

UNIVERSITY OF CAPE TOWN

MASTERS THESIS

The Bayesian Description Logic *BALC*

Author:
Leonard BOTHA

Supervisor:
Prof. Thomas MEYER
Co-supervisor:
Asst. Prof. Rafael PEÑALOZA

*A thesis submitted in fulfillment of the requirements
for the degree of Master of Science*

in the

Faculty of Science

Centre for Artificial Intelligence Research
Department of Computer Science

June 17, 2018



CENTRE FOR ARTIFICIAL
INTELLIGENCE RESEARCH

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

Declaration of Authorship

I, Leonard BOTHA, declare that this thesis titled, “The Bayesian Description Logic *BALC*” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Signed by candidate

Date: 31-01-2018

“This is presented as a work of fiction and dedicated to nobody”

Post Office

University of Cape Town

Abstract

Faculty of Science
Department of Computer Science

Master of Science

The Bayesian Description Logic $\mathcal{BALC}$

by Leonard BOTHA

Description Logics (DLs) that support uncertainty are not as well studied as their crisp alternatives. This limits their application in many real world domains, which often require reasoning about uncertain or contradictory information. In this thesis we present the Bayesian Description Logic $\mathcal{BALC}$, which takes existing work on Bayesian Description Logics and applies it to the classical Description Logic $\mathcal{ALC}$.

We define five reasoning problems for $\mathcal{BALC}$; two versions of concept satisfiability (called total and partial respectively), knowledge base consistency, three subsumption problems (positive subsumption, p-subsumption, exact subsumption), instance checking, and the most likely context problem. Consistency, satisfiability, and instance checking have not previously been studied in the context of contextual Bayesian DLs and as such this is new work. We then go on to provide algorithms that solve all of these reasoning problems, with the exception of the most likely context problem.

We found that all reasoning problems in $\mathcal{BALC}$ are in the same complexity class as their classical variants, provided that the size of the Bayesian Network is included in the size of the knowledge base. That is, all reasoning problems mentioned above (excluding most likely context) are exponential in the size of the knowledge base and the size of the Bayesian Network.

Acknowledgements

First and foremost I would like to thank my supervisors, Prof. Thomas Meyer and Asst. Prof. Rafael Peñaloza, without whom this thesis would not exist. They not only provided guidance and a wealth of experience in the field, but went above and beyond in helping me access opportunities I would not otherwise have had access to.

Secondly, I would like to thank Mercy Lio for the kindness and friendliness she showed to someone completely out of their depth. Without her my time in Bolzano would have been far quieter.

Thirdly, I would like to thank ABSA Bank and Dr. Corrie Botha. Without ABSA's well timed decision to freeze my bank accounts while I was abroad, with only enough money in my wallet for a single pizza, I would never have learned how quickly one can transfer money from South Africa to Bolzano, Italy. Dr. Botha, on the other hand, has my thanks for his hasty trip to Western Union to ensure the pizza fund was temporarily replenished.

Finally, I would like to thank Annabel Botha and Santam Personal Insurance. Santam for the repairs to my vehicle and for replacing my glasses and Annabel for her patience and understanding with said matters.

Note that this thesis was generously funded by the Centre for Artificial Intelligence Research. This bursary, the CSIR-DST Inter-Programme Bursary, helped fund travel and face to face collaboration which was instrumental to the results obtained.

Contents

Declaration of Authorship	i
Abstract	iii
Acknowledgements	iv
1 Introduction	1
1.1 Description Logics	1
1.2 Bayesian Networks	3
1.3 Probabilistic Description Logics	3
2 Description Logics	5
2.1 The Description Logic $\mathcal{EL}$	6
2.2 The Description Logic $\mathcal{ALC}$	7
2.2.1 Syntax and Semantics	8
2.3 Classical Reasoning Services	10
2.3.1 Consistency Checking	11
2.3.2 Subsumption Checking	13
2.3.3 Satisfiability	14
2.3.4 Instance Checking	15
3 Bayesian Networks	17
3.1 Bayesian Networks	19
4 Probabilistic Description Logics	23
4.1 Overview of Existing Work	23
4.2 The Description Logic $\mathcal{BEL}$	25
4.2.1 Expressivity and Reasoning Complexity	27
Positive Subsumption	27
Certain Subsumption	28
Exact Subsumption	28
Most Likely Context	28
5 The Description Logic $\mathcal{BALC}$	29
5.1 Syntax and Semantics	30
5.1.1 Syntax	30
5.1.2 Semantics	31
5.2 Total Concept Satisfiability	35
5.2.1 Constructing the context ϕ_K^C	38
Preliminaries	39
Algorithm	44
Termination, Soundness, and Completeness	45
5.2.2 Constructing ϕ_B and $\bar{\phi}_B$	54
Constructing $\bar{\phi}_B$	55

Constructing ϕ_B	55
Complexity	55
5.2.3 Using ϕ_K^C and $\bar{\phi}_B$ to determine total concept satisfiability	56
5.3 Consistency	57
5.4 Subsumption	58
5.4.1 Positive Subsumption	61
5.4.2 Exact Subsumption	64
5.4.3 p -Subsumption	66
5.5 Partial Concept Satisfiability	67
5.6 Instance Checking	70
5.7 Most Likely Context	72
6 Conclusion	75
6.1 Summary of Contribution	75
6.1.1 Syntax and Semantics	75
6.1.2 Total concept satisfiability	75
6.1.3 Full knowledge base consistency	76
6.1.4 Subsumption of concepts	76
6.1.5 Partial (probabilistic) concept satisfiability	77
6.1.6 Instance Checking	77
6.2 Open Issues and Future Work	77
Bibliography	79

List of Figures

2.1	Expansion rules for $\mathcal{ALC}$ knowledge base consistency	12
3.1	Joint probability distribution over Boolean random variables $V = \{X, Y\}$	18
3.2	Example of a Bayesian Network	20
5.1	Expansion rules for constructing $\phi_{\mathcal{K}}^C$	45

*Dedicated to my most dearest intimate close lifelong¹ friend
Alexandra "Alexei" Lyndall McGregor*

¹since 2017

Chapter 1

Introduction

Currently Description Logics (or DLs) that support uncertainty are not as well studied as their crisp alternatives. The lack of mature probabilistic reasoning services in the field of Description Logics limits their application in many real world domains, which often require reasoning about uncertain or contradictory information. Existing work attempts to address these issues by enriching Description Logics with possibility, uncertainty, or vagueness. An overview of these concepts, applied to Description Logics, is provided in work by Lukasiewicz et al [20]. In our research we present a contextual probabilistic Description Logic and as such do not further explore possibility or vagueness and instead focus solely on probability.

Dealing specifically with probability, numerous approaches have been investigated for enriching Description Logics with uncertainty. The probabilistic syntax, and semantics, of these approaches vary from the level of what can be annotated with a probability to how probabilities are represented. For example, some probabilistic description logics support a probability range while others support only a fixed value. However, most probabilistic DLs do not support conditional probabilities. This forces numerous independence assumptions when modeling, which may be incorrect or too strong. These independence assumptions are often an attempt to reduce reasoning complexity, as assuming all random variables are independent makes probabilistic reasoning easier.

The focus of this work is on Bayesian Description Logics: DLs that have been enriched with a Bayesian Network, and as such have support for conditional probabilities as their core feature. We do however present other related probabilistic Description Logics in order to give a more complete overview of the field. Note that Bayesian Description Logics are not the only approach that allows for conditional probabilities, but that representing a probability distribution as a Bayesian Network has several advantages. Chief among these is the compact representation that Bayesian Networks offer (potentially exponentially better than a naive distribution) as well as providing a representation that is fairly easy to reason about intuitively. This helps to both store large distributions as well as making it easier to gather information from domain experts.

In the remaining sections of this chapter we provide a very brief introduction to Description Logics, Bayesian Networks, and Probabilistic Description Logics.

1.1 Description Logics

Description Logics (DL) form a family of well studied knowledge representation formalisms [23, 4]. They are characterized by a precise logic-based semantics and the use of *concepts*, *roles* and *individuals* to encode a domain.

In Description Logics concepts are an abstract or generic idea from a specific domain. They are often represented as unary predicates as in first-order logics; where

the intuition is that all individuals in a concept fall into some specific class. For example if a university is being modeled using a Description Logic concepts could include: *Student*, *Lecturer*, and *Course*.

Description Logic roles describe binary relationships between concepts, while individuals are specific instances of objects from the domain. Putting this all together an example from the university domain might be *teaches* or *attends* for roles, while individuals could be *Dave*, *Mary* or *John*.

Description Logics have concept constructors which allow the inductive definition of new concept descriptions based on the available constructors. Common constructors in DLs include: top concept ($\top$), bottom concept ($\perp$), conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$), value restriction ($\forall$), and existential restriction ($\exists$). Note that the top and bottom concept are special constructors that denote concepts on their own. Generally conjunction ($A \sqcap B$) is equivalent to intersection when concepts are interpreted as sets, while disjunction ($A \sqcup B$) is equivalent to union and negation ($\neg A$) to set complement. The concept description $\exists r.C$ describes the set of individuals that all are related to an individual in the concept C via the role r , while $\forall r.C$ describes the set of all individuals x where for all individuals y related to x via role r , y is an element of C . The top and bottom concept respectively are the set of the entire domain and the empty set.

For example if we would like to encode some information about the concept *CurrentStudent* from our university domain we might do so by using a General Concept Inclusion (GCI) as follows:

$$CurrentStudent \sqsubseteq Student \sqcap \exists attends.Lecture$$

This statement says that every individual in our domain that is a *CurrentStudent* must also be a *Student* and must also *attend* at least one *lecture*. Note that GCIs are not symmetric. That is not all *Students* that *attend* a *Lecture* are necessarily *CurrentStudents*.

Domain knowledge encoded using a Description Logic is usually represented in a knowledge base. A DL knowledge base generally consists of both a terminological component, called a TBox, and an assertional component, called an ABox [4, 23, 20]. The TBox contains terminological information on the concept level of the knowledge domain, encoded as a hierarchy of concepts that are partially ordered by a subsumption relation ($\sqsubseteq$), i.e. concept B is subsumed by concept A if the set of objects in concept B is completely contained in the set of objects in A. More simply, the TBox defines the different concepts in the domain and how they are related [23]. The *CurrentStudent* axiom shown previously is an example of an axiom that would appear in a TBox.

The ABox contains assertional information about individuals in the domain in the form of concept ($A(a)$), and role ($r(a, b)$), assertion axioms [20, 4]. That is, in the case of $A(a)$, that individual a is a member of concept A , and for $r(a, b)$, that individuals a and b are related via r . More concretely, the concept assertion *CurrentStudent*(*Dave*) is the assertion that the individual *Dave* is a *CurrentStudent*. While the role assertion *attends*(*Dave*, *MathLecture*) asserts that *Dave* attends a *MathLecture*. Both of these assertions could be found in the ABox of our university ontology.

This combination, of a TBox and ABox, allows Description Logics to represent a domain of knowledge in the form of logical sentences [23]. The expressivity of this KB varies according to the expressive power of the DL used. More expressive DLs can enforce more restrictions on the kind of sentences that can be constructed but

suffer increased reasoning complexity. Intuitively, a more expressive Description Logic can add more conditions (or restrictions) that state the requirements for an individual to be a member of a concept; however, this often results in reasoning being more complex. In Chapter 2 we present both the lightweight Description Logic $\mathcal{EL}$ as well as the more fully featured (but still decidable) Description Logic $\mathcal{ALC}$.

1.2 Bayesian Networks

There is a great deal of research that examines different techniques for representing uncertainty. Depending on the technique used what exactly can be modeled varies. As such for any problem the probability model that best suits it needs to be carefully considered. In this work we chose to use Bayesian Networks as the underlying probability model because of their ability to compactly represent conditional probability distributions and their previous use in related work.

Bayesian Networks [24] are a member of the family of probabilistic graphical models. They are a way of encoding probability distributions using directed acyclic graphs (DAG). This encoding allows Bayesian Networks to use the conditional independences between random variables to compactly represent a distribution. This has the advantage of greatly reducing the amount of space required to represent a distribution. In extreme cases Bayesian Networks can represent in polynomial space what would require a naive distribution an exponential amount of space to represent [11]. This makes them an attractive way to encode conditional probability distributions.

We will provide an overview of probability as well as a thorough explanation of Bayesian Networks in Chapter 3. However, we will briefly explain Bayesian Networks here as well. In a Bayesian Network each node in the DAG represents a random variable, and each edge is a dependency between the connected variables. Intuitively this means that if two nodes are connected in the DAG then the probability of one node is dependent on the value of the other node. It is further required that a conditional probability distribution be given for each random variable given its parents in the DAG. This conditional distribution is usually a table containing a distribution for each combination of parents of the random variable. For example this table may tell us that a node has probability 0.6 of being true if another node is true or probability 0.25 of being true if the other node is false.

Using a Bayesian Network the probability of any world (the probability of all the random variables taking on specific values) can be calculated in linear time. This is done by looking up the probability of each random variable taking on its assigned value (given the assigned values of its parents) and then multiplying all these values together. However inference is NP-hard in the general case (where we do not have a value for all random variables), but is polynomial in certain special cases (depending on the DAG not the valuation). We will not go into further detail now but provide formal definitions and complexity results in Chapter 3.

1.3 Probabilistic Description Logics

Probabilistic Description Logics (PDLs) are an attempt to enrich classical Description Logics, which can only encode certain knowledge, with the ability to make statements which may not always hold (or have some probability of holding). This allows the representation of uncertain knowledge, or knowledge that only holds some of the time. This field of study can be easily motivated by considering the canonical

example of the penguin (this example is often used to illustrate the limitations of classical Description Logics).

If we have an ontology that stores information about animals we might have the following axiom in our TBox $Birds \sqsubseteq Flying \sqcap Animal$. This just states that birds are flying animals, which seems like a sensible statement. We now want to model penguins and add the following statement $Penguin \sqsubseteq Bird \sqcap \neg Flying$. This statement claims that penguins are flightless birds, which again seems sensible. However we are now left with the conclusion that there can exist no penguins! Intuitively we can see why this is true by noticing that since penguins are birds they must fly, but since penguins are flightless they also must not fly. Since this is a contradiction the only way our knowledge base can be consistent is to have no penguins.

Obviously since penguins do actually exist this presents a severe limitation to the application of Description Logics to real world problems. Fortunately Probabilistic Description Logics give us a way to address this problem. Using a PDL we could instead say that only *most* birds are flying. This suddenly means that we can have penguins again since we have now removed the previous contradiction.

Exactly how we make the statement that only *most* birds fly varies between PDLs. That is, the semantics of the actual probabilities, what the probabilities associated with statements mean, varies depending on the Description Logic. In some Probabilistic Description Logics probabilities the probabilistic information that is encoded is the likelihood for an individual to be a member of a concept [27, 17]. For example how likely it is for a bird to fly. Other Probabilistic Description Logics instead take a contextual approach [4] and associate with statements a context which has a probability to hold. For example under what conditions do birds fly and how likely are these conditions.

Furthermore, Probabilistic Description Logics vary greatly in the type of probabilistic information that can be encoded. For example some PDLs can only encode probabilistic terminological information (statements like only some birds fly) while others can only handle probabilistic assertional information (assertions that only a specific bird flies). Finally, PDLs also differ in the underlying Description Logic that is used. Depending on this logic the different PDLs have varying degrees of expressive power.

The PDL we present in this work, Bayesian $\mathcal{ALC}$ ($\mathcal{BALC}$), is uncommon as it is capable of representing both classical and probabilistic information that can be either terminological or assertional in nature. Most other PDLs cannot do this, and in fact all PDLs similar to $\mathcal{BALC}$ in some way (that use $\mathcal{ALC}$ as the underlying DL or similar probabilistic semantics) that we examined at the time of writing were unable to do this. We should mention that $\mathcal{BALC}$ follows the probabilistic semantics approach of encoding certain information in uncertain contexts (conditions). So in $\mathcal{BALC}$ we would encode that birds that are not penguins fly instead of saying most birds fly.

We will go into more depth on Probabilistic Description Logics in Chapter 4. In particular we will examine the Probabilistic Description Logic $\mathcal{BEL}$ [4], on which we based $\mathcal{BALC}$, in greater detail.

Chapter 2

Description Logics

Description Logics (DL) are a well studied family of knowledge representation formalisms. They are typically used to describe some abstraction of a domain of interest [2]. That is, Description Logics provide a formal means to model some domain such that there is no ambiguity in the modeling language. Note that this does not mean that there is no ambiguity in the domain itself, just that there is no ambiguity in what the Description Logic statements mean.

Description Logics make the assumption that the domain of interest is populated by elements, or individuals. The elements in a domain can be described using *concepts* and *roles*. Intuitively concepts are categories that an individual belong to; for example an individual could be a member of the concept *cat*. Informally concept membership can be understood as an *is a* relationship. As such the statement that an individual is a member of the concept *cat* intuitively just means that the individual is a cat. On the other hand roles are binary relationships between individuals. Intuitively if two individuals are connected via a role then they are related by that role name. For example if a *cat* individual is related to a *person* individual via the *owner* role then intuitively the *cat* is *owned* by the *person*.

Another important distinction in Description Logics is the difference between assertional and terminological knowledge. We call statements that describe the relationship between concepts terminological while statements that describe individuals are assertional. This distinction is useful as it allows us to differentiate between statements that describe the domain and statements about individuals in the domain. More concretely the previous examples we gave about cats and their owners are both examples of assertional knowledge. This is because both of the previous statements give us information about a specific individual(s); that is a specific *cat* and its relationship to a *person*. An example of terminological information would be something like the statement that all *cats* are *mammals*; which affects the entire class of individuals that are *cats*.

We have now provided an intuition of how information is represented in Description Logics (concepts and roles); as well as the different types of information present in a knowledge base (terminological and assertional). We can now start providing some more formal definitions, starting with a more rigorous definition of both concepts and roles.

- **Concepts:** concepts are interpreted as sets of elements, from the domain, and can be viewed as unary predicates. In Description Logics concepts are constructed from *concept names* and *role names* using the constructors specific to the Description Logic. We call the set of elements a concept represents its extension.
- **Roles:** roles are interpreted as binary relationships between elements and can be viewed as binary predicates. Some Description Logics have constructors

that allow the construction of new roles, or the specification of relationships between roles. However both $\mathcal{EL}$ and $\mathcal{ALC}$ do not have such constructors and as such we will not cover this in more detail.

Each DL has what is called a *concept language* which defines which constructors the DL supports. The *concept language* is a formal language that allows us to build *concept descriptions* or *compound concepts* (and possibly *role descriptions*) from concept names, role names, and other primitives. In this work we are primarily interested in the Description Logic $\mathcal{ALC}$ as this is the underlying DL for $\mathcal{BALC}$. However since much of the background work is based on $\mathcal{BEL}$ we also provide a brief overview of $\mathcal{EL}$ which underlies $\mathcal{BEL}$.

For the rest of this chapter we start by providing the syntax as well as some of the semantics of $\mathcal{EL}$ in Section 2.1. Note that we do not define constructs that are shared with $\mathcal{ALC}$ in this section to avoid repetition. We do however mention which $\mathcal{ALC}$ constructs are shared with $\mathcal{EL}$ in Section 2.2. Also in Section 2.2 we provide the formal syntax and semantics of $\mathcal{ALC}$. Once we have done this we present some classical reasoning problems for Description Logics in Section 2.3. We explain how these reasoning problems are relevant (or not relevant) to both $\mathcal{EL}$ and $\mathcal{ALC}$.

2.1 The Description Logic $\mathcal{EL}$

In our initial introduction to Description Logics we mentioned that as a rule more expressive Description Logics have poor reasoning performance. In fact reasoning in many DLs is not tractable, and in some cases is even undecidable. This has lead to the study of more lightweight Description Logics, of which $\mathcal{EL}$ is one.

As far as expressivity (available constructors) goes $\mathcal{EL}$ is very limited. It provides only the top concept ($\top$), conjunction ($\sqcap$), and existential restriction ($\exists r.C$) as concept constructors. In exchange $\mathcal{EL}$ provides polynomial time inference which is exceptionally good in the field of Description Logics.

While the concept constructors may seem very limiting they are still expressive enough for many applications. For example the notion of a *CurrentStudent* we introduced earlier can still be expressed:

$$\text{CurrentStudent} \sqsubseteq \text{Student} \sqcap \exists \text{attends.Lecture}$$

Furthermore there are practical applications that are backed by $\mathcal{EL}$ terminologies, of which the SNOMED medical terminology is a good example [26]. Having provided some motivation for the use of $\mathcal{EL}$ we now move onto formal definitions.

In this section we will present only the basic syntax and semantics of $\mathcal{EL}$ [1] and will save the definition of other constructs (ABoxes, TBoxes, and Knowledge Bases) for Section 2.2 to avoid unnecessary repetition.

Definition 2.1.1 (Syntax). Let $\mathbf{C}$ be a set of concept names and $\mathbf{R}$ be a set of role names disjoint from $\mathbf{C}$. The set of $\mathcal{EL}$ concept descriptions over $\mathbf{C}$ and $\mathbf{R}$ is inductively defined as follows:

- every concept name is an $\mathcal{EL}$ concept description
- $\top$ is an $\mathcal{EL}$ concept description
- if C and D are $\mathcal{EL}$ concept descriptions and r is a role name, then the following are also $\mathcal{EL}$ concept descriptions:

- $C \sqcap D$ (conjunction)
- $\exists r.C$ (existential restriction)

As a Description Logic $\mathcal{EL}$ follows a model-based semantics based on interpretations. Intuitively, an interpretation can be thought of as a mapping between concrete elements in some domain and the concepts and roles of a particular knowledge base. Informally we will say that a specific interpretation models an $\mathcal{EL}$ ontology if it satisfies the statements in the ontology (we will formally define models in Section 2.2). More formally we define an $\mathcal{EL}$ interpretation as follows:

Definition 2.1.2 (Semantics). An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, called the interpretation domain, and a mapping $\cdot^{\mathcal{I}}$ that maps every:

- concept name $A \in \mathbf{C}$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- role name $r \in \mathbf{R}$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

The mapping $\cdot^{\mathcal{I}}$ is extended to compound concepts as follows:

- $\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \exists c \in \Delta^{\mathcal{I}} \text{ with } \langle d, c \rangle \in r^{\mathcal{I}} \text{ and } c \in C^{\mathcal{I}}\}$

Having now defined interpretations we would normally proceed with defining the components of a knowledge base. However, since we save this for the section on $\mathcal{ALC}$ we can go no further without repetition. Therefore we now move directly onto $\mathcal{ALC}$.

