

Human-readable MiniZinc model is compiled into FlatZinc, a more efficient language. FlatZinc provides an interface that is compatible with many solvers such as Choco, JaCoP and OR-Tools.

Region sets are transformed into bool arrays. For example, the complex term $\{1, 5, 7\}$ corresponds to the C in the following code:

```
var bool: r1;
var bool: r5;
var bool: r7;
array [1..3] of var bool : C = [r1, r5, r7];
```

Alternatively, region sets can be transformed into sets. Let $n = 2^t - 1$ where t is the number of atomic terms. The same complex term can be defined as

```
var set of {1, 5, 7}: C;
```

Syllogistic propositions are transformed into boolean constraints

- $\nexists$(C) corresponds to `constraint exactly(0, C, true)`;
- $\exists$(C) corresponds to `constraint at_least(1, C, true)`;
- $\exists!$(C) corresponds to `constraint exactly(1, C, true)`;

or set constraints

- $\nexists(\mathtt{C})$ corresponds to $\mathtt{constraint}\ \mathtt{card}(\mathtt{C}) = 0$;
- $\exists(\mathtt{C})$ corresponds to $\mathtt{constraint}\ \mathtt{card}(\mathtt{C}) > 0$;
- $\exists!(\mathtt{C})$ corresponds to $\mathtt{constraint}\ \mathtt{card}(\mathtt{C}) = 1$;

5.5. Theorem Proving in First-Order Logic

It is already stated that SLs are fragments of DLs which are fragments of FOL. Thus, SLs are fragments of FOLs. In this section, we define rules for direct transformation from SLs to FOL.

Let $\mathtt{X}$ be an atomic term and $\mathtt{C}$ and $\mathtt{D}$ complex terms. $\pi(\cdot)$ maps terms and propositions of SLs into FOL formulas such that

1. $\pi(\bot, \alpha) = \bot$,
2. $\pi(\top, \alpha) = \top$,
3. $\pi(\mathtt{X}, \alpha) = X(\alpha)$,
4. $\pi(\neg\mathtt{C}, \alpha) = \neg\pi(C, \alpha)$,
5. $\pi(\mathtt{C} \sqcap \mathtt{D}, \alpha) = \pi(C, \alpha) \wedge \pi(D, \alpha)$,
6. $\pi(\mathtt{C} \sqcup \mathtt{D}, \alpha) = \pi(C, \alpha) \vee \pi(D, \alpha)$,
7. $\pi(\exists(\mathtt{C})) = \exists\alpha.\pi(C, \alpha)$,
8. $\pi(\nexists(\mathtt{C})) = \forall\alpha.\pi(\neg C, \alpha)$,
9. $\pi(\exists!(\mathtt{C})) = \exists!\alpha.\pi(C, \alpha)$.

Here, the reason of introduction of the arbitrary variable α is that SLs are variable-free.

5.5.1. TPTP

TPTP is both a problem library and a special input format for theorem provers (Sutcliffe (2017)).

The Conference on Automated Deduction (CADE) Automated Theorem Prover (ATP) System Competition (CASC) is the annual world championship for sound, fully-automatic theorem proving (Sutcliffe (2016)). In the competition, problems are chosen from the TPTP problem library. Thus, all the participant theorem provers are able to process the problems in the TPTP format.

These participants include Vampire which is a very fast automated theorem prover for first-order logic. Vampire is the winner of CASC-26 organized in 2017.

Here, a more readible intermediate language called RuleML is used to represent syllogisms as theorems. Then, the program RuleML2TPTP is used to convert RuleML into TPTP.

It is trivial to convert syllogisms into RuleML format using the transformation rules for SLs into FOL. For example, $\texttt{A}(\texttt{X}, \texttt{Y})$ where `X` and `Y` are atomic terms, corresponds to the following code:

```
<Forall>
   <Var>a</Var>
   <Implies>
      <if>
         <Atom> <Rel>X</Rel><Var>a</Var> </Atom>
      </if>
      <then>
         <Atom> <Rel>Y</Rel><Var>a</Var> </Atom>
      </then>
   </Implies>
</Forall>
```

5.6. Entailment Checking in Description Logics

Syllogisms can be represented as DL ontologies. Terms of SLs and DLs have the same syntax. Thus, translation rules for `ComSyl` propositions into $\mathcal{ALC}$ axioms and for $\texttt{ComSyl}^+$ propositions into $\mathcal{ALCO}$ axioms is necessary and sufficient.