2.2 The Description Logic $\mathcal{ALC}$

Generally moving into an intractable complexity class is a very poor result in computing. However it is often the case for Description Logics that it is necessary in order to have the expressivity needed to model some domain. This can be quite easily seen if we examine which constructors are missing from $\mathcal{EL}$. A keen reader (or reader familiar with logics) would have noticed that a constructor as simple as negation is not present in $\mathcal{EL}$. This means that you cannot express that some individual is not a member of some concept. As a consequence this means that if we state that some individual is wet we do not know that they are not also simultaneously dry. Clearly there is necessity for a logic that has more expressivity even if its worst case complexity is less than ideal.

The Description Logic $\mathcal{ALC}$ attempts to address the issue of expressivity by including some of the more obvious constructors while remaining decidable. $\mathcal{ALC}$ was first presented by Schmidt-Schauß et al. [25] and, as has been mentioned, is fairly expressive. It offers various constructors namely: negation($\neg$), bottom concept($\perp$), top concept($\top$), existential quantifier($\exists$), universal quantifier($\forall$), disjunction($\sqcup$), and conjunction($\sqcap$) constructors. These constructors give the nice result that $\mathcal{ALC}$ is propositionally complete (it supports all Boolean set operations on concepts). However this expressivity is at the expense of reasoning complexity, since even the simplest reasoning services of $\mathcal{ALC}$ are not tractable.

Having provided an introduction to the logic we now give the syntax and semantics and then define what is meant by an $\mathcal{ALC}$ ABox, TBox, and knowledge base.

2.2.1 Syntax and Semantics

We present here the relevant definitions from the textbook by Baader et al. [2] that will be used in the rest of this thesis. Note that the syntax of $\mathcal{EL}$ is compatible with $\mathcal{ALC}$ in the sense that all valid $\mathcal{EL}$ statements are $\mathcal{ALC}$ statements.

Definition 2.2.1 (Syntax). Let $\mathbf{C}$ be a set of concept names and $\mathbf{R}$ be a set of role names disjoint from $\mathbf{C}$. The set of $\mathcal{ALC}$ concept descriptions over $\mathbf{C}$ and $\mathbf{R}$ is inductively defined as follows:

- every concept name is an $\mathcal{ALC}$ concept description
- $\top$ and $\perp$ are $\mathcal{ALC}$ concept descriptions
- if C and D are $\mathcal{ALC}$ concept descriptions and r is a role name, then the following are also $\mathcal{ALC}$ concept descriptions:
 - $C \sqcap D$ (conjunction)
 - $C \sqcup D$ (disjunction)
 - $\neg C$ (negation)
 - $\exists r.C$ (existential restriction)
 - $\forall r.C$ (value restriction)

By defining the syntax of $\mathcal{ALC}$ we are able to distinguish between well formed expressions and those that are not. Importantly we now have a more expressive language and can express statements like “things that are wet are not dry” and “all cats are either wild or tame”. Note that neither of these statements can be expressed in $\mathcal{EL}$. We will provide the $\mathcal{ALC}$ statements that correspond to these statements once we have defined the semantics of the logic. Like $\mathcal{EL}$ the semantics of $\mathcal{ALC}$ is based on interpretations which are formalized as follows.

Definition 2.2.2 (Semantics). An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, called the interpretation domain, and a mapping $\cdot^{\mathcal{I}}$ that maps every:

- concept name $A \in \mathbf{C}$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- role name $r \in \mathbf{R}$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

The mapping $\cdot^{\mathcal{I}}$ is extended to compound concepts as follows:

- $\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$
- $\perp^{\mathcal{I}} := \emptyset$
- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \exists c \in \Delta^{\mathcal{I}} \text{ with } \langle d, c \rangle \in r^{\mathcal{I}} \text{ and } c \in C^{\mathcal{I}}\}$
- $(\forall r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \forall c \in \Delta^{\mathcal{I}}, \text{ if } \langle d, c \rangle \in r^{\mathcal{I}}, \text{ then } c \in C^{\mathcal{I}}\}$

Now that we have a formal definition of what $\mathcal{ALC}$ statements mean we can convert the previous natural language sentences into $\mathcal{ALC}$ statements. We could express the notion that “things that are wet are not dry” by asserting that the concepts are disjoint:

$$Wet \sqcap Dry \sqsubseteq \perp$$

This statement essentially claims that there are no individuals that are both Wet and Dry. We could express the statement that “all cats are either wild or tame” as:

$$Cat \sqsubseteq Wild \sqcup Tame$$

which directly states that all individuals that are a member of Cat must be either a member of the concept $Wild$ or a member of the concept $Tame$. Note that our examples have only used some of new constructors and were chosen to be more illustrative than accurate.

We have now defined the concept language $\mathcal{ALC}$ and some of its semantics. However we have still not provided a link between DL statements and interpretations. Furthermore we have no way to efficiently group several terminological or assertional statements together to form a knowledge base. Fortunately we can address both these issues simultaneously by defining TBoxes, ABoxes, and then combining these to form a knowledge base.

Definition 2.2.3 (TBox). For C and D possibly compound $\mathcal{ALC}$ concepts, an expression of the form $C \sqsubseteq D$ is called an $\mathcal{ALC}$ general concept inclusion and abbreviated GCI. We use $C \equiv D$ as an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. A finite set of GCIs is called an $\mathcal{ALC}$ TBox. An interpretation $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation that satisfies each GCI in a TBox $\mathcal{T}$ is called a model of $\mathcal{T}$.

Note that the definition of a TBox can be modified for $\mathcal{EL}$ by simply requiring that the concepts C and D be $\mathcal{EL}$ concepts instead of $\mathcal{ALC}$ concepts.

The definition of a TBox allows us to group a collection of GCIs together into a single construct. This allows the reasoning about interpretations that model multiple statements at once, which is essential in order to encode any non-trivial domain. For example any interpretations that model a TBox that contains both our statements about cats and wetness would respect them both. Suddenly if we are told that we see a *dry cat* we know that that it is not *wet*. This kind of inference is very useful as it enables modelers to build up stores of terminological information which in turn allows the modeling more complex domains.

Now that we are able to capture terminological domain information we still need a way to capture knowledge about individuals in our domain. We do this by defining a similar construct to a TBox for assertional knowledge. Note that we can again modify this definition for $\mathcal{EL}$ by replacing $\mathcal{ALC}$ concepts with $\mathcal{EL}$ concepts in the definition.

Definition 2.2.4 (ABox). Let $\mathbf{I}$ be a set of individual names disjoint from $\mathbf{R}$ and $\mathbf{C}$. For $a, b \in \mathbf{I}$, C a possibly compound $\mathcal{ALC}$ concept, and $r \in \mathbf{R}$ a role name, an expression of the form

- $C(a)$ is called an $\mathcal{ALC}$ concept assertion, and
- $r(a, b)$ is called an $\mathcal{ALC}$ role assertion.

A finite set of $\mathcal{ALC}$ concept and role assertions is called an $\mathcal{ALC}$ ABox. An interpretation function $\cdot^{\mathcal{I}}$ is additionally required to map every individual name $a \in \mathbf{I}$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ satisfies

- a concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and
- a role assertion $r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$.

An interpretation that satisfies each concept assertion and each role assertion in an ABox $\mathcal{A}$ is called a model of $\mathcal{A}$.

By formalizing the ABox we are now able to represent information about specific individuals in a domain. For example if we know that a individual Tom is black we could add the assertion $hasColour(Tom, Black)$ to our ABox. This is useful as we can now make statements that affect only *Tom* and not all *cats*, e.g. we can say that *Tom* is *black* without claiming that all *cats* are *black*.

Clearly it would be useful when modeling a domain to be able to use both terminological and assertional statements. The most obvious way to do this would be to define some construct that combines both an ABox and a TBox. Which is exactly what is typically done when defining a knowledge base.

Definition 2.2.5 (Knowledge Base). An $\mathcal{ALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of an $\mathcal{ALC}$ TBox $\mathcal{T}$ and an $\mathcal{ALC}$ ABox $\mathcal{A}$. An interpretation that is both a model of $\mathcal{A}$ and of $\mathcal{T}$ is called a model of $\mathcal{K}$.

This definition can be modified to work for $\mathcal{EL}$ by simply using an $\mathcal{EL}$ TBox and an $\mathcal{EL}$ ABox instead of the $\mathcal{ALC}$ equivalents. Note that both $\mathcal{ALC}$ and $\mathcal{EL}$ knowledge bases need not contain both terminological and assertional knowledge. It is valid to have a KB with either an empty ABox or TBox. This would be denoted as either $(\mathcal{T}, \emptyset)$ or $(\emptyset, \mathcal{A})$.

We have now introduced the syntax used to encode information in both $\mathcal{ALC}$ and $\mathcal{EL}$. Furthermore we have also provided the semantics for knowledge bases in both Description Logics. However simply writing down what we know about a domain in a structured form is of limited use. After all the goal of representing information in a structured form is often to use this information to come to new conclusions. In Description Logics this process is called inference (or reasoning). In the next section we present some of the reasoning problems that exist for Description Logics and provide specific details for both $\mathcal{ALC}$ and $\mathcal{EL}$.

2.3 Classical Reasoning Services

In this section we present reasoning services that are typically present in classical Description Logics. In particular we present reasoning services that are relevant for $\mathcal{ALC}$. We also provide a complexity class for each problem in $\mathcal{ALC}$ as well as $\mathcal{EL}$. Furthermore we also present an exponential time algorithm for knowledge base consistency for $\mathcal{ALC}$ in some detail. Note that since most reasoning problems in $\mathcal{ALC}$ are reducible to each other, in polynomial time, they all tend to fall into the same complexity class. In fact all reasoning problems presented in this section are reducible to each other, and in particular to satisfiability, which is convenient since Schmidt-Schauß et al. [25] have shown that satisfiability is PSPACE-hard for $\mathcal{ALC}$ (for acyclic TBoxes). Therefore it follows that all problems in this section are PSPACE-hard for $\mathcal{ALC}$ (with acyclic TBoxes). However, in the general case these problems are all in EXPTIME since the general case permits cyclic TBoxes (which ruins the PSPACE complexity result). In contrast most reasoning problems in $\mathcal{EL}$ are tractable and therefore are in P (have polynomial time complexity). In what follows we will start by presenting the consistency problem, and will then move onto subsumption, satisfiability, and finally instance checking.

2.3.1 Consistency Checking

For an interpretation, in both $\mathcal{EL}$ and $\mathcal{ALC}$, we have said that it models a knowledge base if, and only if, it satisfies all statements in the knowledge base. Intuitively an interpretation is an instance of a domain that the knowledge base represents if it models the knowledge base. While this all seems perfectly reasonable we have not yet mentioned how we determine if a knowledge base has a model. Obviously this is important as a knowledge base with no models is probably a modeling error. This brings us to the problem of consistency checking for a knowledge base.

We say that a knowledge base is consistent if, and only if, there exists a model for it. The consistency checking problem attempts to find such a model, or show that no such model can exist. Consistency is formally defined in $\mathcal{ALC}$ as:

Definition 2.3.1 (Consistency). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ALC}$ knowledge base. We say that $\mathcal{K}$ is *consistent* if it has a model.

It is interesting (and important) to mention here that this reasoning problem has no equivalent in $\mathcal{EL}$ as all $\mathcal{EL}$ knowledge bases are trivially consistent. This is a result of $\mathcal{EL}$ not having constructors that can lead to inconsistencies in the knowledge base.

In $\mathcal{ALC}$ consistency reasoning services often makes use of a tableaux algorithm in order to construct a model of a given KB. Should this algorithm succeed then a model has been found for the KB meaning that it is consistent. It has also been shown, for the standard tableaux algorithm, that should this algorithm fail then no model can exist for the given KB and therefore it is inconsistent.

The $\mathcal{ALC}$ tableaux algorithm works by applying a set of expansion rules to a knowledge base in an attempt to construct a model for the knowledge base. The application of these rules forms a tree where each branch in the tree is the result of a disjunction (intuitively an or) in an assertion. The algorithm then picks one of these branches and expands it until no rules are applicable, in which case a model has been found, or it finds an inconsistency, in which case the algorithm backtracks to the last branch. If the algorithm finds a fully expanded ABox (an ABox to which no further rules can be applied) then it terminates and returns that the KB is consistent. Alternatively if it has followed all branches in the execution tree and found a clash (a pair of contradictory assertions) in each ABox then it terminates and returns inconsistent. This tableaux algorithm has been shown to be both sound and complete, and as such is a valid procedure for consistency.

Note that in the case where the ABox is empty the algorithm requires that an assertion be inserted into the ABox asserting that there is an individual in the interpretation. This individual does not need to be a member of any concept defined in the TBox and is usually just asserted to be a member of the top concept. We provide the standard $\mathcal{ALC}$ tableaux rules, in Figure 2.1, from the textbook by Baader et al. [1] for $\mathcal{ALC}$ knowledge base consistency checking. Note that we have modified the rules to be consistent with the terminology we will use later on for $\mathcal{BALC}$, but have not changed their semantics.

While most of these rules are simple the $\exists$ -rule is slightly more intricate. The $\exists$ -rule uses a concept called blocking which we will explain in greater detail. On a high level blocking exists to ensure that the tableaux algorithm does not enter a loop where new individuals are inserted indefinitely. This problem can easily be seen if we consider the following knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

$$\mathcal{T} = \{\top \sqsubseteq \exists r.C\}$$

$\sqcap$ -rule	if 1. $(C \sqcap D)(a) \in \mathcal{A}$, and 2. $\{C(a), D(a)\} \not\subseteq \mathcal{A}$ then $\mathcal{A} \rightarrow \mathcal{A} \cup \{C(a), D(a)\}$
$\sqcup$ -rule	if 1. $(C \sqcup D)(a) \in \mathcal{A}$, and 2. $\{C(a), D(a)\} \cap \mathcal{A} = \emptyset$ then $\mathcal{A} \rightarrow \mathcal{A} \cup \{X(a)\}$ for some $X \in \{C, D\}$
$\exists$ -rule	if 1. $(\exists r.C)(a) \in \mathcal{A}$, and 2. there is no b such that $\{r(a, b), C(b)\} \subseteq \mathcal{A}$, and 3. a is not blocked. then $\mathcal{A} \rightarrow \mathcal{A} \cup \{r(a, d), C(d)\}$, where d is new in $\mathcal{A}$.
$\forall$ -rule	if 1. $\{(\forall r.C)(a), r(a, b)\}$, and 2. $C(b) \notin \mathcal{A}$ then $\mathcal{A} \rightarrow \mathcal{A} \cup \{C(b)\}$
$\sqsubseteq$ -rule	if 1. $C \sqsubseteq D \in \mathcal{T}, E(a) \in \mathcal{A}$, and 2. $(\neg C \sqcup D)(a) \notin \mathcal{A}$ then $\mathcal{A} \rightarrow \mathcal{A} \cup \{(\neg C \sqcup D)(a)\}$
$\equiv$ -rule	if 1. $C \equiv D \in \mathcal{T}, E(a) \in \mathcal{A}$, and 2. $((\neg C \sqcup D) \sqcap (\neg D \sqcup C))(a) \notin \mathcal{A}$ then $\mathcal{A} \rightarrow \mathcal{A} \cup \{((\neg C \sqcup D) \sqcap (\neg D \sqcup C))(a)\}$

FIGURE 2.1: Expansion rules for $\mathcal{ALC}$ knowledge base consistency

$$\mathcal{A} = \{\top(x)\}$$

This knowledge base has the following trivial model $\mathcal{I}$:

$$\begin{aligned}\Delta^{\mathcal{I}} &= \{x\} \\ C^{\mathcal{I}} &= \{x\} \\ r^{\mathcal{I}} &= \{(x, x)\}\end{aligned}$$

This is just the model that contains a single individual x (that is in C) that is related to itself via r . However if we apply the tableaux rules without blocking we will immediately infer that there must exist some individual y such that y is a member of C and that $r(x, y)$. Having inferred this we will now infer that there must exist some individual z such that z is a member of C and $r(y, z)$. Given this new inference we will start the cycle again. This process will repeat itself indefinitely if we do not implement blocking to prevent it. We formally define blocking for $\mathcal{ALC}$ as follows:

Definition 2.3.2 ($\mathcal{ALC}$ Blocking). An individual name b is blocked by an individual name a in an $\mathcal{ALC}$ ABox $\mathcal{A}$ if

- a is an ancestor of b and
- $\{C|b : C \in \mathcal{A}\} \subseteq \{C|a : C \in \mathcal{A}\}$

We say that an individual name b is blocked in $\mathcal{A}$ if it is blocked by some individual name a or if one or more of its ancestors is blocked in $\mathcal{A}$.

This concludes the presentation of the standard consistency algorithm for $\mathcal{ALC}$. We have presented this algorithm here in some detail as numerous reasoning problems for $\mathcal{ALC}$ can be reduced to instances of consistency checking. In particular satisfiability, subsumption, and instance checking can all be reduced to consistency

checks. Since we now have an algorithm for checking consistency we also have an algorithm for each of these problems, provided we give a reduction in each case. For each of the remaining problems we will provide an explanation of this reduction once the problem has been properly introduced. As a final note the algorithm provided above for consistency checking is exponential time in the size of the knowledge base. If the reader is interested in the PSPACE algorithm we point them to the paper by Schmidt-Schauß et al. [25] for the details.

2.3.2 Subsumption Checking

Now that we have a procedure for checking whether there exists an instance of a modeled domain (whether knowledge base has a model) we can start thinking about what consequences we could infer. Perhaps the simplest consequence we would like to infer is whether all individuals in one concept are also members of another. In the context of a library if we have an ontology that represents this domain we might expect to be able to draw some conclusions from it about relationships between different classes of books. For example we would expect it to be true that all textbooks should be non-fiction (even if we do not explicitly encode this information). We would represent the notion that all textbooks are non-fiction as $\text{Textbook} \sqsubseteq \text{NonFiction}$ in both $\mathcal{ALC}$ and $\mathcal{EL}$. This type of relationship, where every individual in some class is a member of another class, is called subsumption in Description Logics.

Definition 2.3.3 (Subsumption Checking). Given a KB $\mathcal{K}$ and concepts C and D . We say that concept C is subsumed by concept D in $\mathcal{K}$ iff in all models of $\mathcal{K}$ every element in the extension of C is in the extension of D . More formally

$$\mathcal{K} \models C \sqsubseteq D \text{ iff } C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \text{ For all models } \mathcal{I} \text{ of } \mathcal{K}.$$

We say that $\mathcal{K}$ entails that C and D are equivalent if C is subsumed by D in $\mathcal{K}$ and vice versa.

An important note on subsumption checking (as well as equivalence checking) is that in Description Logics that support full disjunction these reasoning problems can be reduced to consistency checking (via concept satisfiability) [14]. In $\mathcal{ALC}$ this reduction makes use of the following equivalences [2]

- $C \sqsubseteq D$ iff $C \sqcap \neg D \sqsubseteq \perp$. That is if $C \sqsubseteq D$ then there can be no element that is in C while **not** being in D .
- $C \equiv D$ iff $C \sqcap \neg D \sqsubseteq \perp$ and $D \sqcap \neg C \sqsubseteq \perp$. Similar to above if $C \equiv D$ then there can be no element that is in C (respectively D) that is not in D (respectively C), otherwise the concepts would not be equivalent.

These equivalences are quite intuitive but do not yet give us a way to convert from subsumption checking to consistency checking. After all, we have just traded one subsumption problem for another. We solve this by providing a theorem from the textbook by Baader et al. [2] that builds on these equivalences to reduce subsumption checking to consistency checking.

Theorem 2.3.4. Given an $\mathcal{ALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and two concepts C and D we have the following:

$$\mathcal{K} \models C \sqsubseteq D \text{ iff } (\mathcal{T}, \mathcal{A} \cup \{(C \sqcap \neg D)(x)\}) \text{ is inconsistent}$$

This theorem relies on the first equality presented earlier (there is no element in C that is not in D) and attempts to refute the given subsumption. It does this by specifying that our knowledge base does in fact have an element in C that is not present in D (and as such the subsumption does not hold). Therefore if the subsumption is entailed by the original knowledge base this new information will cause a clash leading to the knowledge base being inconsistent. In this way we can test a subsumption via a consistency check.

Since this reduction requires only the insertion of one new assertion into the knowledge base its output is polynomial in the size of its input. Therefore it follows that the subsumption problem for $\mathcal{ALC}$ is in EXPTIME and can be solved by the consistency algorithm from the previous section.

In contrast to the approach followed for $\mathcal{ALC}$ we cannot attempt to refute a subsumption in $\mathcal{EL}$ in order to check it. This is due to $\mathcal{EL}$ not having a consistency problem in the first place (all $\mathcal{EL}$ knowledge bases are consistent). Therefore we would never find a clash and so could not refute the subsumption. Instead $\mathcal{EL}$ uses a technique called consequence based reasoning, which has the nice property that it runs in polynomial time on the size of the input knowledge base [1].

This consequence based subsumption algorithm tries to directly prove that a subsumption holds by iteratively generating inferences that follow from the terminology. The algorithm is generally presented as a set of rules, which insert new information into the terminology. This chain of inferences is deterministic and since there are only a polynomial number of consequences and the rules that generate consequences are polynomial the overall algorithm is polynomial. Once no more rule applications are possible the terminology will now explicitly contain all subsumption relations and the terminology can now be checked to see if it entails a specific subsumption.

In the example of subsumption we provided previously ($Textbook \sqsubseteq NonFiction$) we represented something that is always true (or at least it should always be true). However there are often examples of things which are only sometimes true, or are only true under certain conditions. For example a textbook on a subject that has since been debunked is no longer non-fictional. We could represent this fact with the statement $Textbook \sqsubseteq NonFiction^{Modern}$, which states that all modern textbooks are non-fiction. It is this type of subsumption we will focus on specifically in this work. We present probabilistic subsumption in detail in Section 5.4 after we have presented the syntax and semantics of $\mathcal{BALC}$.

2.3.3 Satisfiability

Satisfiability, or more specifically concept satisfiability, is the reasoning problem that tests whether it is possible for the extension of a concept to contain an element. That is concept satisfiability checks whether it is possible for the concept to ever be non-empty. A concept C is *unsatisfiable* for a given KB $\mathcal{K}$ if there exists no model of $\mathcal{K}$ where the extension of C is non-empty (there is no model with an element in C). More formally unsatisfiability is defined as follows

Definition 2.3.5 (Unsatisfiability). Given a KB $\mathcal{K}$, and a concept C , the concept C is unsatisfiable with respect to the KB $\mathcal{K}$ iff there is no model $\mathcal{I}$ of $\mathcal{K}$ for which $C^{\mathcal{I}} \neq \emptyset$. We say that a KB is C -unsatisfiable [21] if the concept C is not satisfiable for the KB.

Checking the satisfiability of concepts is often used in ontology debugging as, when modeling a domain, unsatisfiable primitive concepts are usually unintentional bugs [23]. After all there is no point in modeling a fancy concept to represent

Textbooks only to have the consequence that the concept is always empty. As such this is an interesting reasoning problem for any practical Description Logic. Furthermore, the core reasoning algorithm for $\mathcal{BALC}$ relies heavily on previous work in satisfiability checking for $\mathcal{ALC}$. This provides ample reason to both present the classical version of this problem and motivates the study of its probabilistic versions later on in this thesis. We examine total concept satisfiability for $\mathcal{BALC}$ as our first reasoning problem in Section 5.2 and a probabilistic version, called partial concept satisfiability, in Section 5.5.

However, before we can move onto the probabilistic versions of this problem we first need to show how it is solved classically. For $\mathcal{ALC}$ we have previously mentioned that it is possible to reduce subsumption to consistency checking. Furthermore it turns out that satisfiability can also be reduced to subsumption in $\mathcal{ALC}$. Since we have presented both of these problems previously we will provide the reduction to both problems below. Note that we have taken these reductions from the textbook by Baader et al. [2].

Theorem 2.3.6. Given an $\mathcal{ALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a concept C we have that:

$$C \text{ is unsatisfiable iff } \mathcal{K} \models C \sqsubseteq \perp$$

and

$$C \text{ is unsatisfiable iff } (\mathcal{T}, \mathcal{A} \cup \{C(x)\}) \text{ is inconsistent}$$

Since it has been shown that subsumption is in EXPTIME for $\mathcal{ALC}$ this immediately leads to the result that satisfiability is also in the same complexity class (via the above reduction). Furthermore since we can reduce this problem to consistency we can use the exponential time consistency algorithm provided previously to perform satisfiability checks.

For $\mathcal{EL}$ the satisfiability problem is not interesting. This is because in $\mathcal{EL}$ there are no constructors that can lead to unsatisfiability [1]. As such in $\mathcal{EL}$ there are no reasoning procedures for this problem.

2.3.4 Instance Checking

In classical Description Logics the instance checking problem determines whether, in a given KB, a named individual is a member of a given concept for all models. Or in other words whether a named individual is necessarily an *instance* of a concept in a given KB. In practical terms given an ontology of animals this would correspond to checking whether a specific animal *Felix* is a *Cat* (in all models of the ontology). We formally define this as follows:

Definition 2.3.7. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, C a possibly complex concept, and b an individual name. We say that b is an instance of C with respect to $\mathcal{K}$, written $\mathcal{K} \models C(b)$, if $b^{\mathcal{I}} \in C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$.

As in the previous reasoning problems instance checking in $\mathcal{ALC}$ can be reduced to consistency checking. Similar to subsumption we check an instance by attempting to refute it. That is we add to our knowledge base the fact that the given named individual is **not** a member of the given concept. We then pass this modified KB to the consistency algorithm. If this new KB is found to be consistent then there exists a model in which the individual name is not an instance of the concept. Therefore we have that the individual is not an instance of the concept. This is the reasoning followed in the textbook by Baader et al. [1] in order to prove the following theorem:

Theorem 2.3.8. Given an $\mathcal{ALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, a concept C , and an individual name b , $\mathcal{K} \models C(b)$ iff $(\mathcal{T}, \mathcal{A} \cup \{\neg C(b)\})$ is not consistent.

Since this problem is again reduced to consistency checking, in linear time (and increases the size of the KB linearly), it is again in EXPTIME and can be solved using the consistency algorithm provided earlier.

For $\mathcal{EL}$ the problem of checking an instance is actually quite simple. If you want to check if some named individual x is a member of concept C you take each $A_i(x) \in \mathcal{K}$ and then check if $\mathcal{K} \models A_i \sqsubseteq C$. If you find such an A_i then x is an instance of C otherwise it is not. Since each subsumption check is polynomial we have that this process takes at most polynomial-time to complete, giving us a complexity result of polynomial-time for instance checking in $\mathcal{EL}$.

We have now given an overview of both the Description Logic $\mathcal{EL}$ and $\mathcal{ALC}$. Furthermore we have presented several classical reasoning problems for which we define probabilistic versions for in $\mathcal{BALC}$. This provides sufficient background in Description Logics to follow the DL components of $\mathcal{BALC}$. In the next chapter we give a brief overview of general probability and then present Bayesian Networks in some detail. Once we have done this we will use the work presented in the current chapter, as well as the next, in order to introduce the field of Probabilistic Description Logics.

Chapter 3

Bayesian Networks

Before we provide the technical details on Bayesian Networks we first cover some of the basics of probability. In this work we are interested in the probability of random variables. A random variable is a variable whose possible values are the outcomes of a random event. For example, if a coin is flipped we could represent this with a random variable which can take the value of either heads or tails. Random variables such as these are known as finite discrete random variables. This is because they can take on only a certain number (finite) of specific values (discrete). There exists random variables that take on an infinite number of discrete variables; or even take on values from a range of real numbers. However, we will exclusively consider finite discrete random variables in this work.

It will be convenient as we go on to have a way to get the set of values that a random variable can take on. We therefore define for convenience the *val* function which gives this set for a given random variable.

Definition 3.0.1 (*val*). Given a random variable X we define $val(X)$ as the set of values that X can take on. For example if we have a random variable X that represents the outcome of rolling a 6 sided dice then $val(X) = \{1, 2, 3, 4, 5, 6\}$.

Note that we have not yet talked about the actual probabilities associated with a random variable. Probabilities are usually represented in what is called a probability distribution. A distribution is the collective name for all the different methods of recording how likely it is that a random variable takes on its different values. For example if we have a random variable representing a fair coin toss its distribution would record that there is a 0.5 probability for heads and a 0.5 probability for tails. Typically for finite discrete variables distributions are represented as tables.

For most situations the probability of a single random variable is not that useful. The fact that many situations involve more than a single random event illustrates this point. For example even a basic coin flip situation requires a random variable for each time we flip the coin. As such we introduce the concept of worlds to help us represent situations involving multiple random variables.

Definition 3.0.2 (*world*). Given a random variable X and some value $x \in val(X)$ we say that $X = x$ is a valuation of X . Now given a set of random variables V , a world ω is a set of valuations such that for every random variable $X \in V$ it contains exactly one valuation of X .

What this definition intuitively says is that a world is the valuation (giving a random variable a value) of all random variables in some model. Carrying on with our coin flip example if we flip two coins we would have a world where we get heads on the first coin flip and tails on the second, as well as a world for each of the other combinations of heads and tails on the two coin flips.

World	Probability
$X \ Y$	0.5
$\neg X \ Y$	0.3
$X \ \neg Y$	0.2
$\neg X \ \neg Y$	0

FIGURE 3.1: Joint probability distribution over Boolean random variables $V = \{X, Y\}$

The ability to express the outcomes of situations involving randomness using worlds is useful but not sufficient. We will also need a way to express the intermediate state of these situations. For example we would like to be able to say we are going to flip one coin and now want to reason about the probability of the outcome before flipping the second coin. We use partial worlds to encode these intermediate stages.

Definition 3.0.3 (partial world). Given a set of random variables V , a partial world is a set of valuations $(X_1 = x_1, \dots, X_n = x_n)$, where $X_i \in V$ and $x_i \in \text{val}(X_i)$. We say that a partial world is inconsistent if it contains more than one valuation for some random variable.

Note that with this definition a world is just a special case of a partial world where it is consistent and each random variable has taken on some value. This means that we can treat worlds as partial worlds when comparing them. When we are comparing two partial worlds the equality case is obvious. Two partial worlds are equal if they assign the same value to all random variables. A more interesting relationship between partial worlds is satisfaction. We say that a partial world α satisfies another partial world β if they match on all valuations in β . Alternatively we say that α satisfies β if all valuations in β are also in α . We normally use the shorthand $\alpha \models \beta$ to denote this.

We have now defined all constructs we need to model situations involving randomness, and can now get around to representing the probabilities of the outcomes. We do this through what are called joint probability distributions. A joint probability distribution over a set of random variables give the probability that the random variables take on certain values. That is, a joint probability distribution provides a probability for each world of our random variables. For example Figure 3.1 shows a joint probability distribution over the Boolean random variables X and Y . From this distribution we can see that the probability of X and $\neg Y$ is 0.2, which we usually denote as $P(X, \neg Y) = 0.2$. Finally note that the probabilities of all worlds must sum to 1. This is a requirement for a valid probability distribution.

It is obvious to see that given some joint distribution P the probability of any world can be acquired. We next show how we can use this to calculate the probability of a partial world given a joint distribution. The definition we provide here for this is taken from the textbook by Darwiche [11].

Definition 3.0.4. The probability of a partial world α is defined as:

$$P(\alpha) = \sum_{\omega \models \alpha} P(\omega)$$

That is the probability of α is equal to the sum of the probabilities over all worlds in which that satisfy α . Note that if α is inconsistent (has more than one valuation

for some random variable) then there exists no world ω such that $\omega \models \alpha$ and as such $P(\alpha) = 0$ when α is inconsistent.

We are now able to calculate the probability of any outcome of a situation involving randomness (for both partial and full worlds). But we are still not able to deal with evidence. Evidence comes into play when we have observed the outcome of some random events. Once an event has been observed it is no longer random. For example if we flip a coin and it lands on heads it is much more likely that we will get two heads after flipping another coin than before we flipped the first. Intuitively this is because we have bettered our odds from 1 in 4 (only one world out of 4 has two heads) to 1 in 2 (we only need to get one more heads to get two heads). We provide here another definition from the textbook by Darwiche [11] that defines the probability of a world given some evidence.

Definition 3.0.5. Given a partial world ϵ as evidence and world ω we have the following:

$$P(\omega|\epsilon) = \begin{cases} 0 & \omega \not\models \epsilon \\ \frac{P(\omega)}{P(\epsilon)} & \omega \models \epsilon \end{cases}$$

This definition allows the calculation of the probability of a world given evidence. It however does not allow the calculation of the probability for a partial world given evidence. We remedy this by providing another theorem from the textbook [11] that provides the calculation of this probability. Note that the probability of multiple partial worlds holding at the same time can be calculated by considering them to be a single partial world, where this new partial world contains all valuations in all of the original partial worlds. This is usually denoted as $P(\alpha, \beta, \dots)$.

Theorem 3.0.6 (Bayes Conditioning). Given partial worlds α, β we have the following:

$$P(\alpha|\beta) = \frac{P(\alpha, \beta)}{P(\beta)}$$

We have now defined, and explained, all concepts required to understand the Bayesian Networks that are used in this work. We provide an overview of Bayesian Networks, including definitions, in the next section.

3.1 Bayesian Networks

Bayesian Networks [24], also known as belief networks, are a type of probabilistic graphical model that use a Directed Acyclic Graph (DAG) as the underlying graphical structure. This DAG is used to represent dependencies between random variables and makes it necessary to consider only the known dependencies between random variables instead of assuming all variables are dependent.

In a normal joint probability distribution an exponential number of probabilities is required with respect to the number of random variables in the domain [16]. This provides difficulties when asking an expert to estimate probabilities as they would be required to estimate a probability for each complete valuation of the random variables. For example if there are 10 Boolean random variables the expert would need to estimate 1024 probabilities. This becomes even more difficult the larger the network grows as the requirement that the probabilities of all worlds sum to 1 still needs to be satisfied. This leads to each world having a very small probability associated with it, which makes it harder for an expert to estimate. Furthermore it

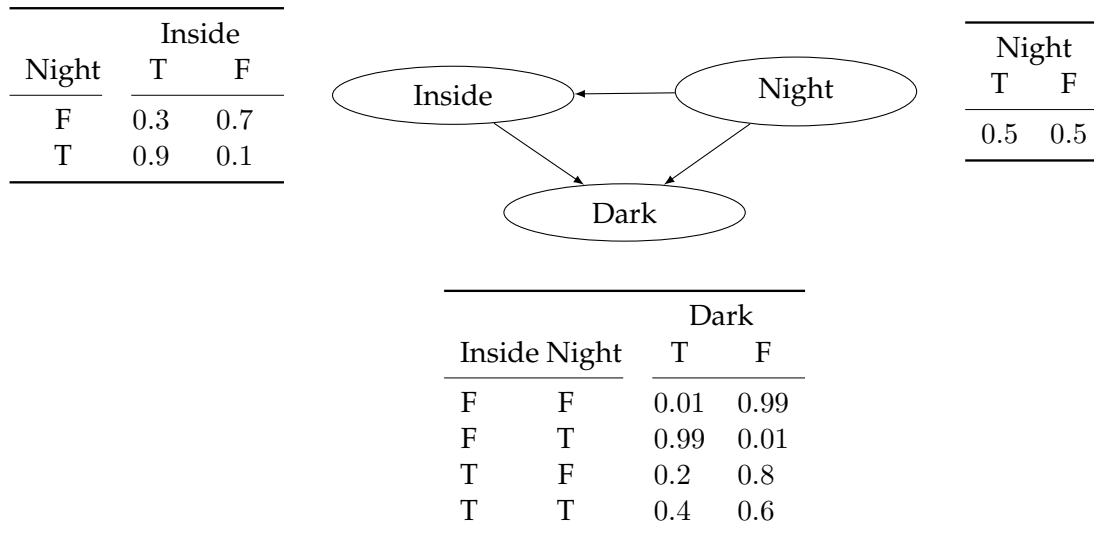


FIGURE 3.2: Example of a Bayesian Network

becomes computationally expensive to manipulate large distributions and they may not fit into memory. As such, for all but distributions over a very small number of random variables, it is not cognitively or computationally efficient to use a normal joint distribution.

Bayesian Networks are a means of representing probability distributions that aim to combat these issues. A Bayesian Network consists of two core components:

- A Directed Acyclic Graph (DAG) where the nodes in the graph represent the random variables in the domain and the edges represent dependencies between random variables. [11]
- A set of local probability models that represent the dependence of each random variable on its parents (as represented in the DAG). These models are typically conditional probability tables (CPTs) [16], but can be other forms of conditional probability distributions (CPDs).

We show an example of a small Bayesian Network in Figure 3.2. This Bayesian Network represents the distribution that indicates how likely it is that it is dark (i.e. no ambient light) in someone's current location. Clearly one of the things which influences light, or its absence, is whether or not it is currently night. We represent this with the node at the top right in the figure. The distribution also takes into account whether the location the person is in is inside a building or not (top left). From the graph we can see that it being Night is not directly dependent on any of the other variables; while it is more likely that people are inside at night. Finally in the CPT for the Dark node we see that: it is really unlikely that it is dark during the day while being outside and vice versa; and it is sometimes dark inside during the day (for example in a room with the lights off).

By utilizing a DAG to represent dependencies (and as such independences) between random variables we are able to specify an exponentially large distribution using only a small number of probabilities (a polynomial number if we fix the maximum number of parents for random variables) [11]. This reduces both the computational and cognitive burden of specifying and manipulating the distribution. Furthermore as Bayesian Networks are guaranteed to define a unique probability

distribution over the network variables [11] they are an attractive representation for probability distributions.

In this thesis we will focus solely on Bayesian Networks over finite discrete random variables. This allows us to make the assumption that the probability model for each random variable will take the form of a Conditional Probability Table (CPT). A CPT specifies the probability of each value of a random variable for each assignment of its parents in table format. Bearing this in mind we formally define a Bayesian Network as:

Definition 3.1.1. A *Bayesian network* is a pair $\mathcal{B} = (G, \Theta)$ where $G = (V, E)$ is a directed acyclic graph and Θ is a set of conditional probability distributions. In the graph G , V is the set of random variables in the distribution and E is the set of dependencies among the random variables. Θ is a set of conditional probability distributions P , one for each node $X \in V$ given its parents:

$$\Theta = \{P(X = x | \pi(X) = x')\}$$

where x and x' represent the valuations of X and $\pi(X)$ (parents of X) respectively.

This definition encodes the local Markov property for Bayesian Networks. That is, every variable is conditionally independent of its non-descendants given a valuation of its parent variables. What this means is that the probability of a random variable is not affected by new information about its non-descendants provided we know the valuation of its parents.

We have now presented sufficient details in order to construct a Bayesian Network. However, we do not yet have a semantics that links this representation to a distribution. We remedy this with the formal definition of the full joint probability of a Bayesian Network.

Definition 3.1.2 (Full joint probability distribution). Every Bayesian Network defines a unique joint probability distribution over its set of random variables V . This distribution can be calculated as:

$$P(V) = \prod_{X \in V} P(X | \pi(X))$$

This is also known as the Chain Rule for Bayesian Networks. The probability of a full valuation of the random variables (world) can be calculated as:

$$P(\omega) = \prod_{X \in V} P(X = x | \pi(X) = x')$$

where x and x' represent the valuations of X and $\pi(X)$ (parents of X) consistent with ω .

Since we have previously stated that the probability calculations involving partial worlds and evidence can be reduced to a series of calculations involving only the probabilities of full worlds we now have that Bayesian Networks can be used to calculate all such inference problems. Note that a wide range of inference algorithms for Bayesian Networks have been developed and inference is usually not done naively as we have just described.

As a final note on Bayesian Networks, it has been shown that inference in Bayesian Networks is NP-hard for general networks, but polynomial when the underlying DAG is a polytree (a graph that forms a tree when the edges are undirected), and linear in the special case where we are calculating the probability of a world [10].

This concludes the overview of the aspects of probability that we will rely on in the remainder of this thesis. In the next chapter we will discuss different ways that probabilistic semantics have previously been combined with Description Logics. This includes both DLs that use Bayesian Networks as well as DLs that use other probabilistic semantics.

Chapter 4

Probabilistic Description Logics

Probabilistic Description Logics (PDLs) are a category of Knowledge Representation (KR) formalisms that combine a classical Description Logic with some form of probabilistic semantics. They are an attempt to enrich existing Description Logics with the capability to represent, or deal with uncertain information. This is useful since classical Description Logics are less suitable in domains where representing information with uncertainty is required [20]. This happens frequently when modeling real world domains.

The development of mature Probabilistic Description Logics would allow many KR systems and ontologies, based on classical DLs, to be applicable to new domains of knowledge. The canonical example of such an application is the Semantic Web, where everybody has the ability to say anything about anything. Probabilistic ontologies would allow reasoning about information on the Semantic Web where conflicting and contradictory information is expected. For example, the reliability of a source could be encoded as a probability. This probability would then propagate as inference is performed using information from this source. Probabilistic Description Logics have also seen use in market analysis[27] where server architecture was modeled using a Bayesian Description Logic.

Although brief, we have now provided some motivation for the merit of further study into PDLs. In part it was this motivation that lead to the work on extending Bayesian $\mathcal{EL}$ ($\mathcal{BEL}$) into Bayesian $\mathcal{ALC}$ ($\mathcal{BALC}$) presented in this thesis. In the next section we present some of the existing work in the field. We provide a brief overview of some PDLs that are related to $\mathcal{BALC}$ in some way. These logics are similar to $\mathcal{BALC}$ in that they either use the same underlying Description Logic ($\mathcal{ALC}$) or have some form of Bayesian Network based probabilistic semantics.

4.1 Overview of Existing Work

Probabilistic Description Logics vary greatly in their underlying Description Logic, probabilistic semantics, and even in what type of information (terminological or assertional) can be encoded. The type of information that can be represented varies in that some PDLs are able to encode only terminological information, while some encode only assertional information, and some can encode both. We will present logics that fall into each of these categories, but focus primarily on PDLs that are similar to $\mathcal{BALC}$ in either their probabilistic semantics or underlying DL. Should the reader wish to get a more holistic overview of the area the survey paper by Lukasiewicz et al. [20] is a good place to start. This paper provides an overview of the topics of possibilistic, probabilistic, and fuzzy Description Logics; and presents multiple DLs that fall into each of these categories.

Before we start presenting existing work we would like to point out that Bayesian $\mathcal{ALC}$ is capable of representing both terminological and assertional information. We would like the reader to bear this in mind while considering other work in the field of PDLs. We start our presentation of existing work by first presenting three previous Description Logics that use $\mathcal{ALC}$ as the underlying DL. Note that none of these previous works use a Bayesian Network as part of the probabilistic semantics and as such are not Bayesian Description Logics like $\mathcal{BALC}$.

In work by Heinsohn[13] the description logic $\mathcal{ALC}$ is extended with probabilistic constraints to create the probabilistic description logic $\mathcal{ALCP}$. $\mathcal{ALCP}$ is able to represent uncertain terminological information but is not capable of representing any assertional information (probabilistic or otherwise). The probabilistic information encoded in $\mathcal{ALCP}$ also does not support conditional dependencies as all that can be expressed is how likely it is that members of one concept are also members of another concept. This information is represented in the form of a range. The intuition behind this particular probabilistic semantics is that all models of an $\mathcal{ALCP}$ knowledge base must have the ratio of overlap between concepts as is specified by the range in the terminology. For example if the knowledge base encodes that the concept *Bird* overlaps with *Flying* in the range $[0.95, 1]$ then in all models between 95% and 100% of *Bird* individuals must be *Flying* individuals.