Let `C` and `D` be complex terms. Propositions of type `A` and `E` correspond to general concept inclusions:

$$\texttt{A}(\texttt{C}, \texttt{D}) \equiv (C \sqsubseteq D)$$

$$\texttt{E}(\texttt{C},\texttt{D}) \equiv (C \sqcap D \sqsubseteq \bot)$$

Let rnd_1 and rnd_2 be random individuals such that no further knowledge about them can be found in the ontology. Propositions of type `I` and `O` correspond to assertions on random individuals:

$$\texttt{I}(\texttt{C},\texttt{D}) \equiv F(rnd_1) \text{ such that } F \equiv C \sqcap D$$

$$\texttt{O}(\texttt{C},\texttt{D}) \equiv F(rnd_2) \text{ such that } F \equiv C \sqcap \neg D$$

$\texttt{ComSyl}^+$ additionally contains propositions of type `S` and $\mathcal{ALCO}$ additionally contains nominals.

Let rnd be a random individual such that no further knowledge about it can be found in the ontology. Propositions of type `S` corresponds to the following concept equivalence:

$$\texttt{S}(\texttt{C}) \equiv (C \equiv \{rnd\})$$

A concept equivalence can be defined using two general concept inclusions (e.g. $C \equiv \{rnd\}$ can be defined using $C \sqsubseteq \{rnd\}$ and $\{rnd\} \sqsubseteq C$).

5.6.1. OWL API

The W3C Web Ontology Language (OWL) is a semantic web language designed for representing complex knowledge bases. Direct semantics of OWL 2, the latest version of OWL, is based on $\mathcal{SROIQ}$.

OWL API is the standard implementation for creating, modifying and storing OWL ontologies. It also provides interface for DL reasoners such as HermiT, FaCT++, Pellet and Racer. This interface allows for consistency and entailment checking.

The correspondence between SLs and OWL API of DLs is as below:

- Terms correspond to *OWLClassExpression*s.
- Propositions of type `A` and `E` correspond to *OWLSubClassOfAxiom*s and *OWLDisjointClassesAxiom*s, respectively.
- Propositions of type `I` and `O` correspond to *OWLClassAssertionAxiom*s.
- Propositions of type `S` correspond to *OWLEquivalentClassesAxiom*s.

CHAPTER 6

CONCLUSION

In this work, we ...

... defined a family of SLs via syntactic restrictions on a generalized SL: $\texttt{PolSyl} \subset \texttt{NegSyl} \subset \texttt{ComSyl} \subset \texttt{ComSyl}^{+}$.

... specified comprehensive frameworks using these logics both for knowledge representation with alternative parts and for automated deduction with sound and complete methods. The software framework includes converter functions and competitive reasoners, and is fully compatible with the semantic web technologies.

... proved that PolSyl and NegSyl are tractable by defining a polynomial-time reasoning algorithm. In other words, categorical polysyllogistic reasoning with atomic negations is scalable.

... also proved that ComSyl and $\texttt{ComSyl}^{+}$ are categorical fragments of $\mathcal{ALC}$ and $\mathcal{ALCO}$, respectively, and therefore, they are in EXPTIME. These findings allow us to combine the best of both worlds: syntax of SLs and reasoners for DLs. Both intuitiveness of the natural syntax of SLs and time complexity of DLs in real world ontologies are already proven. This bridge is a chance to utilize the big data of natural language through an ontology learner as well as to develop easy-to-use end-user programs for manual knowledge management.

As future work, we will develop the software framework beyond the prototypical implementation and perform experiments. Furthermore, it is interesting to investigate if other features can be added to ComSyl or $\texttt{ComSyl}^{+}$ without sacrificing intuitiveness and algorithmic properties if possible (e.g. non-categorical roles or extended quantifiers).

REFERENCES

Alvarez, E. and M. Correia (2012). Syllogistic with indefinite terms. *History and Philosophy of Logic 33*(4), 297–306.

Alvarez-Fontecilla, E. (2016). Canonical syllogistic moods in traditional aristotelian logic. *Logica Universalis 10*(4), 517–531.