In work by Jaeger[15] the description logic $\mathcal{ALC}$ is extended with a probabilistic semantics to form the new probabilistic description logic $\mathcal{PALC}$. $\mathcal{PALC}$ is capable of expressing both terminological and assertional probabilistic knowledge. For example $\mathcal{PALC}$ allows statements like $P(\text{FlyingBird}|\text{Bird}) = 0.95$ and $P(\text{Opus} \in \text{AntarcticBird}) = 0.9$, which intuitively say that most birds fly and that Opus is very likely to be an Antarctic bird.

Finally, in work done by Dürig et al.[12] the description logic $\mathcal{ALC}$ is extended to allow probabilistic information about assertional knowledge to be represented. This allows one to encode degrees of belief in concept and role assertions for individuals but does not support probabilistic terminological information.

We now present three previous PDLs that use a Bayesian Network as part of their probabilistic semantics. Bayesian Description logics are a subset of Probabilistic Description Logics where the underlying probabilistic semantics leverage a Bayesian Network in some way. What exactly is represented by this Bayesian Network is dependent on the actual probabilistic semantics of the logic. Furthermore, another notable difference between Bayesian Description Logics is how reasoning is performed. In some work [17, 27] reasoning is reduced to inference in Bayesian Networks, while reasoning in $\mathcal{BEL}$ [4] is more involved. It is important to point out that in previous work performing reasoning using Bayesian inference, without using existing reasoning techniques for Description Logics, requires restrictions to the associated DL. In the work by Yelland [27] an almost trivial DL is used and P-CLASSIC [17] enforces several constraints on terminological knowledge in order to allow inference to be done in the Bayesian Network.

P-CLASSIC [17] is a probabilistic version of the description logic CLASSIC. In addition to terminological knowledge the language is capable of expressing uncertainty about the basic properties of an individual, the number of fillers for its roles, and the properties of these fillers. An effective inference procedure for *probabilistic subsumption* is provided. The provided inference procedure supports up to polynomial time complexity if the underlying Bayesian Network supports polynomial time inference. However, P-CLASSIC does not support assertional knowledge and has certain other restrictions [4, 17]. In P-CLASSIC the DL CLASSIC is restricted

in ways that allows probabilistic subsumption to be calculated almost solely in the underlying Bayesian Network. As such no existing reasoner has to be leveraged.

In work by Yelland [27] the DL used is incredibly basic. The only constructors are conjunction and role quantification (a combination of existential and universal restriction). Nodes in the Bayesian Network are then labeled with DL concepts, which allows reasoning about probability of concept subsumption. It should be mentioned that there are restrictions on how the nodes in the BN may be labeled. For example nodes may not be labeled with concepts containing conjunctions. These restrictions allow reasoning to be reduced to Bayesian Inference and as such no DL reasoner is required.

Finally, we come to Bayesian $\mathcal{EL}$, which is typically abbreviated as $\mathcal{BEL}$. This is the Probabilistic DL on which we directly based $\mathcal{BALC}$. As such we will present it in far greater detail than the other existing work. $\mathcal{BEL}$ uses the lightweight Description Logic $\mathcal{EL}$ as its underlying Description Logic. This design choice gives it good complexity results on the DL end, which means that the overall complexity of the logic is bounded by the complexity of the probabilistic semantics. The semantics of $\mathcal{BEL}$ uses a Bayesian Network in order to encode conditional probabilities. This has the consequence, as we will see in the next section, that there are special cases where reasoning is polynomial (in the Bayesian Network). This has the result, that in the best case, reasoning in $\mathcal{BEL}$ can have polynomial-time complexity. However, in the general case reasoning in $\mathcal{EL}$ is exponential (in the size of the Bayesian Network).

Unlike the other PDLs presented above, $\mathcal{BEL}$ is a contextual PDL. What this means is that in $\mathcal{BEL}$ information is not directly annotated with a probability. Instead terminological information (axioms) are optionally annotated with a context (some valuations of random variables¹). An axiom is only required to hold in probabilistic worlds that satisfy the context of the axiom. In this way the probability that any context is valid is encoded in the Bayesian Network. Inference in $\mathcal{BEL}$ involves checking in which probabilistic worlds a given consequence or entailment holds, and then calculating the probability of these worlds. That is, determining in which worlds the axioms, with contexts satisfied by these worlds, entail the consequence and then calculating the probability of these worlds. For example, if we wanted to calculate the probability that *Textbooks* are *NonFiction* we would find all worlds that have axioms that lead to the consequence that *Textbook* $\sqsubseteq$ *NonFiction*. We would then sum up the probability of each of these worlds to get the probability of the subsumption. We will present this reasoning problem (probabilistic subsumption) and others, as well as more information about $\mathcal{BEL}$, in the next section.

4.2 The Description Logic $\mathcal{BEL}$

The probabilistic Description Logic $\mathcal{BEL}$ is described in its founding work as a Context Sensitive Bayesian Description Logic [4]. This means that the logic is designed to represent certain contextual domain knowledge in uncertain contexts. The original work uses plants making photosynthesis in the presence of light, water, and carbon dioxide as an example. We believe this a good example to illustrate how terminological information is represented in $\mathcal{BEL}$ and present it below.

$$Plant \sqsubseteq \exists makes.Photosynthesis, \{light = t, CO_2 = t, water = t\}$$

¹Note that a valuation of a random variable is just an assignment of a particular value to the variable. This is often denoted as $X = x$ where X is the random variable and x is one of the values it can take on.

This axiom states that Plant must make Photosynthesis if light, CO_2 , and water are present. Importantly it claims nothing otherwise. The likelihood of the existence of water, light, and CO_2 is represented as a probability. This probability is encoded as part of a Bayesian Network; which uses *light*, CO_2 , and *water* as Boolean random variables. Note that for this example the BN would have at least a random variable for each of *light*, *water*, and CO_2 but could additionally have other random variables as well. We mentioned previously that should an axioms context not hold (for example if there is no *water*) then the classical portion of the axiom is not required to hold (*Plants* don't have to make *Photosynthesis*). This is a very important concept to understand as we will often refer to axioms holding. This in turn can happen either if the context is satisfied and the axiom holds classically, or more simply if the context of the axiom is not satisfied.

Work on $\mathcal{BEL}$ is currently ongoing with various different avenues of research being investigated. We next briefly present work published at time of writing. An extension to Bayesian Description Logics[6] has been investigated where a dynamic Bayesian Network is used instead of a conventional Bayesian Network. This research specifically focused on updating probabilities in discrete time steps to reflect changing domain conditions (over time). This allows reasoning about probabilistic entailments at different points in time as well as about future events (given some evidence at different times). The main reasoning task considered in this framework is computing the probability of observing a consequence at different points in time. For example, if we modeled a restaurant we could work out the probability of finding breakfast food at different points in time.

Conjunctive query answering in $\mathcal{BEL}$ [5] has also been investigated. Intuitively this problem takes in a query (in the form of a set of assertions) and determines with what probability the knowledge base entails this query. Multiple problems in this field were investigated with results given for probabilistic query entailment, top-k answers (k most likely answers to the query), and top-k contexts (k contexts entailing the query with largest probability). It was found that these problems are tractable under certain assumptions.

Work has also been done in trying to develop glass box algorithms for reasoning problems in $\mathcal{BEL}$ [7]. That is, algorithms that do not rely on existing reasoners. Specifically an algorithm for exact subsumption, determining the precise probability of a subsumption, was developed. Note that the initial work in $\mathcal{BEL}$ did not use a glass box approach to solve reasoning problems. Algorithms that used existing $\mathcal{EL}$ reasoners were developed instead. While such a blackbox approach has advantages it often requires inefficient use of existing reasoners. This is often due to the requirement of having to work around the capabilities of the existing reasoners. The new glass box algorithm, however, has provided a means to reduce reasoning in $\mathcal{BEL}$ entirely to Bayesian inference. While this result seems promising it introduces problems by violating constraints required for efficient reasoning in BNs. Particularly it negatively impacts the tree width (a measure of how close a BN is to a tree) of the underlying Bayesian Network which makes reasoning more expensive.

Finally, work on $\mathcal{BEL}$ has also included the development of a prototypical reasoner[3]. The developed reasoner is capable of solving probabilistic subsumption problems for $\mathcal{BEL}$ ontologies. The development of the reasoner followed a blackbox approach, where existing technologies were used in order to perform reasoning. In particular the BORN reasoner exploits highly optimized methods from probabilistic logic programming in order to implement the $\mathcal{EL}$ completion rules. This approach is possible because of the simple nature of the logical structures present in $\mathcal{EL}$.

This concludes the overview of existing work on $\mathcal{EL}$, and general PDLs. We next present some of the reasoning problems present in $\mathcal{BEL}$ in more detail. We focus on problems for which we have algorithms for in $\mathcal{BALC}$. As such we present only the problems from the founding work and do not present any of the later results.

4.2.1 Expressivity and Reasoning Complexity

Before considering any probabilistic reasoning problems the type of knowledge that can be represented using $\mathcal{BEL}$ should be understood. We have mentioned that $\mathcal{BEL}$ represents certain domain knowledge in uncertain contexts. Furthermore design decisions taken with regards to the underlying Description Logic and Bayesian Network affect the expressivity of the language as well. The most obvious limitation of this being that domain knowledge can only be encoded in the $\mathcal{EL}$. This DL is very minimal DL and focuses on reasoning efficiency over expressibility, and as such is very restrictive over what can be expressed. Secondly the Bayesian Networks permitted in $\mathcal{BEL}$ ontologies are limited to finite discrete random variables. So more expressive Bayesian Networks, for example Bayesian Networks using continuous random variables, are thus not supported.

Having examined the expressivity of $\mathcal{BEL}$ we now focus on the reasoning problems and complexity of the language. Four reasoning problems were initially defined in $\mathcal{BEL}$ [4]. The problems are: exact subsumption, positive subsumption, certain subsumption, and finding the most likely context for a subsumption.

Positive Subsumption

Informally positive subsumption determines whether a given subsumption holds with a non-zero probability in a given knowledge base. That is, this problem attempts to determine if there exists some probabilistic world (with positive probability) such that the subsumption holds in this world. In $\mathcal{BEL}$ (and in $\mathcal{BALC}$) subsumptions are annotated with a context. As for axioms, this context determines in which worlds the subsumption is required to hold classically. This gives the result that a subsumption can hold in a probabilistic world if either its context is **not** satisfied or if it holds classically.

For example if we wanted to test whether a Plant makes Photosynthesis when there is light ($P(\text{Plant} \sqsubseteq \exists \text{makes. Photosynthesis}, \{\text{light} = t\}) > 0$) we would attempt to find either a Bayesian world which satisfies the context (i.e. where light is present) where plants make photosynthesis or try to find a world which does not satisfy the context (a world without light). If we find such a world and it has positive probability we know that the subsumption is satisfied in at least this world, giving it a positive probability.

Without a formal definition of the semantics of $\mathcal{BEL}$ we cannot present this problem in further detail. We do not provide the required formalisms here to avoid repetition as they overlap greatly with $\mathcal{BALC}$. We will, however, cover the specific problem of positive subsumption, as well as what subsumption means in general in $\mathcal{BALC}$ in far greater detail later.

The positive subsumption problem in $\mathcal{BEL}$ was shown to be NP-Complete[4]. This has the not unexpected consequence that enriching $\mathcal{EL}$ with a Bayesian Network negatively impacts the complexity of the language. In contrast we show that this same problem (positive subsumption) has exponential complexity in $\mathcal{BALC}$, which is in the same complexity class as the subsumption algorithm we previously presented for $\mathcal{ALC}$.

Certain Subsumption

The certain subsumption problem is essentially the dual to the positive subsumption problem. Instead of trying to find at least one world that satisfies the given subsumption we are now interested in showing that there are no worlds, with positive probability, where the subsumption does not hold. Slightly more formally the certain subsumption problem is the decision problem that determines if a subsumption holds with probability 1 in a $\mathcal{BEL}$ knowledge base. Note this problem can be more easily solved by finding a world that does not entail the subsumption (as we described previously). Carrying on with our plant example we would attempt to determine if the knowledge base certainly subsumes our subsumption $Plants \sqsubseteq \exists make.Photosynthesis, \{light = t\}$ by showing that there is no world with light where the subsumption does not hold classically. In $\mathcal{BEL}$ this problem has been shown to be coNP-Complete [4].

Exact Subsumption

The exact subsumption problem, as its name suggests, is the problem of determining the exact probability of a given subsumption. In Bayesian Description Logics this is equivalent to finding all worlds, with non-zero probabilities, that entail a subsumption and summing up their probabilities. For the plants example ($Plants \sqsubseteq \exists make.Photosynthesis, \{light = t\}$) we would have to find all worlds with positive probability in which there is light and plants make photosynthesis, as well as all worlds with positive probability without light. The summation over the probabilities of each of these worlds will give us the probability that this subsumption holds in our knowledge base. In the original work [4] a PSPACE algorithm was provided that solved the exact subsumption problem; we will later present an algorithm for this problem in $\mathcal{BALC}$ that is in EXPTIME.

Most Likely Context

This brings us to the final reasoning problem from $\mathcal{BEL}$ that we define in $\mathcal{BALC}$. The most likely context problem can be informally described as: given a subsumption, $C \sqsubseteq D$, return the context, κ , in which the subsumption holds with the largest probability. Note that, unlike the other subsumption problems, the given subsumption does not have an attached context. This is because we are now attempting to *find* the most likely context for the subsumption. More practically, if we wanted to determine which conditions lead to plants making photosynthesis with the highest probability we could do this by solving an instance of the most likely context problem. We would pass a most likely context reasoning service the subsumption $Plants \sqsubseteq \exists make.Photosynthesis$ and it would return the conditions under which this subsumption holds that has the highest probability. From this informal definition (and example) we can see that this problem is specific to Bayesian Description Logics, or at least to contextual probabilistic DLs. As such this reasoning problem is not present in many other probabilistic Description Logics. In the initial work by Ceylan [4] an EXPTIME algorithm has been provided that solves the most likely context problem. We have formally defined the most likely context problem in $\mathcal{BALC}$, but leave finding an algorithm that solves this problem for future work.

Chapter 5

The Description Logic $\mathcal{BALC}$

In the previous chapters we have presented both classical Description Logics and Bayesian Networks. We then went on to provide an overview of existing probabilistic Description Logics, including previous Bayesian Description Logics. We now build on this and present our probabilistic Description Logic $\mathcal{BALC}$.

In this chapter we present our work on the contextual Bayesian Description Logic Bayesian $\mathcal{ALC}$ ($\mathcal{BALC}$). This includes the syntax, semantics, and some basic reasoning problems. Our work on $\mathcal{BALC}$ was inspired by previous work done by Ceylan [4] on the Bayesian Description Logic $\mathcal{BEL}$ (See Section 4.2) which, to the best of our knowledge is the first work on a contextual probabilistic Description Logic. Using the work on $\mathcal{BEL}$ as a base we have lifted the probabilistic semantics of $\mathcal{BEL}$ to the more expressive Description Logic $\mathcal{ALC}$ and extended the probabilistic semantics to support both assertional and terminological information (both probabilistic and classical). The rest of this chapter is laid out as presented below.

We start by giving the syntax and semantics of the language in Section 5.1. Once we have covered the syntax and semantics of $\mathcal{BALC}$ we present the reasoning problems we have defined. These reasoning problems are a mixture of problems from both classical Description Logics and problems that were defined for $\mathcal{BEL}$. In particular we present work on the following reasoning problems for $\mathcal{BALC}$: concept satisfiability, knowledge base consistency, subsumption, instance checking, and the problem of finding the most likely context for an entailment (definition and possible avenues of future research only).

We have split the concept satisfiability problem into two sections as the problem has two variations in $\mathcal{BALC}$, a total version (which corresponds closely to the classical case) and a partial version (which is a probabilistic version of the problem). In the total version we require that a given concept be satisfiable in all possible worlds, while the partial version only requires that the concept be satisfiable in some world with positive probability. In Section 5.2 we present the former, and in Section 5.5 we present the latter.

After we present the total concept satisfiability problem we show how $\mathcal{BALC}$ consistency can be reduced to total concept satisfiability. This reduction is based on the fact that for a knowledge base to be consistent the top concept must be satisfiable. We present both the consistency problem, and the reduction to total concept satisfiability, in Section 5.3.

We next address subsumption of concepts in $\mathcal{BALC}$ knowledge bases in Section 5.4. We start by formally defining subsumption in $\mathcal{BALC}$ and then introduce three types of subsumption problems in $\mathcal{BALC}$: positive subsumption, exact subsumption, and p -subsumption. In $\mathcal{BALC}$ a subsumption can hold with some probability. We split the subsumption problem into different categories depending on how accurately we want to determine this probability. The positive subsumption problem determines whether a given subsumption holds with a probability greater than

0, the exact subsumption problem calculates the exact probability for a given subsumption, and the p -subsumption problem is to determine if a subsumption holds with probability at least p . We show that the positive subsumption problem can be reduced to an instance of $\mathcal{BALC}$ consistency and provide an algorithm for exact subsumption and p -subsumption.

In Section 5.6 we present instance checking in $\mathcal{BALC}$: checking whether an individual is certainly a member of a given concept in all knowledge bases. We show that this problem can be reduced to an instance of the exact subsumption problem in $\mathcal{BALC}$.

Finally, we formally define the problem of finding the most likely context for a subsumption in Section 5.7. We also outline some possible approaches that we are following to find an algorithm for this problem. Furthermore we explain how we could extend the notion of a most likely context to the other reasoning problems we have defined for $\mathcal{BALC}$.

5.1 Syntax and Semantics

As is implied by the name of our logic we use the existing $\mathcal{ALC}$ concept language as the underlying Description Logic for $\mathcal{BALC}$. As such, the concept language for $\mathcal{BALC}$ is the same as for $\mathcal{ALC}$. Our contribution to this is enriching $\mathcal{ALC}$ with the addition of a probabilistic world semantics based on the semantics of $\mathcal{BEL}$. We do this by defining a probability distribution over a set of worlds where each world corresponds to an $\mathcal{ALC}$ interpretation. Each of these world-interpretation pairs are required to be a model of an $\mathcal{ALC}$ KB defined by the $\mathcal{BALC}$ axioms (minus their contexts) that have their contexts satisfied by the world. We call such an interpretation, linked to a specific world, a V -interpretation. After presenting the syntax of $\mathcal{BALC}$ we describe how V -interpretations, and a distribution over such interpretations, form the basis of the semantics for $\mathcal{BALC}$.

5.1.1 Syntax

Previously we have seen that classical Description Logics have special constructs to group different types of domain knowledge together. Typically this is done by using an ABox to store assertional information, a TBox to store terminological information, and a knowledge base to group together a TBox and an ABox. $\mathcal{BALC}$ follows this same approach but additionally also extends regular $\mathcal{ALC}$ TBoxes and ABoxes with the notion of contexts. In what follows we first define what constitutes a $\mathcal{BALC}$ knowledge base, this includes the new TBox and ABox constructs. We then move on to formally define V -interpretations and the requirements for a V -interpretation to be a model of a $\mathcal{BALC}$ TBox, ABox, and knowledge base. Finally, we use V -interpretations to define probabilistic models for $\mathcal{BALC}$. Note that probabilistic models are the $\mathcal{BALC}$ equivalent of classical $\mathcal{ALC}$ models. However V -interpretations are not the equivalent of classical interpretations. This distinction will become clear once we have defined these constructs.

Definition 5.1.1 (KB). Let V be a finite set of discrete random variables disjoint from the sets of concept and role names. A V -literal is an ordered pair of the form (X_i, x) , where $X_i \in V$ and $x \in \text{val}(X_i)$. We refer to such ordered pairs as literals as they are a generalization of Boolean literals which are often denoted as x or $\neg x$ for the random variable X . We will use this notation for Boolean random variables (X for (X, T) and $\neg X$ for (X, F)) in our examples as it greatly simplifies the presentation.

A V -context is any set of V -literals. We say that a V -context is consistent if it contains at most one pair for each random variable. Most importantly this means that a consistent V -context does not contain any conflicting literals, that is literals which claim a random variable takes on differing values. Note that a context does not have to contain a value for each random variable, and as such could be a partial valuation of the variables in V . We refer to V -contexts as primitive contexts as we later define the notion of a complex context.

A V -restricted general concept inclusion (V -GCI) is an expression of the form $(C \sqsubseteq D)^\kappa$ where C and D are $\mathcal{ALC}$ concepts and κ is a V -context. A V -TBox is a finite set of V -GCIs.

Similarly a V -restricted assertion (V -assertion) is an expression of the form $C(x)^\kappa$ or $r(x, y)^\kappa$ where C is an $\mathcal{ALC}$ concept, r is an $\mathcal{ALC}$ role name, x, y are individual names, and κ is a V -context. A V -ABox is a finite set of V -assertions.

Finally, a $\mathcal{BALC}$ knowledge base (KB) over V is a triple $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ where $\mathcal{B}$ is a Bayesian Network over V , $\mathcal{T}$ is a V -TBox, and $\mathcal{A}$ is a V -ABox.

The idea behind V -GCIs and V -assertions is that the axiom or assertion is only required to hold in the given context. For example if we have the V -GCI $(Student \sqsubseteq \exists attend.Class)^{\neg Vacation}$ our Knowledge Base would require that Students attend at least one Class when it's not vacation, but implies nothing about what students do when it is vacation. This is quite useful as we might have some students that attend summer school in their vacation and some that do not. We could represent information about one such individual as $(\exists attends.Class)(Dave)^{Vacation}$. That is, the assertion that Dave attends class during Vacation. Note that the requirement for GCIs and assertions to have contexts does not prevent the encoding of classical statements. If we want to encode that an axiom or assertion should hold in all contexts we simply annotate it with the empty set. Intuitively this works because if a statement has $\emptyset$ as its context then it is impossible for there to be a world that does not satisfy its context. We sometimes abbreviate $(C \sqsubseteq D)^\emptyset$ as $(C \sqsubseteq D)$ and $C(x)^\emptyset$ as $C(x)$ for convenience. As a further convenience, when it is apparent from the context, we will omit the V prefix and refer only to literals, contexts, GCIs, assertions, ABoxes, and TBoxes.

5.1.2 Semantics

As $\mathcal{BALC}$ is based on a model-theoretic semantics we next need to define what constitutes a model of a $\mathcal{BALC}$ knowledge base. In order to do this we must first define interpretations for $\mathcal{BALC}$. Since $\mathcal{BALC}$ is a Probabilistic Description Logic, interpretations differ from classical $\mathcal{ALC}$ interpretations as they must take into account the probabilistic world semantics of Bayesian Networks as well. In order to take into account the semantics of both $\mathcal{ALC}$ and Bayesian Networks we first define an interpretation for a specific Bayesian World. We do this by defining an interpretation as is usual for $\mathcal{ALC}$ with the addition of a valuation (world) in which this interpretation holds.

Definition 5.1.2 (V -interpretation). Let V be a finite set of discrete random variables disjoint from the sets of concept and role names. A V -interpretation is a tuple $\mathcal{V} = (\Delta^\mathcal{V}, \cdot^\mathcal{V}, v^\mathcal{V})$ where $\Delta^\mathcal{V}$ is a non-empty set called the domain, $v^\mathcal{V}$ is a valuation function defined as $v^\mathcal{V} : V \rightarrow \cup_{X \in V} val(X)$ such that $v^\mathcal{V}(X) \in val(X)$, and $\cdot^\mathcal{V}$ is an interpretation function that maps every concept name C to a set $C^\mathcal{V} \subseteq \Delta^\mathcal{V}$ and every role name r to a binary relation $r^\mathcal{V} \subseteq \Delta^\mathcal{V} \times \Delta^\mathcal{V}$. The interpretation function $v^\mathcal{V}$ is extended to complex $\mathcal{ALC}$ concepts as per usual (see Section 2.2.1).

Given a valuation function $v^\mathcal{V}$, a Bayesian world ω , and a context κ we will use the notation $v^\mathcal{V} = \omega$ when a valuation function assigns each random variable the same value as it has in the Bayesian World ω , $v^\mathcal{V} \models \kappa$ when for all $(X, x) \in \kappa$ we have that $v^\mathcal{V}(X) = x$, and $\omega \models \kappa$ when we have that $\omega = v^\mathcal{V}$ such that $v^\mathcal{V} \models \kappa$.

The idea behind a V -interpretation is that, intuitively, it should be a model of the classic $\mathcal{ALC}$ knowledge base consisting of all statements that have a context that matches the valuation function of the V -interpretation (we will formalize this when we cover the semantics of $\mathcal{BALC}$). However, V -interpretations are not able to model an entire $\mathcal{BALC}$ knowledge base as they focus on only a single world while a Knowledge base can encode information across multiple worlds. We need a formalism that is capable of representing information across multiple worlds. For this purpose we define probabilistic interpretations, which combine multiple V -interpretations and a probability distribution in order to model Description Logic information across probabilistic worlds.

Definition 5.1.3 (Probabilistic Interpretation). A probabilistic interpretation is a pair of the form $\mathcal{P} = (\mathcal{J}, \mathcal{P}_\mathcal{J})$, where $\mathcal{J}$ is a finite set of V -interpretations and $\mathcal{P}_\mathcal{J}$ is a probability distribution over $\mathcal{J}$ such that $\mathcal{P}_\mathcal{J}(\mathcal{V}) > 0$ for all $\mathcal{V} \in \mathcal{J}$.

A probabilistic interpretation is essentially a set of $\mathcal{ALC}$ interpretations which hold only in their associated world (where the world matches the valuation function) with a probability distribution over them and can be thought of as being a claim that for each world the information in the associated V -interpretation holds with the probability specified in the distribution.

We have now defined V -interpretations as well as $\mathcal{BALC}$ TBoxes, ABoxes, and Knowledge bases. We now put these concepts together and define what it means for a V -interpretation to be a model of these constructs.

Definition 5.1.4 (model). We say that the V -interpretation $\mathcal{V}$ is a *model* of the GCI $(C \sqsubseteq D)^\kappa$, denoted as $\mathcal{V} \models (C \sqsubseteq D)^\kappa$, iff (i) $v^\mathcal{V} \not\models \kappa$, or (ii) $C^\mathcal{V} \subseteq D^\mathcal{V}$. Similarly $\mathcal{V}$ is a *model* of the assertion $C(x)^\kappa$ (respectively $r(x, y)^\kappa$), denoted as $\mathcal{V} \models C(x)^\kappa$ (respectively $\mathcal{V} \models r(x, y)^\kappa$), iff (i) $v^\mathcal{V} \not\models \kappa$, or (ii) $x^\mathcal{V} \in C^\mathcal{V}$ (respectively $(x^\mathcal{V}, y^\mathcal{V}) \in r^\mathcal{V}$). It is a model of the TBox $\mathcal{T}$ iff it is a *model* of all the GCIs in $\mathcal{T}$. It is a model of the ABox $\mathcal{A}$ iff it is a *model* of all the assertions in $\mathcal{A}$. It is a model of the knowledge base $\mathcal{K}$ iff it is a *model* of both $\mathcal{T}$ and $\mathcal{A}$.

The crux of this definition is in the requirements V -interpretation must fulfill in order to be a model of some contextual statement (we use the term statement when referring to both axioms or assertions). We have previously explained that axioms and assertions in $\mathcal{BALC}$ are only required to hold when their context is satisfied. This is encoded in the definition of a model in requirements (i) and (ii) for both assertions and axioms. Intuitively requirement (i) just states that when the context of a V -interpretation does not match the context of a statement then the statement is trivially satisfied (since the context is not satisfied it is not required to hold). Requirement (ii) is only relevant in the cases where the valuation function of the V -interpretation satisfies the context and exists to ensure that when the context and the world are compatible the V -interpretation respects said axiom or assertion. That is when the context is satisfied V -interpretations should be models of the statement in the classical sense.

Having now defined what is required for a V -interpretation to be a model of all the components of a $\mathcal{BALC}$ knowledge base we are ready to define what is required for a probabilistic interpretation to be a model of a knowledge base. We describe the use of V -interpretations to form probabilistic models for such a knowledge base.

Definition 5.1.5 (Probabilistic model). Given a probabilistic interpretation $\mathcal{P}$, where $P = (\mathcal{J}, P_{\mathcal{J}})$, we say that this probabilistic interpretation $\mathcal{P}$ is a *model* of the GCI $(C \sqsubseteq D)^{\kappa}$, denoted as $\mathcal{P} \models (C \sqsubseteq D)^{\kappa}$, iff every $\mathcal{V} \in \mathcal{J}$ is a model of $(C \sqsubseteq D)^{\kappa}$. We say that $\mathcal{P}$ is a model of the TBox $\mathcal{T}$ iff every $\mathcal{V} \in \mathcal{J}$ is a model of $\mathcal{T}$.

Similarly we say that $\mathcal{P}$ is a model of the assertion $C(x)^{\kappa}$ (respectively $r(x, y)^{\kappa}$), denoted as $\mathcal{P} \models C(x)^{\kappa}$ (respectively $\mathcal{P} \models r(x, y)^{\kappa}$), iff every $\mathcal{V} \in \mathcal{J}$ is a model of $C(x)^{\kappa}$ (respectively $r(x, y)^{\kappa}$). We say that $\mathcal{P}$ is a model of the ABox $\mathcal{A}$ iff every $\mathcal{V} \in \mathcal{J}$ is a model of $\mathcal{A}$.

Finally we say that the distribution $P_{\mathcal{J}}$ is consistent with the Bayesian Network $\mathcal{B}$ (where the random variables V in $\mathcal{B}$ are the same as in the V -interpretations) if for every possible world ω of the variables in V it holds that

$$\sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} = \omega} P_{\mathcal{J}}(\mathcal{V}) = P(\omega)$$

The probabilistic interpretation $\mathcal{P}$ is a model of the Knowledge Base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ iff it is a (probabilistic) model of both $\mathcal{T}$ and $\mathcal{A}$, and is also consistent with $\mathcal{B}$.

By defining the probability distribution associated with probabilistic models in this way we achieve the result that the probabilities of partial valuations (contexts) is preserved. That is the probabilities are the same as in the Bayesian Network. We prove this in the theorem below.

Theorem 5.1.6 (Probability of a context). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ be a KB and κ a primitive context. For every model $\mathcal{P}$ of $\mathcal{K}$ it holds that

$$\sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \kappa} P_{\mathcal{J}}(\mathcal{V}) = P(\kappa)$$

We will often use the syntax $P(\kappa)$, where κ is a context, when dealing with probabilities associated with the Bayesian Network. This can be done as there is a clear mapping between contexts and partial worlds. For example consider the case where $\kappa = \{(X, x), (Y, y)\}$. Then we have that

$$P(\kappa) = P(\{(X, x), (Y, y)\}) \equiv P(X = x, Y = y)$$

The change between contexts and partial worlds, when dealing with probabilities associated with Bayesian Networks, is purely syntactic and as such we will use them interchangeably.

Proof. By definition it holds that

$$P(\kappa) = \sum_{\omega \models \kappa} P(\omega) = \sum_{\omega \models \kappa} \sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} = \omega} P_{\mathcal{J}}(\mathcal{V}) = \sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \kappa} P_{\mathcal{J}}(\mathcal{V})$$

□

It should be noted that this result, and its proof, carries over exactly from $\mathcal{BEL}$ [9] and is not a new property of $\mathcal{BALC}$. However, we next use this result to prove a new result about conditional probabilities of contexts. That is, we show that conditional probabilities behave as one would expect in $\mathcal{BALC}$.

Theorem 5.1.7 (Conditional probability of a context). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ be a KB and κ, ϵ primitive contexts. For every model $\mathcal{P}$ of the KB it holds that

$$P_{\mathcal{B}}(\kappa|\epsilon) = \frac{\sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \kappa \wedge \epsilon} P_{\mathcal{J}}(\mathcal{V})}{\sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \epsilon} P_{\mathcal{J}}(\mathcal{V})}$$

More informally this theorem states that the probability of some primitive context κ given a set of evidence in the form a primitive context ϵ is equal to the probability of both κ and ϵ holding simultaneously normalized by the probability of the ϵ holding.

Proof. By definition it holds that:

$$P(\kappa|\epsilon) = \frac{P(\kappa, \epsilon)}{P(\epsilon)} = \frac{P(\kappa \wedge \epsilon)}{P(\epsilon)} = \frac{\sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \kappa \wedge \epsilon} P_{\mathcal{J}}(\mathcal{V})}{\sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \epsilon} P_{\mathcal{J}}(\mathcal{V})}$$

□

We will use the conversion between the probability of a context (partial world) and the probability of V -interpretations often in $\mathcal{BALC}$ reasoning problems as it forms a bridge between the distribution present in probabilistic interpretations and the Bayesian Network of a $\mathcal{BALC}$ knowledge base. Next we present another mechanism that will prove useful when dealing with $\mathcal{BALC}$ reasoning problems.

When reasoning about a $\mathcal{BALC}$ knowledge base it will be useful to refer to the specific TBox (or ABox) associated with a specific Bayesian world, that is the TBox (or ABox) containing only the axioms that hold in the given world. We call this reduced TBox (or ABox) a restriction to the world ω , denoted as $\mathcal{T}_{\omega}$ (or respectively $\mathcal{A}_{\omega}$). We formally define the restriction of TBox (or Abox) as follows:

Definition 5.1.8 (Restriction). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ be a $\mathcal{BALC}$ knowledge base. The *restriction* of $\mathcal{T}$ to a world ω of the Bayesian Network $\mathcal{B}$ is the TBox

$$\mathcal{T}_{\omega} := \{(C \sqsubseteq D) | (C \sqsubseteq D)^{\kappa} \in \mathcal{T}, \omega \models \kappa\}$$

Similarly the restriction of $\mathcal{A}$ to a world ω is the Abox

$$\mathcal{A}_{\omega} := \{\alpha | \alpha^{\kappa} \in \mathcal{A}, \alpha \in \{C(x), r(x, y)\}, \omega \models \kappa\}$$

Notice that this definition only allows for restricting the components of a knowledge base to full valuations (worlds) of the variables in the Bayesian Network. We now define restrictions that work for partial worlds (contexts). Intuitively a partial restriction fixes the variables in V to be consistent with a given context κ . Note that now all axioms and assertions that have a context not consistent with κ are trivially satisfied and can be removed from both the ABox and TBox of the given knowledge base. Furthermore as we have fixed all random variables in κ we can remove all literals in κ from the ABox and TBox as well. Formally we define a partial restriction as follows:

Definition 5.1.9 (Partial Restriction). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ be a $\mathcal{BALC}$ knowledge base. The *partial restriction* of $\mathcal{T}$ to a primitive context ϵ of the Bayesian Network $\mathcal{B}$ is the TBox

$$\mathcal{T}_{\epsilon} := \{(C \sqsubseteq D)^{\kappa \setminus \epsilon} | C \sqsubseteq D^{\kappa} \in \mathcal{T}, \omega \models \kappa \text{ for every } \omega \text{ s.t. } \omega \models \epsilon\}$$

Similarly the *partial restriction* of $\mathcal{A}$ to a world ω is the Abox

$$\mathcal{A}_\omega := \{\alpha^{\kappa \setminus \epsilon} \mid \alpha^\kappa \in \mathcal{A}, \alpha \in \{C(x), r(x, y)\}, \omega \models \kappa \text{ for every } \omega \text{ s.t. } \omega \models \epsilon\}$$

Note that unlike in a full restriction the axioms and assertions still have a context associated with them. Also notice that in the case where κ contains a literal for each random variable in V the partial restriction is a full restriction.

Finally, we conclude this section by defining subsumption of concepts in $\mathcal{BALC}$. We will cover the topic of subsumption in detail in section 5.4 but provide the definition here as well, as it will be convenient to be able to use the notion of a contextual subsumption being entailed by a knowledge base in the following section.

Definition 5.1.10 (Subsumption). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ be a $\mathcal{BALC}$ knowledge base, C, D possibly complex concepts, and κ a context. We say that C is contextually subsumed by D w.r.t. $\mathcal{K}$, denoted $\mathcal{K} \models (C \sqsubseteq D)^\kappa$, if every probabilistic model of $\mathcal{K}$ is a probabilistic model of $(C \sqsubseteq D)^\kappa$.

We have now defined all formalisms required to present the first reasoning problem for $\mathcal{BALC}$. Generally knowledge base consistency is presented first in any work dealing with a Description Logic. However we differ from this approach and first present a version of concept satisfiability. We present the work in this order because, unlike in regular $\mathcal{ALC}$, consistency is not the most basic reasoning problem for $\mathcal{BALC}$. In the following section we present the first satisfiability problem for $\mathcal{BALC}$, namely total concept satisfiability.

5.2 Total Concept Satisfiability

We presented the satisfiability problem for $\mathcal{ALC}$ in Section 2.3.3; where we mentioned that a concept C is satisfiable if there is some model that has an individual in C . We also mentioned that having a procedure to test concept satisfiability is often very useful when modeling an ontology as having primitive concepts that are unsatisfiable is usually unintended. This motivated the study of the satisfiability problem in $\mathcal{BALC}$. We have split the concept satisfiability problem for $\mathcal{BALC}$ into two related subproblems namely: total concept satisfiability and partial concept satisfiability. We present the former now and the latter in Section 5.5.

In $\mathcal{BALC}$ we say that a concept C is totally satisfiable in some knowledge base $\mathcal{K}$ if it is satisfiable in all contexts of the knowledge base. That is there is some probabilistic model of $\mathcal{K}$ that has an individual in C in each of its V -interpretations. Alternatively, total concept satisfiability can also be thought of as requiring that a concept be classically satisfiable in each restricted knowledge base $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ where ω corresponds to some probabilistic world. We formalize this definition below:

Definition 5.2.1 (Total Concept Satisfiability). A concept C is totally satisfiable with respect to a $\mathcal{BALC}$ KB $\mathcal{K}$ iff there exists a probabilistic model $\mathcal{P}$ of $\mathcal{K}$ s.t. C contains at least one individual in all of the V -interpretations of $\mathcal{P}$.

This definition means that total concept satisfiability in $\mathcal{BALC}$ is similar to concept satisfiability in $\mathcal{ALC}$ in that both require that a model exists where a given concept is satisfiable (can have an element in some model). The only difference being that in $\mathcal{BALC}$ we propagate this condition to the V -interpretations.

For example consider the KB consisting of the TBox $\mathcal{T}$ with $P(\alpha) > 0$ in the BN:

$$\mathcal{T} = \{(A \sqsubseteq B)^\top, (A \sqsubseteq \neg B)^\alpha\}$$

In this KB the concept A is clearly not totally satisfiable since in any V -interpretation $\mathcal{V}$ s.t. $\mathcal{V} \models \alpha$ we have that A must be empty. However, if we modify the KB slightly by setting $P(\alpha) = 0$ then the concept A is totally satisfiable. This is because if $P(\alpha) = 0$ then there exists no V -interpretation $\mathcal{V}$ s.t. $\mathcal{V} \models \alpha$ and the TBox axiom that forces A to be empty does not need to hold.

Clearly in $\mathcal{ALC}$ there can be no uncertainty about whether a concept is satisfiable or not. However, this is not the case in $\mathcal{BALC}$. In $\mathcal{BALC}$ there may exist some worlds where a concept might not be satisfiable. As such, as we will see in Section 5.5, it is possible for a concept to be satisfiable only in some worlds and thus have some probability of being satisfiable that we can calculate based on the probability of each of these worlds. In a sense the total concept satisfiability problem is a special case of the general probabilistic satisfiability problem where we require a concept to be satisfiable with a probability of 1 (satisfiable in all worlds with positive probability). Finally note that as a consequence of the definition we have the following lemma:

Lemma 5.2.2. An inconsistent $\mathcal{BALC}$ knowledge base (a knowledge base which has no models) has no concept that is totally satisfiable.

Now that we have a formal definition to work from we can start solving the actual reasoning problem. At this point we have no algorithms that solve any reasoning problems for $\mathcal{BALC}$. Therefore we cannot follow the same approach as for $\mathcal{ALC}$ and try to reduce total satisfiability to consistency (we could but since we have no algorithm for consistency the problem would remain unsolved). Instead we found an algorithm that directly solves the total concept satisfiability problem.

The method that we have found is capable of determining whether (or not) a concept is not totally satisfiable. This solves the total concept satisfiability problem since if we show that a concept cannot be not totally satisfiable then clearly it must be totally satisfiable. We test whether a concept is not totally satisfiable by trying to find a full context (containing a literal for each random variable in V) in which the restricted KB is unsatisfiable for the concept classically and where the context has probability greater than 0. If we can do this we have shown that there exists some world in which the concept cannot be satisfied and therefore is not totally satisfiable. Furthermore if it is not possible to find such a context then the concept must be totally satisfiable. We formally state and prove this result in the following theorem:

Theorem 5.2.3. Given a $\mathcal{BALC}$ knowledge base $\mathcal{K}$ and a concept C , C is not totally satisfiable in $\mathcal{K}$ iff there exists a world ω with $P(\omega) > 0$ and C is unsatisfiable in the $\mathcal{ALC}$ knowledge base $(\mathcal{T}_\omega, \mathcal{A}_\omega)$.

Proof. ($\Leftarrow$) If we have a world ω with $P(\omega) > 0$ and C is not satisfiable in $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ then for all models of $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ we have that $C^\mathcal{I} = \emptyset$. We also have that in any probabilistic model $\mathcal{P}$ of $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ there must exist some V -interpretation $\mathcal{V}$, with $v^\mathcal{V} = \omega$ (since we have that $P(\omega) > 0$). Therefore, because $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ is not C -satisfiable, there must be a $\mathcal{V}$, with $v^\mathcal{V} = \omega$, in all probabilistic models of $\mathcal{K}$ where $C^\mathcal{I} = \emptyset$. Thus C is not totally satisfiable in $\mathcal{K}$.

($\Rightarrow$) If C is not totally satisfiable in $\mathcal{K}$ then in all probabilistic models of $\mathcal{K}$ there exists at least one V -interpretation $\mathcal{V}$ with $C^\mathcal{I} = \emptyset$. Now assume that there exists a way to pick V -interpretations, across all probabilistic models of $\mathcal{K}$, such that a V -interpretation with $C^\mathcal{I} \neq \emptyset$ can be found for each ω . This contradicts the initial assumption that C is not totally satisfiable as we are now able to construct a probabilistic model of $\mathcal{K}$ that has $C^\mathcal{I} \neq \emptyset$ in all V -interpretations. Therefore there exists

at least one world ω with $P(\omega) > 0$ that has $C^{\mathcal{I}} = \emptyset$ in all V -interpretations (in all probabilistic models) with $v^{\mathcal{V}} = \omega$. $\square$

Having now found a way to test whether a knowledge base is totally satisfiable we define the constructs that we will use to determine whether a world meeting the requirements exists or not. Firstly we define a new special context ϕ_K^C . We define this context such that all worlds that have the given concept C unsatisfiable will satisfy ϕ_K^C .

Definition 5.2.4 (ϕ_K^C). We use ϕ_K^C to denote the context that describes all worlds that lead to restricted $\mathcal{BALC}$ knowledge bases not being C totally satisfiable. That is $\omega \models \phi_K^C$ iff $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ has C unsatisfiable classically.

This concept is useful as it satisfies the first requirement of Theorem 5.2.3 (we now have all worlds that have our concept unsatisfiable). In order to simplify our presentation we will use ϕ_K when we are using this context in the special case where $C = \top$. Following this approach of using a context to represent a requirement we next define the contexts that will help us deal with worlds with a probability greater than 0.

Definition 5.2.5 (ϕ_B and $\bar{\phi}_B$). We use ϕ_B to denote the context that describes all worlds with probability greater than 0 in the Bayesian Network. That is if $\omega \models \phi_B$ then $P(\omega) > 0$.