Baader, F., S. Brandt, and C. Lutz (2008). Pushing the el envelope further.

Baader, F., D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider (2010). *The Description Logic Handbook: Theory, Implementation and Applications* (2nd ed.). New York, NY, USA: Cambridge University Press.

Baader, F., I. Horrocks, C. Lutz, and U. Sattler (2017). *Introduction to Description Logic*. Cambridge University Press.

Brachman, R. J. and H. J. Levesque (1984). The tractability of subsumption in frame-based description languages. In *AAAI*, Volume 84, pp. 34–37.

Calvanese, D., G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati (2007). Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated reasoning 39*(3), 385–429.

Çine, E. (2018). Syllogistic knowledge bases with description logic reasoners. In *Computer Science and Engineering (UBMK), 2018 International Conference on*. IEEE.

Çine, E. and B. İ. Kumova (2017). An extended syllogistic logic for automated reasoning. In *Computer Science and Engineering (UBMK), 2017 International Conference on*, pp. 759–763. IEEE.

Donini, F. M. and F. Massacci (2000). Exptime tableaux for alc. *Artificial Intelligence 124*(1), 87–138.

Evans, G. (1977). Pronouns, quantifiers, and relative clauses (i). *Canadian journal of*

philosophy 7(3), 467–536.

Frege, G. (1879). *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. L. Nebert.

Grimm, S. and P. Hitzler (2009). A preferential tableaux calculus for circumscriptive alco. In *International Conference on Web Reasoning and Rule Systems*, pp. 40–54. Springer.

Hammer, E. M. (1998). Semantics for existential graphs. *Journal of Philosophical Logic 27*(5), 489–503.

Horrocks, I., O. Kutz, and U. Sattler (2006). The even more irresistible sroiq. *Kr 6*, 57–67.

Kazakov, Y. and B. Motik (2006). A resolution-based decision procedure for shoiq. In *International Joint Conference on Automated Reasoning*, pp. 662–677. Springer.

Krötzsch, M., F. Simancik, and I. Horrocks (2012). A description logic primer. *arXiv preprint arXiv:1201.4089*.

Lutz, C., F. Wolter, and M. Zakharyaschev (2008). Temporal description logics: A survey. In *Temporal Representation and Reasoning, 2008. TIME'08. 15th International Symposium on*, pp. 3–14. IEEE.

McCluskey, E. J. (1956). Minimization of boolean functions. *Bell Labs Technical Journal 35*(6), 1417–1444.

Moral, S. et al. (2017). Fuzzy description logics–a survey. In *Scalable Uncertainty Management: 11th International Conference, SUM 2017, Granada, Spain, October 4-6, 2017, Proceedings*, Volume 10564, pp. 31. Springer.

Moss, L. (2011a). Syllogistic logic with complements. In *Games, Norms and Reasons*, pp. 179–197. Springer.

Moss, L. S. (2009). Intersecting adjectives in syllogistic logic. In *MOL*.

Moss, L. S. (2010). Syllogistic logics with verbs. *J. Log. Comput. 20*, 947–967.

Moss, L. S. (2011b). Syllogistic logic with comparative adjectives. *Journal of Logic, Language and Information 20*, 397–417.

Murinová, P. and V. Novák (2016). Intermediate syllogisms in fuzzy natural logic. *Journal of Fuzzy Set Valued Analysis 2016*(2), 99–111.

Nethercote, N., P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack (2007). Minizinc: Towards a standard cp modelling language. In *CP*.

Nishihara, N., K. Morita, and S. Iwata (1990). An extended syllogistic system with verbs and proper nouns, and its completeness proof. *Systems and Computers in Japan 21*, 96–111.

Pereira-Fariña, M., J. C. Vidal, F. DÍaz-Hermida, and A. BugarÍn (2014). A fuzzy syllogistic reasoning schema for generalized quantifiers. *Fuzzy Sets and Systems 234*, 79–96.

Petrick, S. R. (1956). A direct determination of the irredundant forms of a boolean function from the set of prime implicants. *Air Force Cambridge Res. Center Tech. Report*, 56–110.

Pratt-Hartmann, I. and L. S. Moss (2009). Logics for the relational syllogistic. *Rew. Symb. Logic 2*, 647–683.