We use $\bar{\phi}_B$ to denote the complement of ϕ_B , that is the context that describes all worlds with probability equal to 0 in the Bayesian Network. That is if $\omega \models \bar{\phi}_B$ then $P(\omega) = 0$.

We have defined two contexts here as it is in some cases more efficient to reason about which worlds are not entailed by $\bar{\phi}_B$ rather than considering which worlds are entailed by ϕ_B . Since we have that a Bayesian world has either a probability greater than 0 or a probability equal to 0 we have the following corollary.

Corollary 5.2.6. For any Bayesian Network B , and a world ω we have that $\omega \models \phi_B \vee \bar{\phi}_B$ iff $\omega \models \top$.

Note that while our definitions of these new contexts are consistent with existing $\mathcal{BALC}$ terminology there are situations where some of these contexts cannot exist. For example we can construct an example where two worlds differ in the valuation of only a single random variable such that there exists no context such that both worlds satisfy it. Consider a set of random variables $V = \{X\}$ with $val(X) = \{1, 2, 3\}$. Then if we have $\omega_1 = (X = 1)$, $\omega_2 = (X = 2)$, and $\omega_3 = (X = 3)$ then there is no context which is satisfied by only ω_1 and ω_2 . We deal with this problem by introducing the $\mathcal{BALC}$ context language in Section 5.2.1. However, since we are not yet interested in finding these contexts we delay the presentation of the context language for simplicities sake. Instead, we next use the newly defined contexts to rephrase Theorem 5.2.3 to use contexts instead of worlds.

Theorem 5.2.7. A $\mathcal{BALC}$ knowledge base $\mathcal{K}$ is not totally concept satisfiable, with regards to a concept C , iff $\phi_K^C \wedge \phi_B$ is satisfiable. That is if there exists a probabilistic world ω such that $\omega \models \phi_K^C \wedge \phi_B$ then C is not totally satisfiable in $\mathcal{K}$.

Proof. ($\Rightarrow$) If $\phi_K^C \wedge \phi_B$ is satisfiable then there exists a world ω with $P(\omega) > 0$ with $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ C -unsatisfiable. Therefore in all probabilistic models of $\mathcal{K}$ there exists at least one V -interpretation with $C^{\mathcal{I}} = \emptyset$.

($\Leftarrow$) If $\mathcal{K}$ is not totally satisfiable for a concept C then there exists a world ω with $P(\omega) > 0$ such that $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ is C -unsatisfiable. Therefore we have that $\omega \models \phi_{\mathcal{K}}^C \wedge \phi_B$. $\square$

Notice that while we have claimed that $\bar{\phi}_B$ is useful we have not yet used it. We remedy this by showing that we can use $\bar{\phi}_B$ instead of ϕ_B when testing for total concept satisfiability.

Theorem 5.2.8. For some knowledge base $\mathcal{K}$, a concept C , and for all worlds ω we have that:

$$\omega \models \phi_{\mathcal{K}} \wedge \phi_B \text{ iff } \omega \models \phi_{\mathcal{K}} \text{ and } \omega \not\models \bar{\phi}_B$$

Proof.

$$\begin{aligned} \text{If } \omega \models \phi_{\mathcal{K}} \wedge \phi_B &\iff \omega \models \phi_{\mathcal{K}} \text{ and } \omega \models \phi_B \\ &\iff \omega \models \phi_{\mathcal{K}} \text{ and } \omega \not\models \bar{\phi}_B \end{aligned}$$

$\square$

Being able to convert between $\bar{\phi}_B$ and ϕ_B will prove useful later as it is common for a Bayesian Network to have few entries in its CPTs with zero probability. This is in turn useful because we will later show that we can construct $\bar{\phi}_B$ in such a way that it has only one primitive context for each zero probability entry in a CPT. This results in situations where $\bar{\phi}_B$ is much smaller than ϕ_B .

We have now given an overview of the total concept satisfiability problem and provided some constructs that can be used to solve the problem. We next present an algorithm to construct $\phi_{\mathcal{K}}^C$, ϕ_B , and $\bar{\phi}_B$; and then show how to practically use these contexts to solve the total concept satisfiability problem.

5.2.1 Constructing the context $\phi_{\mathcal{K}}^C$

The algorithm we present for constructing $\phi_{\mathcal{K}}^C$ is based on previous work done by Lee et al. [19, 22] for finding A maximally satisfiable subsets (A-MSS) of Description Logic Terminologies. That is for finding maximal subsets of terminologies such that the concept A is satisfiable. In the existing work this is done by asserting that a concept A has at least one element and then using a tracing algorithm, based on the standard tableaux algorithm, to determine which axioms in the terminology (if any) lead to inconsistency. These axioms can then be excluded to form the A-MSS.

In investigating finding A-MSS Lee et al. also solved the dual problem of finding the minimal inconsistent subsets of a terminology (for a given satisfiability requirement). Informally a minimal inconsistent subset (MinA) of a terminology is one of the smallest subsets of axioms that lead the terminology to be inconsistent. More formally a MinA $M \subseteq \mathcal{T}$ is a subset of a terminology $\mathcal{T}$ such that there exists no $M' \subseteq M$ such that M' is inconsistent. It is this reasoning problem (finding MinAs) that we utilize to determine if a $\mathcal{BALC}$ ontology is totally satisfiable for a given concept.

In the original work each axiom in the terminology ($\mathcal{ALC}$ TBox) is annotated with a unique label. An ABox $\mathcal{A} = \{A(x)^\top\}$ is then created. This ABox contains the assertion that the concept A is satisfiable (A contains individual x) and this assertion is labeled with $\top$. In this case $\top$ represents its corresponding propositional variable and indicates that this assertion must always hold. The expansion rules in figure 5.1 are then applied until each ABox is fully expanded. In the case of this algorithm an ABox is considered fully expanded once none of the tableaux rules can be applied

to it. Once all ABoxes have been fully expanded the MinAs (or A-MSS) are found by looking for clashes in these ABoxes. For each pair of assertions of the form $A(x)^\psi$ and $\neg A(x)^\phi$ we have a clash indicating that the TBox axioms that are consistent with ψ and ϕ lead to inconsistency when holding simultaneously. If there is at least one clash in each final ABox a propositional formula is created using the labels such that the prime implicants of this clash resolution formula are exactly the A-MSS or MinAs (depending on formula construction). Note that if a final ABox contains no clashes then we have found a model of the terminology such that the concept A is satisfiable. Also note that the label $\top$ attached to the original assertion now indicates true in the clash resolution formula.

We have built on this original work since we require full knowledge base support and not just support for terminologies. This turns out to be simpler than may be expected as we just have to show that the existing method holds for knowledge bases as well as terminologies. Furthermore instead of labeling our axioms with unique labels we instead use contexts as labels (which may not be unique and are not Boolean valued) which adds a bit more intricacy.

Preliminaries

We are not yet ready to present our algorithm for concept satisfiability. This is because we have not yet defined all constructs, and terminology, that we will use in the actual algorithm. We present this preliminary information in this section so that the reader can focus on the content of the algorithm when we present it. We start off by presenting the *BALC* context language, and then move onto the terminology used in our algorithm.

We have previously mentioned that our current contexts are not sufficiently expressive to represent the special contexts that we require for solving the total concept satisfiability problem. We also mentioned that we address this by defining something we call the *BALC* context language. We present this context language formally now.

Context Language In the work that follows the primitive contexts we have defined previously are not sufficient for the reasoning algorithms that we have developed. In addition to the ability to encode partial, to full, Bayesian worlds we require a context framework that allows us to make simple logical statements about contexts. In particular we need a language that allows us to express that a statement is required to hold in the following cases:

- Two or more contexts hold (corresponding to propositional conjunction) for example $A(x)^{\phi \wedge \psi}$ which states that x is an instance of A if both context ϕ and context ψ are satisfied, or
- At least one of a set of contexts hold (corresponding to propositional disjunction). For example $A(x)^{\phi \vee \psi}$ which states that x is an instance of A if at least one of ϕ or ψ is satisfied.

Since our contexts are not propositional in nature, we have not restricted the random variables to be Boolean, we cannot simply use propositional logic for our context language. There are two options for dealing with this problem. The first option is to encode our contexts into propositional variables. This would allow us to use propositional logic as in the original work. The second approach is to define some new language that can deal with our contexts. We follow the second approach

and present here a context language with set based semantics that has the expressive power that we require. We formalize this context language by introducing the notion of a complex context (we will refer to the previously defined contexts as primitive contexts from here on).

Definition 5.2.9 (Complex Context). A complex context ϕ is a finite set containing one or more primitive contexts. Note that this allows us to easily convert from primitive contexts to complex contexts by simply enclosing primitive contexts in an additional set. For example the primitive context κ would be converted into the complex context $\{\kappa\}$.

We have previously defined the semantics of contexts by using the notion of a valuation function $v^\mathcal{V}$ (which was associated to a V -interpretation). Recall that this valuation function simply maps each random variable in the BN to one of its possible valuations and as such is equivalent to some Bayesian world ω . We then defined what it meant for a valuation function (world) to satisfy a context. That is $\omega \models \kappa$ when for all $(X, x) \in \kappa$ we have that $v^\mathcal{V}(X) = x$ (where $v^\mathcal{V} = \omega$). We now extend this to complex contexts.

Given a valuation function $v^\mathcal{V}$ and a complex context $\phi = \{\alpha_1, \dots, \alpha_n\}$ we say that $v^\mathcal{V} \models \phi$ iff $v^\mathcal{V}$ satisfies at least one $\alpha_i \in \phi$. This immediately gives the result that if $v^\mathcal{V} \models \kappa$ then $v^\mathcal{V} \models \{\kappa\}$ as complex contexts are consistent with primitive contexts. Since this is the case from this point on we will assume that all contexts are in complex form unless explicitly stated otherwise. Finally we say that $\phi \models \psi$ iff for all $v^\mathcal{V} \models \phi$ then $v^\mathcal{V} \models \psi$, or alternatively $\phi \models \psi$ iff for all Bayesian worlds ω such that $\omega \models \phi$ then $\omega \models \psi$.

The problem of checking if one complex context entails another is actually more involved than it would seem at first. Obviously we could iterate over all probabilistic worlds and then check that each world that entails ϕ entails ψ . However, this is very inefficient for small contexts. To further complicate matters we cannot simply do a comparison between all primitive contexts and check that for each primitive context α in ϕ we have a context β in ψ such that $\alpha \models \beta$. This is easy to see through a small example.

Let $V = \{X, Y\}$ where X and Y are Boolean and let $\phi = \{x\}$ and $\psi = \{xy, x\neg y\}$. Clearly we now have that $\phi \models \psi$ and that $\psi \models \phi$ (since the contexts are equivalent). However, we have no context α in ϕ such that there is a β in ψ with $\alpha \models \beta$. Clearly we need to do more than just naively compare primitive contexts in order to determine entailments between contexts.

We next present an algorithm that solves this problem by merging contexts in ψ , where possible, in order to deal with situations such as the previous example.

Theorem 5.2.10. For complex contexts ϕ and ψ we can check if $\phi \models \psi$ in polynomial time on the size of the contexts using the Algorithm 1.

Proof. In the presented algorithm we have only two nested for loops with no conditions that lead to the outer for loop being run again. Since we know that both loops are finite (since contexts are finite) we have that the algorithm can only run through the outer loop $|\phi|$ times and the inner loop $|\psi|$ times (per outer loop iteration). This gives the result that the algorithm must terminate in $O(|\phi||\psi|)$ time, which is polynomial on the size of the input contexts.

Next assume that the algorithm has returned that $\phi \models \psi$. Then we know that we have iterated over all α in ϕ and either found a β in ψ such that $\alpha \models \beta$ or we have found a set of contexts in ψ_α contained in ψ such that $\alpha \models \psi_\alpha$. Therefore we know

Algorithm 1 Checking complex context entailment

```

1: for  $\alpha \in \phi$  do
2:    $C = \emptyset$ 
3:   for  $\beta \in \psi$  do
4:     if  $\alpha \wedge \beta$  consistent then
5:        $\gamma = \beta \setminus \alpha$ 
6:       if  $|\gamma| = 0$  then
7:          $C[X] = val(X)$ 
8:       else
9:         for  $(X, x) \in \gamma$  do
10:           $C[X] = C[X] \cup \{x\}$ 
11:        end for
12:      end if
13:    end if
14:  end for
15:  for  $X \in keys(C)$  do
16:    if  $|C[X]| < |val(X)|$  then
17:      return  $\phi \not\models \psi$ 
18:    end if
19:  end for
20:  return  $\phi \models \psi$ 
21: end for

```

that if some world ω we have that $\omega \models \phi$ then it models either a primitive context in ψ or some subset of ψ and therefore we know that $\phi \models \psi$.

Finally if the algorithm returns that $\phi \not\models \psi$ then we have that for some α in ϕ we have some random variable X such that $|C[X]| < |val(X)|$. We know that $(X, x) \in \alpha$ (for any x) and therefore there is some world ω such that $\omega \models \{(X, x)\}$ (where $x \notin C[X]$) and $\omega \models \alpha$ but that $\omega \not\models \psi$ and therefore $\phi \not\models \psi$. $\square$

Having now defined complex contexts we next formalize the operations that make up the $\mathcal{BALC}$ context language.

Definition 5.2.11 (Context Language). Given complex contexts $\phi = \{\alpha_1, \dots, \alpha_n\}$ and $\psi = \{\beta_1, \dots, \beta_m\}$ we define the following operations

$$\phi \vee \psi := \phi \cup \psi$$

and

$$\phi \wedge \psi := \bigcup_{\alpha \in \phi, \beta \in \psi} \{\alpha \cup \beta\} = \{\alpha \cup \beta \mid \alpha \in \phi, \beta \in \psi\}$$

That is we define operations that fulfill the roles of propositional disjunction ($\vee$) and propositional conjunction ($\wedge$), where disjunction has the property that either one of the two contexts holds and conjunction requires that both hold. We next show that the definition of the context language actually satisfies these properties.

Theorem 5.2.12. Given complex contexts ϕ and ψ we have the following properties

$$1) \omega \models \phi \vee \psi \text{ iff } \omega \models \phi \text{ or } \omega \models \psi$$

and

$$2) \omega \models \phi \wedge \psi \text{ iff } \omega \models \phi \text{ and } \omega \models \psi$$

Proof for 1). This proof is straightforward as it follows from the definition of a complex context and entailment for complex contexts.

($\Leftarrow$) if $\omega \models \phi_i$, where $\phi_i \in \{\phi, \psi\}$, then there exists a primitive context $\alpha \in \phi_i$ such that $\omega \models \alpha$. Therefore since $\alpha \in \phi \cup \psi$ then $\omega \models \phi \vee \psi$.

($\Rightarrow$) if $\omega \models \phi \vee \psi$ then there exists some primitive context $\alpha \in \phi \cup \psi$ such that $\omega \models \alpha$. Therefore $\alpha \in \phi$ or $\alpha \in \psi$ therefore $\omega \models \phi$ or $\omega \models \psi$. $\square$

Proof for 2). ($\Leftarrow$) if $\omega \models \phi$ and $\omega \models \psi$ then there exists primitive contexts α and β such that $\alpha \in \phi$ with $\omega \models \alpha$ and $\beta \in \psi$ with $\omega \models \beta$. Since both α and β are consistent with ω we have that $\omega \models \alpha \cup \beta$ (note that $\alpha \cup \beta$ is still a primitive context). Now we have $\alpha \cup \beta \in \bigcup_{\alpha_i \in \phi} \bigcup_{\beta_j \in \psi} \{\alpha_i \cup \beta_j\}$ and since $\omega \models \alpha \cup \beta$ we have that $\omega \models \phi \wedge \psi$.

($\Rightarrow$) if $\omega \models \phi_1 \wedge \phi_2$ then there exists some primitive context $\psi \in \phi_1 \wedge \phi_2$ such that $\omega \models \psi$. We know that $\psi \in \bigcup_{\alpha_i \in \phi} \bigcup_{\beta_j \in \psi} \{\alpha_i \cup \beta_j\}$ (by definition). Therefore there exists some primitive contexts α and β such that $\alpha \subseteq \psi$ and $\beta \subseteq \psi$ with $\alpha \in \phi_1$ and $\beta \in \phi_2$. Finally since $\omega \models \psi$ we have that $\omega \models \alpha$ and $\omega \models \beta$. $\square$

Having proved Theorem 5.2.12 we have now shown that complex contexts provide the necessary framework to make the simple logical statements we require for the core reasoning algorithm for $\mathcal{BALC}$. That is we are now able to express that a statement is required to hold either in the case where one or more contexts hold or in the case where an entire set of contexts must hold. We can now see that the contexts ϕ_K^C , ϕ_B , and $\bar{\phi}_B$ are defined in such a way that can exist for a $\mathcal{BALC}$ knowledge base. This can be seen by taking into account that complex contexts are just sets of primitive contexts, which when fully specified are essentially worlds of the Bayesian Network, thus can represent any subset of Bayesian Worlds. Therefore if a world exists such that it satisfies any specific requirements it can be encoded as a primitive context and stored as part of a complex context. Therefore we have that any set of Bayesian worlds can be represented as a complex context.

We next define two special complex contexts that will be of use later, namely the top ($\top$) and bottom ($\perp$) context. Intuitively we would like for $\top$ to be equivalent to any concept that encodes all Bayesian Worlds and for $\perp$ to be equivalent to any context that is inconsistent, that is any context that encodes no worlds of the Bayesian Network. This leads to the following definition:

Definition 5.2.13. For all valuation functions v^ν we have that $v^\nu \models \top$ and $v^\nu \not\models \perp$. Note we also have that if there are n primitive contexts then we have that $v^\nu \models \top$ iff $v^\nu \models \{\alpha_1, \dots, \alpha_n\}$ and $v^\nu \models \perp$ iff $v^\nu \models \emptyset$.

In this sense the top and bottom context are equivalent to their propositional counterparts where top satisfies all valuations and bottom satisfies no valuation. As in propositional logic the definition of $\top$ and $\perp$ immediately have the following lemma:

Lemma 5.2.14. Given $\top$ and $\perp$ defined as above and a complex context ϕ we have the following equivalences for all worlds ω :

$$\omega \models \top \wedge \phi \text{ iff } \omega \models \phi$$

and

$$\omega \models \top \vee \phi \text{ iff } \omega \models \top$$

and

$$\omega \models \perp \vee \phi \text{ iff } \omega \models \phi$$

and

$$\omega \models \perp \wedge \phi \text{ iff } \omega \models \perp$$

where we have that $\phi \equiv \psi$ iff for all Bayesian worlds ω if $\omega \models \phi$ then $\omega \models \psi$ and vice versa.

This lemma, and the concept of equivalence between contexts, is useful when simplifying contexts and leads to the following result.

Theorem 5.2.15. Given a set of random variables V and a context ϕ there exists a context ψ such that $\phi \equiv \psi$ and $|\psi| \leq \exp(|V|)$.

Proof. For a set of random variables V there are $\exp(|V|)$ different Bayesian worlds, where each world ω is a full valuation of all random variables. We have already seen that contexts are simply a way of denoting partial worlds of a Bayesian Network and can as such represent a world as well (simply set each random variable in the context). As such we can trivially construct a new complex context ψ for any context ϕ such that $\psi = \{\alpha \mid \alpha = \omega \text{ and } \omega \models \phi\}$. $\square$

This theorem effectively gives us an upper bound on the size of a context. Note however that it is possible to construct a context that is larger than $\exp(V)$ but we now know that we can replace this context with an equivalent smaller context.

We now have a context language of sufficient expressive power in order to use the reasoning algorithms we have developed for $\mathcal{BALC}$. We next explain the terminology we use in the modified tableaux algorithm.

Modifications to $\mathcal{ALC}$ tableaux Before going into the details of the algorithm for constructing ϕ_K^C we provide here some definitions that are used in the tableaux. In particular we define revised blocking and $\mathcal{A}$ -insertability which replace blocking and equality blocking respectively. This replacement is necessary as the addition of contexts equality blocking and blocking are both too strict. Therefore we weaken them appropriately. We first define the concept of $\mathcal{A}$ -insertability:

Definition 5.2.16 ($\mathcal{A}$ -insertable). An assertion $C(x)^\phi$ is $\mathcal{A}$ -insertable in an ABox $\mathcal{A}$ iff whenever there is a ψ such that $C(x)^\psi \in \mathcal{A}$, then $\phi \not\models \psi$.

The check for $\mathcal{A}$ -insertability is used instead of equality blocking in the tracing algorithm as equality blocking is no longer sufficient. This is because the addition of contexts to assertions means that the same assertion could have differing causes for appearing in the ABox. This needs to be reflected in the context of the assertion. As such we need to not only check whether an assertion is already in the ABox but also, in the case that the assertion is in the ABox, check that the reason (the label of the assertion) is not already present. As such we say that an assertion is $\mathcal{A}$ -insertable if it is not in the ABox or if the label of the assertion is not entailed by the existing label. This results in only strictly weaker labels being applied since if a stronger label (a label entailed by more worlds) is present it will block a weaker label. The check for $\mathcal{A}$ -insertability is also a requirement for termination which we will show later.

The next definition defines an operator that we use to simplify the presentation of the expansion rules in Figure 5.1. We use the $\oplus$ operator to insert assertions into our ABoxes in such a way that it smartly deals with the fact that the assertion may already be present with a different context.

Definition 5.2.17 ($\oplus$). In the expansion rules $\oplus$ is used as shorthand for the following:

$\mathcal{A} \oplus C(x)^\phi := (\mathcal{A} \setminus \{C(x)^\psi\}) \cup \{C(x)^{\phi \vee \psi}\}$ if $C(x)^\psi \in \mathcal{A}$ and $\mathcal{A} \cup \{C(x)^\phi\}$ otherwise.
 $\mathcal{A} \oplus r(x, y)^\phi := (\mathcal{A} \setminus \{r(x, y)^\psi\}) \cup \{r(x, y)^{\phi \vee \psi}\}$ if $r(x, y)^\psi \in \mathcal{A}$ and $\mathcal{A} \cup \{r(x, y)^\phi\}$ otherwise.

This operator is “smart” in the sense that it modifies the context of an assertion (by adding a new context) if it exists in the ABox or inserts the assertion (with its current context) if it does not.

Finally we modify classical blocking to take contexts into account. Classical blocking, as in $\mathcal{ALC}$ tableaux, is not sufficient when finding MinAs using labels [18]. This is because in classical blocking the labels (contexts) are not taken into account. This can result in a situation where an assertion $C(x)^\phi$ is blocked even though it differs from all $C(x)^\psi \in \mathcal{A}$ because the conclusion was reached via different contexts (a different chain of inferences). To get around this we use revised blocking as defined by Lee in his PhD thesis [18].

Definition 5.2.18 (revised blocking). An individual x blocks an individual y iff x is an ancestor of y and for every $C(y)^\psi \in \mathcal{A}$, it is the case that $C(x)^\phi \in \mathcal{A}$ for some ϕ such that $\psi \models \phi$ [19]. An individual x is an ancestor of individual y if x is created before y by the existential rule.

We have now modified the portions of the tableaux algorithm that were not compatible with contexts and are now ready to present the algorithm for constructing $\phi_{\mathcal{K}}^C$. Note that in the presentation we will use following terminology *clash*, *rule application*, and *fully expanded*. We will say that there is a clash in an ABox if it contains contradictory assertions. For example if both $C(a)$ and $\neg C(a)$ are in the same ABox. A rule application refers to applying one of the expansions rules to an ABox in order to generate a new ABox. Finally, we say an ABox is fully expanded if none of the expansions rules can be applied to it, i.e. the preconditions for all rules are not met for the ABox.

Algorithm

Given a $\mathcal{BALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ and a concept C we start by asserting that there exists an individual of concept C by adding it into $\mathcal{A}$. That is we modify $\mathcal{K}$ to $\mathcal{K}' = (\mathcal{T}, \mathcal{A} \cup \{C(x)^\top\}, \mathcal{B})$ where x is a new individual name. We then apply the expansion rules in Figure 5.1 until none are applicable in any of the generated ABoxes or we find a fully expanded ABox with no clashes (an ABox to which no rules can be applied with no clash). If such an ABox is found we return $\phi_{\mathcal{T}} = \perp$ (the context stating that there is no way to create a world such that $\mathcal{K}$ has $\mathcal{A}$ not totally satisfiable). Intuitively we can do this because we have found a model that shows that C is totally satisfiable in $\mathcal{K}$ and therefore cannot be not totally satisfiable.

If at least one clash is found in each completely expanded ABox then we know that there exists some valuation ω s.t. $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ is inconsistent (note this might not be a world as there may be conflicting valuations for some random variable). We now need to construct and return a context encoding these valuations.

We do this by selecting a context representing a clash from each final ABox and then combining these contexts. This newly formed context now has the property that using only axioms and assertions that hold in this context, we find that the knowledge base has C classically unsatisfiable. We explain how we combine clashes next, note that instead of creating these contexts one at a time we create all of them simultaneously. Suppose $\mathcal{A}_1 \dots \mathcal{A}_n$ are the completely expanded ABoxes then

$$\phi_{\mathcal{A}_i}^C = \bigvee_{C(x)^\phi, \neg C(x)^\psi \in \mathcal{A}_i} (\phi \wedge \psi)$$

$\sqcap$ -rule	if 1. $(C_1 \sqcap C_2)(x)^\phi \in \mathcal{A}$, and 2. either $C_1(x)^\phi$ or $C_2(x)^\phi$ is $\mathcal{A}$ -insertable then $\mathcal{A}' := (\mathcal{A} \oplus C_1(x)^\phi) \oplus C_2(x)^\phi$.
$\sqcup$ -rule	if 1. $(C_1 \sqcup C_2)(x)^\phi \in \mathcal{A}$, and 2. both $C_1(x)^\phi$ and $C_2(x)^\phi$ are $\mathcal{A}$ -insertable then $\mathcal{A}' := \mathcal{A} \oplus C_1(x)^\phi, \mathcal{A}'' := \mathcal{A} \oplus C_2(x)^\phi$.
$\exists_1$ -rule	if 1. $(\exists R.C)(x)^\phi \in \mathcal{A}$, and 2. there exists $\alpha \in \phi$ such that $(\exists R.C)(x)^\alpha$ is $\mathcal{A}$ -insertable then $\mathcal{A}' := \mathcal{A} \oplus (\exists R.C)(x)^\alpha$ for all $\alpha \in \phi$
$\exists_2$ -rule	if 1. $(\exists R.C)(x)^\alpha \in \mathcal{A}$, and 2. there is no z such that both $R(x, z)^\alpha$ and $C(z)^\alpha$ are not $\mathcal{A}$ -insertable, and 3. x is not blocked then $\mathcal{A}' := (\mathcal{A} \oplus R(x, y)^\alpha) \oplus C(y)^\alpha$, where y is a new individual name and $y > y'$ for all individual names $y' \in \mathcal{A}$.
$\forall$ -rule	if 1. $\{(\forall R.C)(x)^\phi, R(x, y)^\psi\} \subseteq \mathcal{A}$, and 2. $C(y)^{\phi \wedge \psi}$ is $\mathcal{A}$ -insertable then $\mathcal{A}' := \mathcal{A} \oplus C(y)^{\phi \wedge \psi}$
$\sqsubseteq$ -rule	if 1. $(C \sqsubseteq D)^\phi \in \mathcal{T}, E(x)^\psi \in \mathcal{A}$, and 2. $(\neg C \sqcup D)(x)^{\phi \wedge \psi}$ is $\mathcal{A}$ -insertable then $\mathcal{A}' := \mathcal{A} \oplus (\neg C \sqcup D)(x)^{\phi \wedge \psi}$
$\equiv$ -rule	if 1. $(C \equiv D)^\phi \in \mathcal{T}, E(x)^\psi \in \mathcal{A}$, and 2. $((\neg C \sqcup D) \sqcap (\neg D \sqcup C))(x)^{\phi \wedge \psi}$ is $\mathcal{A}$ -insertable then $\mathcal{A}' := \mathcal{A} \oplus ((\neg C \sqcup D) \sqcap (\neg D \sqcup C))(x)^{\phi \wedge \psi}$

FIGURE 5.1: Expansion rules for constructing $\phi_{\mathcal{K}}^C$

is the context encoding all clashes for the i -th final ABox. After constructing such a context for each final ABox we combine them into the context $\phi_{\mathcal{K}}^C$ as shown below.

$$\phi_{\mathcal{K}}^C = \bigwedge_{i=1}^n \phi_{\mathcal{A}_i}^C$$

This context, $\phi_{\mathcal{K}}^C$, is the cross product of all clashes in the final ABoxes. It contains a context representing each way a clash can be chosen from each final ABox.

Having now that we have explained the construction of $\phi_{\mathcal{K}}^C$ we next show that our algorithm has the desired properties, that is our algorithm terminates and is sound and complete.

Termination, Soundness, and Completeness

In this section we show that the algorithm for constructing $\phi_{\mathcal{K}}^C$ terminates in finite time (termination), that for all worlds that model $\phi_{\mathcal{K}}^C$ they have C unsatisfiable restricted TBoxes and ABoxes (soundness), and that if there exists a world that leads to a restricted knowledge base that it will model $\phi_{\mathcal{K}}^C$ (completeness). We start by showing that our algorithm terminates.

Theorem 5.2.19 (Termination). The tableaux algorithm used to construct $\phi_{\mathcal{K}}^C$ terminates in finite time.

Proof. We show that the algorithm can take at most a finite number of steps before termination. We examine each of the rules presented in Figure 5.1 individually and show that each rule can only be applied a finite number of times.

If we have some $(C \sqcap D)(x)^\phi$ in our ABox we can apply the $\sqcap$ -rule to it at most once per context per individual due to the $\mathcal{A}$ -insertability requirement. This gives us that in the worst case we will have that each individual will feature in all conjunctions present in the TBox $\mathcal{T}$, i.e. we will have a $(C \sqcap D)(x)$ for each conjunction in $\mathcal{T}$, for each individual for each context. We assume for the time being that the number of individuals in $\mathcal{A}$ are finite (we prove this later). We know that $\mathcal{T}$ is finite (so we have only a finite number of conjunctions) and we also know that for any knowledge base the number of contexts is finite. This gives us that there are only ever a finite number of statements such as $(C \sqcap D)(x)^\phi$ in $\mathcal{A}$. Therefore we have that the $\sqcap$ -rule can only be applied a finite number of times.

For the $\sqcup$ -rule again notice that, similar to the $\sqcap$ -rule, it can only be applied once per statement of the form $(C \sqcup D)(x)^\phi$. This is again due to the $\mathcal{A}$ -insertable requirement. So if we again assume that the number of individuals in $\mathcal{A}$ is finite then we have only a finite number of disjunctions, a finite number of contexts, and a finite number of individuals. This means that we only have a finite number of statements of the form $(C \sqcup D)(x)^\phi$. Since we require that new insertions into the $\mathcal{A}$ need to be $\mathcal{A}$ -insertable we can apply the $\sqcup$ -rule only once per $(C \sqcup D)(x)^\phi$. However, unlike the $\sqcap$ -rule, each application of this rule generates two new ABoxes. However in each of these ABoxes all applications of prior rules are preserved. This gives us that new ABoxes are “simpler” than the initial ABox. As such the number of possible rule applications decreases in the set of ABoxes each time we apply the $\sqcup$ -rule. Therefore we have that both the number of ABoxes output by this rule and the number of times this rule can be applied must be finite.

It is easy to see that the $\exists_1$ -rule can only be applied a finite number of times if we make the assumption that the number of individuals is finite. This is because the number of $\exists$ statements is bounded by the size of the finite TBox and the number of contexts is finite. This means that the number of distinct statements that the $\exists_1$ -rule can be applied to are finite without even considering the $\mathcal{A}$ -insertability requirement. However we do require the $\mathcal{A}$ -insertability requirement to ensure that this rule is not repeatedly applied to the same statement.

For the $\exists_2$ -rule consider that the number of statements of the form $(\exists r.C)(x)^\alpha$ is equal to the number of times the $\exists_1$ -rule is applied. We know that this is a finite number therefore we immediately have a finite number of statements that the $\exists_2$ -rule can be applied to. Since the $\exists_2$ -rule can only be applied once per statement we have that this rule is only applied a finite number of times. Note that since this is the only rule which inserts new individuals into an ABox we have that the number of individuals present in any ABox is bounded by the applications of this rule and the number of individual present at the start. This gives the result that there are only a finite number of individuals in any ABox.

The $\forall$ -rule rule can only be applied a finite number of times for similar reasons to the other rules. That is the number of $\forall$ statements is bounded by the size of the TBox, number of individuals, and number of contexts. Since all of these are finite there are only a finite number of statements that this rule can be applied to. Furthermore, since we have an $\mathcal{A}$ -insertability requirement we have that the rule can only be applied once per statement. Therefore this rule can only be applied a finite number of times.

Finally $\sqsubseteq$ -rule and $\equiv$ -rule can only be applied once per individual in the ABox per context (associated with the individual). Since we have that the number of individuals and contexts are finite we have that applications of these rules must be finite.

As all rules can only be applied a finite number of times the expansion must terminate in finite time and since all final ABoxes are finite there can only be a finite number of clashes used to construct the clash resolution formula. $\square$

We next prove an intermediate result that will be useful when proving soundness and completeness. Since the expansion rules in Figure 5.1 are a modified version of the standard $\mathcal{ALC}$ consistency algorithm we have the nice result that we can still use these rules to determine consistency for $\mathcal{ALC}$ knowledge bases. More specifically we have the result that a $\mathcal{BALC}$ knowledge base can only have an unsatisfiable context if the $\mathcal{ALC}$ knowledge base formed by removing all contexts is inconsistent when we require the given concept to be non-empty. We formally state and prove this result in the following theorem:

Theorem 5.2.20. The expansion rules for the total concept satisfiability algorithm (Figure 5.1) will only find a clash-free ABox iff an $\mathcal{ALC}$ consistency algorithm would return consistent if all labels were removed from $\mathcal{K}$. That is the tableaux algorithm here is a valid consistency algorithm for $\mathcal{ALC}$.

Proof. Throughout this proof we will use $\mathcal{A}_L, \mathcal{T}_L$ to indicate ABoxes and TBoxes labeled with contexts and will use $\mathcal{A}_U, \mathcal{T}_U$ to indicate their unlabeled counterparts.

$\sqcap$ -Rule: For this rule it is not possible for either of the algorithms to insert a new assertion into their respective ABox independently. This can be seen by considering the following cases.

Case 1: $\mathcal{A}_L = \{(C_1 \sqcap C_2)^\phi\}$ and $\mathcal{A} = \{(C_1 \sqcap C_2)(x)\}$. In this case neither the labeled or unlabeled ABox contain either of the assertions to be inserted. As such $C_1(x)$ and $C_2(x)$ can trivially be inserted for each algorithm giving us:

$$\mathcal{A}_L = \{(C_1 \sqcap C_2)^\phi, C_1(x)^\phi, C_2(x)^\phi\}$$

and

$$\mathcal{A} = \{(C_1 \sqcap C_2)(x), C_1(x), C_2(x)\}$$

It is obvious in this case that the resulting ABoxes are identical if labels are removed.

Case 2: $\mathcal{A}_L = \{(C_1 \sqcap C_2)^\phi, C_1(x)^\psi\}$ and $\mathcal{A} = \{(C_1 \sqcap C_2)(x), C_1(x)\}$. In this case both ABoxes already contain one of the new assertions. Applying the rules gives us:

$$\mathcal{A}_L = \{(C_1 \sqcap C_2)^\phi, C_1(x)^{\psi \wedge \phi}, C_2(x)^\phi\}$$

and

$$\mathcal{A}_U = \{(C_1 \sqcap C_2)(x), C_1(x), C_2(x)\}$$

Again we can see that all potential ABoxes are equivalent without labels.

Case 3: $\mathcal{A}_L = \{(C_1 \sqcap C_2)^\phi, C_1(x)^\alpha, C_2(x)^\beta\}$ and $\mathcal{A} = \{(C_1 \sqcap C_2)(x), C_1(x), C_2(x)\}$. In this case both ABoxes contain the assertion we are attempting to insert. In the classical case this blocks the rule application entirely which leaves $\mathcal{A}_U$ unchanged. For $\mathcal{A}_L$ either the insertion is blocked if $\alpha \models \phi$ and $\beta \models \phi$ leaving $\mathcal{A}_L$ unchanged or we get

$$\mathcal{A}_L = \{(C_1 \sqcap C_2)^\phi, C_1(x)^{\alpha \wedge \phi}, C_2(x)^{\beta \wedge \phi}\}$$

However in both cases we are left with ABoxes that are identical without labels.

Since we have that in all cases the algorithms output equivalent ABoxes (without labels) we have now shown that the $\sqcap$ -Rule is consistency preserving.

$\sqcup$ -Rule: We again consider all relevant cases and show that the consistency result is the same for both algorithms when labels are removed.

Case 1: $\mathcal{A}_L = \{(C_1 \sqcup C_2)(x)^\phi\}$ and $\mathcal{A}_U = \{(C_1 \sqcup C_2)(x)\}$. In this case neither of the new assertions appear in either ABox and as such both algorithms apply the rule. This gives us:

$$\mathcal{A}_L^1 = \{(C_1 \sqcup C_2)(x)^\phi, C_1(x)^\phi\}, \mathcal{A}_L^2 = \{(C_1 \sqcup C_2)(x)^\phi, C_2(x)^\phi\}$$

and

$$\mathcal{A}_U^1 = \{(C_1 \sqcup C_2)(x), C_1(x)\}, \mathcal{A}_U^2 = \{(C_1 \sqcup C_2)(x), C_2(x)\}$$

In this case $\mathcal{A}_L^i$ and $\mathcal{A}_U^i$ are equivalent for $i \in [1, 2]$ without labels.

Case 2:

$$\mathcal{A}_L = \{(C_1 \sqcup C_2)(x)^\phi, C_1(x)^\alpha, C_2(x)^\beta\}$$

and

$$\mathcal{A}_U = \{(C_1 \sqcup C_2)(x), C_1(x), C_2(x)\}$$

In this case both ABoxes already contain the assertions to be inserted. In the classical case this results in the rule being blocked leaving the original ABox unchanged. For $\mathcal{BALC}$ we have 2 cases: if the new assertions are not $\mathcal{A}$ -insertable then this rule also blocks leaving us with the original ABox. Otherwise we end up with 2 new ABoxes which contain the same assertions with differing labels. However in all cases all ABoxes are equivalent to each other when labels are removed.

Case 3: $\mathcal{A}_L = \{(C_1 \sqcup C_2)(x)^\phi, C_1(x)^\alpha\}$ and $\mathcal{A}_U = \{(C_1 \sqcup C_2)(x), C_1(x)\}$. In this case one of the new assertions already appears in each ABox. For the classical case this blocks and for $\mathcal{BALC}$ this blocks if $\alpha \models \phi$. In the case where both block the original ABoxes stay the same and are equivalent without labels. If $\alpha \not\models \phi$ we get two new ABoxes

$$\mathcal{A}_L^1 = \{(C_1 \sqcup C_2)(x)^\phi, C_1(x)^{\alpha \vee \phi}\}$$

and

$$\mathcal{A}_L^2 = \{(C_1 \sqcup C_2)(x)^\phi, C_1(x)^\alpha, C_2(x)^\phi\}$$

Clearly $\mathcal{A}_L^1$ is equivalent to $\mathcal{A}_U$ without labels. However the additional ABox $\mathcal{A}_L^2$ is not equivalent to $\mathcal{A}_U$ when labels are removed. Note however that if $\mathcal{A}_U$ is inconsistent then both $\mathcal{A}_L^1$ and $\mathcal{A}_L^2$ are inconsistent (both contain a $C_1(x)$) and, in order to not find a clash free ABox both $\mathcal{A}_L^1$ and $\mathcal{A}_L^2$ would need to be inconsistent which would mean $\mathcal{A}_U$ would also be inconsistent. Therefore this additional ABox has no effect on consistency in the unlabeled case.

We have now shown that in all cases the $\sqcup$ -Rule gives the same consistency results in the unlabeled case.

$\exists_1$ -Rule: In order to even consider applying this rule we require an assertion of the form $(\exists r.C)(x)^\phi$ be in $\mathcal{A}_L$. Should this rule then be applicable (new assertion is $\mathcal{A}$ -insertable) we will insert something of the form $(\exists r.C)(x)^\alpha$ into $\mathcal{A}_L$. Note that $(\exists r.C)(x)^\phi$ and $(\exists r.C)(x)^\alpha$ are already identical without labels. This immediately gives us the this rule preserves equivalence minus labels and as such consistency.

$\exists_2$ -Rule: In order to prove this result it is sufficient to consider only the cases where assertions are added to only one of $\mathcal{A}_L$ or $\mathcal{A}_U$. In all other cases they remain equivalent minus labels. Furthermore we only have to consider the case where assertions are added only to $\mathcal{A}_L$ as both the subset blocking and blocking condition are more strict in the unlabeled case (in the unlabeled case conflicting assertions must exist **and** have a stronger context while the unlabeled case only requires existence). We therefore only look at cases that result in

$$\mathcal{A}_U \rightarrow \mathcal{A}_U$$

and

$$\mathcal{A}_L \rightarrow \mathcal{A}_L \cup \{r(x, y)^\phi, C(y)^\phi\}$$

Case 1: $\exists z$ st $\{r(x, y), C(z)\} \subseteq \mathcal{A}_U$ but both $r(x, y)^\kappa, C(y)^\kappa$ are $\mathcal{A}$ -insertable and y not blocked (for $\mathcal{A}_L$). This gives us

$$\mathcal{A}_U = \{(\exists r.C)(x), r(x, z), C(z)\}$$

and

$$\mathcal{A}_L = \{(\exists r.C)(x)^\kappa, r(x, z)^\alpha, C(z)^\beta\} \text{ s.t } \alpha \not\models \phi \text{ and } \beta \not\models \phi$$

After applying the $\exists$ -Rule to $\mathcal{A}_L$ we have

$$\mathcal{A}_L = \{(\exists r.C)(x)^\kappa, r(x, y)^\kappa, r(x, z)^\alpha, C(y)^\kappa, C(z)^\beta\}$$

Therefore we can see that iff $\mathcal{A}_U$ is consistent then $\mathcal{A}_L$ is consistent. If $\mathcal{A}_U$ inconsistent then there exists $\neg C(Z) \in \mathcal{A}_U$ so $\mathcal{A}_L$ inconsistent without labels as well. If $\mathcal{A}_L$ inconsistent then either $\neg C(z)^\alpha \in \mathcal{A}_L$ so $\neg C(z) \in \mathcal{A}_U$ and both are inconsistent (without labels) or there exists $\neg C(y)^\beta \in \mathcal{A}_L$ but y is a new individual in name so $\neg C(y)$ is inferred so $\neg C(y)$ will also be inferred so both ABoxes are inconsistent without labels.

Case 2: y is blocked for $\mathcal{A}_U$ but not for $\mathcal{A}_L$. This gives us $\mathcal{A}_U = \{(\exists r.C)(x), C(z)\}$ and $\mathcal{A}_L = \{(\exists r.C)(x)^\alpha, C(z)^\phi\}$ with $\phi \not\models \alpha$. Applying the $\exists_2$ -Rule gives

$$\mathcal{A}_L = \{(\exists r.C)(x)^\alpha, C(z)^\phi, r(x, y)^\alpha, C(y)^\alpha\}$$

Again we have that $\mathcal{A}_U$ is consistent iff $\mathcal{A}_L$ is consistent (by the same reasoning as the previous case).

$\forall$ -Rule: The $\forall$ -Rule is only applicable when we have both a universal restriction and a role assertion in our ABox, that is $\mathcal{A}_U = \{(\forall r.C)(x), r(x, y)\}$ and $\mathcal{A}_L = \{(\forall r.C)(x)^\phi, r(x, y)^\psi\}$. We now have two cases

Case 1: $C(y)$ is not in our ABox, i.e. $\mathcal{A}_U$ and $\mathcal{A}_L$ are as above. Then applying the $\exists$ -Rule results in

$$\begin{aligned} \mathcal{A}_U &= \{(\forall r.C)(x), r(x, y), C(y)\} \\ \mathcal{A}_L &= \{(\forall r.C)(x)^\phi, r(x, y)^\psi, C(y)^{\phi \wedge \psi}\} \end{aligned}$$

Which are clearly equivalent minus labels.