Quine, W. V. (1986). *Philosophy of logic* (2nd ed.). Harvard University Press.

Rudolph, S. (2011). Foundations of description logics. In *Reasoning Web. Semantic Technologies for the Web of Data*, pp. 76–136. Springer.

Russell, S. and P. Norvig (2009). *Artificial Intelligence: A Modern Approach* (3rd ed.). Upper Saddle River, NJ, USA: Prentice Hall Press.

Schwartz, D. G. (2014). Qualified syllogisms with fuzzy predicates. *International Journal*

of Intelligent Systems 29(10), 926–945.

Sutcliffe, G. (2016). The CADE ATP System Competition - CASC. *AI Magazine 37*(2), 99–101.

Sutcliffe, G. (2017). The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning 59*(4), 483–502.

van Rooij, R. (2010). Extending syllogistic reasoning. In *Logic, Language and Meaning*, pp. 124–132. Springer.

van Rooij, R. (2012). The propositional and relational syllogistic. *Logique et Analyse*, 85–108.

APPENDIX A

SUMMARY

A.1. Syntax of PolSyl

Let X and Y be atomic terms, and P a proposition. The production rule for propositions is defined as:

$$\mathtt{P} ::= \mathtt{A}(\mathtt{X},\mathtt{Y}) \mid \mathtt{E}(\mathtt{X},\mathtt{Y}) \mid \mathtt{I}(\mathtt{X},\mathtt{Y}) \mid \mathtt{O}(\mathtt{X},\mathtt{Y})$$

An example proposition is $\mathtt{E}(\mathtt{cat},\mathtt{human})$.

A.2. Syntax of NegSyl

Let X be an atomic term, P a proposition, and C and D possibly negated atomic terms. The production rules for possibly negated atomic terms and propositions are defined as:

$$\mathtt{C},\mathtt{D} ::= \mathtt{X} \mid \neg\mathtt{X}$$

$$\mathtt{P} ::= \mathtt{A}(\mathtt{C},\mathtt{D}) \mid \mathtt{E}(\mathtt{C},\mathtt{D}) \mid \mathtt{I}(\mathtt{C},\mathtt{D}) \mid \mathtt{O}(\mathtt{C},\mathtt{D})$$

An example proposition is $\mathtt{A}(\neg\mathtt{animal},\neg\mathtt{cat})$.

A.3. Syntax of ComSyl

Let X be an atomic term, P a proposition, and C and D complex terms. The production rules for complex terms and propositions are defined as:

$$\mathtt{C},\mathtt{D} ::= \mathtt{X} \mid \top \mid \bot \mid \neg\mathtt{C} \mid \mathtt{C} \sqcap \mathtt{D} \mid \mathtt{C} \sqcup \mathtt{D}$$

$$\mathtt{P} ::= \mathtt{A}(\mathtt{C},\mathtt{D}) \mid \mathtt{E}(\mathtt{C},\mathtt{D}) \mid \mathtt{I}(\mathtt{C},\mathtt{D}) \mid \mathtt{O}(\mathtt{C},\mathtt{D})$$

An example proposition is $\mathtt{I}(\neg\mathtt{small} \sqcap \top, \mathtt{huge})$.

A.4. Syntax of $\mathtt{ComSyl}^+$

Let $\mathtt{X}$ be an atomic term, $\mathtt{P}$ a proposition, and $\mathtt{C}$ and $\mathtt{D}$ complex terms. The production rules for complex terms and propositions are defined as:

$$\mathtt{C},\mathtt{D} ::= \mathtt{X} \mid \top \mid \bot \mid \neg\mathtt{C} \mid \mathtt{C} \sqcap \mathtt{D} \mid \mathtt{C} \sqcup \mathtt{D}$$

$$\mathtt{P} ::= \mathtt{A}(\mathtt{C},\mathtt{D}) \mid \mathtt{E}(\mathtt{C},\mathtt{D}) \mid \mathtt{I}(\mathtt{C},\mathtt{D}) \mid \mathtt{O}(\mathtt{C},\mathtt{D}) \mid \mathtt{S}(\mathtt{C})$$

An example of proposition is $\mathtt{S}(\mathtt{best} \sqcap \mathtt{author})$.