Case 2: We have a $C(y)$ in our ABox, i.e.

$$\begin{aligned} \mathcal{A}_U &= \{(\forall r.C)(x), r(x, y), C(y)\} \\ \mathcal{A}_L &= \{(\forall r.C)(x)^\phi, r(x, y)^\psi, C(y)^\alpha\} \end{aligned}$$

For $\mathcal{A}_U$ this results in the rule application being blocked and $\mathcal{A}_U$ remains unchanged. For $\mathcal{A}_L$ we have two cases either $\alpha \models \phi \wedge \psi$ and $\mathcal{A}_L$ remains unchanged or we get

$$\mathcal{A}_L = \{(\forall r.C)(x)^\phi, r(x, y)^\psi, C(y)^{\alpha \vee (\phi \wedge \psi)}\}$$

both of which are equivalent, minus labels, to $\mathcal{A}_U$.

$\sqsubseteq$ -Rule and $\equiv$ -Rule: The proof for these rules is identical. Firstly note that both rules attempt to insert the same assertions for $\mathcal{ALC}$ and $\mathcal{BALC}$ (minus labels). Secondly note that these rules are only blocked if the assertion is already in the ABox (for $\mathcal{BALC}$ it is also required that it has a stronger context). Finally note that in the case of $\mathcal{BALC}$ where the assertions are $\mathcal{A}$ -insertable but also already present in the

ABox that new insertions are not added (only the contexts are modified). As such after the application of this rule both ABoxes must have the same assertions in them, and are thus equivalent minus labels. $\square$

Having now proved this intermediate result we are ready to show that our construction of ϕ_K^C is sound. That is if some world ω is entailed by our construction of ϕ_K^C then there can be no model which has an element in C in any V -interpretation consistent with ω .

Theorem 5.2.21 (Soundness). Given ϕ_K^C constructed for a $\mathcal{BALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ using the algorithm above then we have that if $\omega \models \phi_K^C$ then $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ has C unsatisfiable.

Proof. If $\omega \models \phi_K^C$ then there exist an $\alpha \in \phi_K^C$ such that $\omega \models \alpha$. We know that for all final ABoxes $\mathcal{A}_i$ we have that there must exist a clash $A(x)^\phi, \neg A(x)^\psi$ such that $\beta \subseteq \phi \wedge \psi$ and $\beta \subseteq \alpha$, and thus $\omega \models \phi \wedge \psi$. We have seen in the proof for Theorem 5.2.20 that all rules, except for the $\sqcup$ -Rule and $\exists_2$ -Rule, result in output ABoxes identical (without labels) to the standard tableaux algorithm. As such if we assume the cases where the $\sqcup$ -Rule and $\exists_2$ -Rule differ do not occur when finding final ABoxes then all assertions in our final ABoxes are present by correct inference given $\mathcal{K}'$. As such only statements satisfied by ω are required for $\mathcal{K}'$ to be inconsistent. Therefore $(\mathcal{T}_\omega, (\mathcal{A} \cup \{C(x)^\top\})_\omega)$ will be inconsistent, and we know that $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ has C unsatisfiable.

We now consider the cases where $\sqcup$ -Rule and $\exists_2$ -Rule give output ABoxes that are not equivalent without labels to standard tableaux and show that these cases are necessary for correct inference in $\mathcal{BALC}$.

$\sqcup$ -Rule: For the $\sqcup$ -Rule the difference occurs in Case 3 of the previous proof. That is when our starting ABox is $\mathcal{A} = \{(C_1 \sqcup C_2)(x)^\phi, C_1(x)^\alpha\}$ where $\alpha \not\models \phi$. Which gives the following ABoxes after rule application:

$$\mathcal{A}^1 = \{(C_1 \sqcup C_2)(x)^\phi, C_1(x)^{\alpha \vee \phi}\}$$

and

$$\mathcal{A}^2 = \{(C_1 \sqcup C_2)(x)^\phi, C_1(x)^\alpha, C_2(x)^\phi\}$$

Since $\alpha \not\models \phi$ there exists some world ω such that $\omega \models \alpha$ but $\omega \not\models \phi$ so if the Knowledge Base was restricted to such a world then $C_1(x)^\phi$ would not exist (its context is not satisfied) and as such $C_2(x)$ would be inferred classically (since the restricted Knowledge Base has no labels). So clearly the ABox $\mathcal{A}^2$ is required.

$\exists_2$ -Rule: In the case of the $\exists_2$ -Rule both of the cases in the previous proof are different from the unlabeled case. We go through the cases in order and show that the inference is required.

Case 1: $\mathcal{A}_U = \{(\exists r.C)(x), r(x, z), C(z)\}$ and $\mathcal{A}_L = \{(\exists r.C)(x)^\kappa, r(x, z)^\alpha, C(z)^\beta\}$ such that $\alpha \not\models \kappa$ and $\beta \not\models \kappa$. In this case we know that there exists a Bayesian World ω such that $\omega \models \kappa$ and either $\omega \not\models \alpha$ or $\omega \not\models \beta$ (it could be both) therefore if we restrict the KB to ω we no longer have one of $r(x, z)^\alpha$ or $C(z)^\beta$ in $\mathcal{A}_L$ which means we need to create new instances to satisfy $(\exists r.C)(x)^\kappa$.

Case 2: $\mathcal{A}_U = \{(\exists r.C)(x), C(z)\}$ such that z blocks y and $\mathcal{A}_L = \{(\exists r.C)(x)^\alpha, C(z)^\psi\}$ with $\psi \not\models \alpha$. Again we know that there exists a world ω such that the restricted knowledge base has $(\exists r.C)(x)^\phi$ not satisfied so we need to create new instances.

We have now shown that if a clash exists in a final ABox that it is a result of correct inference given $\mathcal{K}'$. As such only statements satisfied by ω are required for $\mathcal{K}'$ to be inconsistent. Therefore $(\mathcal{T}_\omega, (\mathcal{A} \cup \{C(x)^\top\})_\omega)$ will be inconsistent, and we know that $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ has C unsatisfiable. $\square$

We will be ready to show that our algorithm for constructing ϕ_K^C is complete once we prove another intermediate result. We next show that if some world leads to an unsatisfiable restricted knowledge base then it will be entailed by our construction of ϕ_K^C .

Theorem 5.2.22. For a given $\mathcal{BALC}$ knowledge base $K = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ if there exists an ω such that $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ has C unsatisfiable then $\omega \models \phi_K^C$ for a construction of ϕ_K^C using the above algorithm.

Proof. If there exists an ω such that $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ has C unsatisfiable then we know that $(\mathcal{T}_\omega, (\mathcal{A} \cup \{C(x)^\top\})_\omega)$ is inconsistent and therefore $(\mathcal{T}, \mathcal{A} \cup \{C(x)\})$ will be inconsistent without labels. Therefore our algorithm will find a clash in all final ABoxes of $\mathcal{K}'$. However since $(\mathcal{T}_\omega, (\mathcal{A} \cup \{C(x)^\top\})_\omega)$ is inconsistent we know that in all final ABoxes there must be a clash $A(x)^\phi, \neg A(x)^\psi$ such that $\omega \models \phi \wedge \psi$. Therefore we have that $\omega \models \phi_K^C$. $\square$

Finally we conclude this section by proving that the tableaux algorithm for constructing ϕ_K^C is complete, meaning that we have a means of finding worlds that lead to unsatisfiable restricted knowledge bases.

Theorem 5.2.23 (Completeness). The algorithm for constructing ϕ_K^C is complete.

Proof. We have shown that the construction of ϕ_K^C terminates and that it is sound, that is that if $\omega \models \phi_K^C$ then $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ has C unsatisfiable. Since we have now also shown that if there exists an ω such that $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ has C unsatisfiable then $\omega \models \phi_K^C$ we have shown that this relation holds in both directions. That is $\omega \models \phi_K^C$ iff $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ has C unsatisfiable. $\square$

Now that we have shown that our algorithm terminates and is sound and complete all that is left for this section is to show the complexity results for the algorithm.

Definition 5.2.24. $\phi \models \psi$ iff for every Bayesian world ω such that $\omega \models \phi$ then $\omega \models \psi$.

Theorem 5.2.25. For any contexts ϕ, ψ if $\phi \models \psi$ then for all contexts α we have that $\phi \wedge \alpha \models \psi$.

Proof. If $\phi \models \psi$ then for all Bayesian worlds ω such that $\omega \models \phi$ then $\omega \models \psi$. Since for all worlds ω such that $\omega \models \phi \wedge \alpha$ we have that $\omega \models \phi$ and $\omega \models \alpha$ we therefore also have that $\omega \models \psi$ and therefore $\phi \wedge \alpha \models \psi$. $\square$

Theorem 5.2.26 (Complexity of constructing ϕ_K^C). The tableaux algorithm for constructing ϕ_K^C , presented in Figure 5.1, is exponential time in the size of the input knowledge base.

Proof. We assume that knowledge bases are of the form $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$, where we:

$$\mathcal{T} = \{(A_1 \sqsubseteq B_1)^{\psi_1}, \dots, (A_l \sqsubseteq B_l)^{\psi_l}\}$$

and

$$\mathcal{A} = \{C_1(x_1)^{\phi_1}, \dots, C_m(x_m)^{\phi_m}, r_1(y_1, z_1)^{\alpha_1}, \dots, r_n(y_n, z_n)^{\alpha_n}\}$$

Note that since all primitive contexts constructed by the tableaux are conjunctions of contexts that appear in $\mathcal{K}$ we can encode all primitive context created by the tableaux as a binary string of length $|l| + |m| + |n| = |\mathcal{A}| + |\mathcal{T}| = |\mathcal{K}|$, where we use $|X|$ to denote the number of statements in either the knowledge base, TBox, or ABox. We

will use $\|X\|$ to denote the number of symbols in the knowledge base, TBox, or ABox respectively.

We formalize this encoding for a primitive context κ as follows:

$$\kappa = \phi_1 \dots \phi_l \psi_1 \dots \psi_m \alpha_1 \dots \alpha_n$$

where $\phi_i, \psi_j, \alpha_k \in \{0, 1\}$ such that a 1 indicates that the relevant primitive context appears in the final context and 0 that it does not. Since there are exactly $2^{|\mathcal{K}|}$ such binary strings we know that there are at most an exponential number of contexts in $|\mathcal{K}|$ generated by the tableaux.

Next let $\mathcal{A}_f$ be the set of final ABoxes that are generated by the tableaux once it has terminated. Let $\mathbf{I}$ be the set of individuals that appear in ABoxes in $\mathcal{A}_f$, let $\mathbf{C}$ be the set of primitive concepts in $\mathcal{K}$, and $\mathbf{R}$ be the set of roles in $\mathcal{K}$. Then for each $\mathcal{A}_i \in \mathcal{A}_f$ we have at most $2 \cdot |\mathbf{I}| \cdot |\mathbf{C}|$ concept assertions (the assertion that every individual is a member of each concept and its negation). Additionally each concept assertion has an associated context ϕ where $|\phi| \leq 2^{|\mathcal{K}|}$.

Next let the number of rule applications required to infer some $C(x)^\beta$, where β is primitive, be P . Note that inferring $C(x)$ for a primitive context is a worse case scenario since this can happen $2^{|\mathcal{K}|}$ times. This gives us an upper bound of:

$$|\mathcal{A}_f| \cdot 2 \cdot |\mathbf{I}| \cdot |\mathbf{C}| \cdot 2^{|\mathcal{K}|} \cdot P$$

This gives us a lower bound of exponential complexity. In order to determine the upper bound we need to determine the size of both $\mathcal{A}_f$ and $\mathbf{I}$ as well as P . However as long as $|\mathcal{A}_f|, |\mathbf{I}|, P$ are at most exponential we have exponential complexity for the tableaux algorithm.

Size of $\mathcal{A}_f$: We first calculate the size of $\mathcal{A}_f$ based on the input knowledge base. The only rule that results in a new ABox being created is the $\sqcup$ -rule, which forks an ABox and inserts each side of the disjunction into its own ABox. Note that we cannot simply calculate the final number of ABoxes based on the number of disjunctions in our knowledge base as there are other rule applications which may create new disjunctions. In particular both the $\equiv$ -rule and $\sqsubseteq$ -rule insert new disjunctions (two in the case of the $\equiv$ -rule and one for the $\sqsubseteq$ -rule). Furthermore all conjunctions could be negated which turns them into disjunctions.

This gives us the result that we have a total of $2 \cdot |\equiv| + |\sqsubseteq| + |\sqcup| + |\sqcap| \leq 2 \cdot \|\mathcal{T}\|$ possible disjunctions in any given knowledge base. Next note that the $\sqcup$ -rule can in the worst case be applied for each context for an individual. This gives us a worst case upper bound in the size of $\mathcal{A}_f$ of:

$$|\mathcal{A}_f| \leq |\mathbf{I}| \cdot 2^{|\mathcal{K}|} \cdot 2^{2 \cdot \|\mathcal{T}\|}$$

size of $|\mathbf{I}|$: The number of distinct individuals in the final ABoxes clearly must be equal to the number of individuals initially in $\mathcal{A}$ plus all new individuals created by the tableaux algorithm. Since only the $\exists_2$ -rule inserts new elements we know that the number of individuals is equal to the number of original individuals plus the number of applications of the $\exists_2$ -rule. Next note that the $\exists_2$ -rule can only be applied if it is not blocked and since it works only with primitive contexts it can only be applied at most $2^{|\mathcal{K}|}$ times per instance in the TBox (note that negations present in the TBox can turn $\forall$ into $\exists$). This gives us an upper bound on new individuals inserted of:

$$|\mathbf{I}| \leq |\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|}$$

size of P If we infer some $D(x)^\beta$, where β is primitive, we applied the $\sqsubseteq$ -rule and $\equiv$ -rule at most $|\mathbf{I}| \cdot |\mathcal{T}| \cdot 2^{|\mathcal{K}|}$ (once per individual per context per axiom). We applied both the $\sqcap$ -rule and $\sqcup$ -rule a maximum of $|\mathbf{I}| \cdot \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|}$ times (once per individual per occurrence in $\mathcal{K}$ per context). We have already seen that the $\exists$ -rule has an upper bound of $\|\mathcal{K}\| \cdot 2^{|\mathcal{K}|}$ applications. Finally we can apply the $\forall$ -rule once per individual per occurrence in $\mathcal{K}$ per context per role (note that we insert roles with the $\exists$ -rule but that these insertions are bounded by $\exists$ -rule application). This gives us an upper bound of $|\mathbf{I}| \cdot \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|} \cdot |\mathbf{R}| = |\mathbf{I}| \cdot \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|} \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|})$ $\forall$ -rule applications. Putting all this together we have:

$$|\mathbf{I}| \cdot |\mathcal{T}| \cdot 2^{|\mathcal{K}|} + 2 \cdot |\mathbf{I}| \cdot \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|} + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|} + |\mathbf{I}| \cdot \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|} \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|})$$

as an upper bound on the number of rule applications. Note that each rule application involves up to 2 context entailment queries ($\phi \models \psi$) which are polynomial in the size of the input contexts. Since our contexts are bounded by $2^{|\mathcal{K}|}$ we have the following upper bound on complexity:

$$5 \cdot |\mathbf{I}| \cdot \|\mathcal{K}\| \cdot 2^{2 \cdot |\mathcal{K}| + 1} \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|})$$

Final complexity Putting all this together we have an upper bound for the complexity of the tableaux algorithm of:

$$\begin{aligned} & |\mathcal{A}_f| \cdot 2 \cdot |\mathbf{I}| \cdot |\mathbf{C}| \cdot 2^{|\mathcal{K}|} \cdot P \\ & \leq |\mathbf{I}| \cdot 2^{|\mathcal{K}|} 2^{2 \cdot \|\mathcal{T}\|} 2 \cdot |\mathbf{I}| \cdot |\mathbf{C}| \cdot 2^{|\mathcal{K}|} \cdot 5 \cdot |\mathbf{I}| \cdot \|\mathcal{K}\| \cdot 2^{2 \cdot |\mathcal{K}| + 1} \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|}) \\ & \leq 10 \cdot |\mathbf{I}|^3 \cdot 2^{4 \cdot |\mathcal{K}| + 1} \cdot \|\mathcal{K}\|^2 \cdot 2^{2 \cdot \|\mathcal{T}\|} \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|}) \\ & \leq 10 \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|})^3 \cdot 2^{4 \cdot |\mathcal{K}| + 1} \cdot \|\mathcal{K}\|^2 \cdot 2^{2 \cdot \|\mathcal{T}\|} \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|}) \\ & \leq 10 \cdot 2^{4 \cdot |\mathcal{K}| + 1} \cdot \|\mathcal{K}\|^2 \cdot 2^{2 \cdot \|\mathcal{T}\|} \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|})^4 \end{aligned}$$

Which gives us an upper bound of $O(2^{|\mathcal{K}|})$ for the worst case complexity of the tableaux algorithm. $\square$

Theorem 5.2.27 (Complexity of converting from $\mathcal{A}_i$ to $\phi_{\mathcal{A}_i}^C$). The complexity of converting all final ABoxes to a their respective context encoding all clashes is exponential time in the size of the input knowledge base to the tableaux algorithm.

Proof. We know that the size of any $\mathcal{A}_i \in \mathcal{A}_f$ is bounded above by $2 \cdot |\mathbf{C}| \cdot |\mathbf{I}| \leq 2 \cdot |\mathbf{C}| \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|})$. Furthermore we have that the complexity of constructing $\phi \wedge \psi$ for any complex concepts ϕ and ψ is $|\phi| \cdot |\psi| \leq 2^{2 \cdot |\mathcal{K}|}$. Both of these are exponential in the number of statements in the knowledge base. Note that in both cases the cause of the exponential complexity arises from the contexts.

Therefore in the worst case, where $\mathcal{A}_i$ has that all individuals are a member of all concepts for all contexts, we have to perform $|\mathbf{I}| \cdot |\mathbf{C}|$ conjunctions of contexts. This gives us the following complexity for each conversion:

$$|\mathbf{C}| \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|}) \cdot 2^{2 \cdot |\mathcal{K}|}$$

Since we have to do this for each $\mathcal{A}_i$ we have a total complexity of:

$$|\mathcal{A}_f| \cdot |\mathbf{C}| \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|}) \cdot 2^{2 \cdot |\mathcal{K}|} \leq |\mathbf{I}| \cdot 2^{|\mathcal{K}|} \cdot 2^{2 \cdot \|\mathcal{T}\|} \cdot |\mathbf{C}| \cdot (|\mathcal{A}| + \|\mathcal{K}\| \cdot 2^{|\mathcal{K}|})$$

Which is in the complexity class of $O(2^{|\mathcal{K}|})$, giving us exponential complexity for converting from the set of final ABoxes to the set of contexts encoding the clashes in the final ABoxes. $\square$

Since the primitive contexts that make up each $\mathcal{A}_i$ are still formed solely from conjunctions of contexts created by the tableaux algorithm we have the following lemma.

Lemma 5.2.28. The size of $\phi_{\mathcal{A}_i}^C$ is exponential in the number of statements in the knowledge base. That is $|\mathcal{A}_i| \leq 2^{|\mathcal{K}|}$.

Theorem 5.2.29. The complexity of constructing $\phi_{\mathcal{K}}^C$ from the set of $\phi_{\mathcal{A}_i}$ is exponential in the size of the input knowledge base.

Proof. We can construct $\phi_{\mathcal{K}}^C$ from the set of $\phi_{\mathcal{A}_i}^C$ by picking two of the contexts $\phi_{\mathcal{A}_j}^C$ and $\phi_{\mathcal{A}_k}^C$ and constructing the new context $\phi_{\mathcal{A}_j}^C \wedge \phi_{\mathcal{A}_k}^C$. This new contexts size is again bounded above by $2^{|\mathcal{K}|}$.

As such if we then take another context from the set of $\phi_{\mathcal{A}_i}^C$ and join the new context and this $\phi_{\mathcal{A}_i}^C$ we again get a context that has at most $2^{|\mathcal{K}|}$ elements. As such the cost of joining the contexts does not increase based on the number of contexts we join (there is no combinatorial explosion). While we do have to do an exponential amount of work and exponential number of times the overall complexity of this is still $O(2^{|\mathcal{K}|})$. $\square$

Theorem 5.2.30. The complexity of the entire process of constructing $\phi_{\mathcal{K}}^C$ is exponential in the size of the input knowledge base.

Proof. We have seen that each subprocess involved in constructing $\phi_{\mathcal{K}}^C$ (initial tableaux, constructing all $\phi_{\mathcal{A}_i}^C$, constructing $\phi_{\mathcal{K}}^C$) is in the complexity class of $O(2^{|\mathcal{K}|})$. Since we run each of these processes sequentially we have a final complexity of $O(2^{|\mathcal{K}|}) + O(2^{|\mathcal{K}|}) + O(2^{|\mathcal{K}|})$ which is again in $O(2^{|\mathcal{K}|})$. $\square$

Having proved the completeness and complexity of our algorithm for constructing $\phi_{\mathcal{K}}^C$ we are almost ready to wrap up total concept satisfiability. We have mentioned previously finding worlds that lead to inconsistent restricted knowledge bases is only part of the requirement of determining whether a concept is totally satisfiable. We now need a means of determining whether these worlds have a probability greater than 0. Instead of directly calculating this value for all worlds we construct a context which encodes the worlds that have this property. We detail this construction in the following section.

5.2.2 Constructing $\phi_{\mathcal{B}}$ and $\bar{\phi}_{\mathcal{B}}$

In this section we provide a method to construct $\phi_{\mathcal{B}}$, that is the context encoding all valuations of the Bayesian Network that have non-zero probabilities, and its complement $\bar{\phi}_{\mathcal{B}}$, the context that encodes all valuations with probability 0. We start by constructing $\bar{\phi}_{\mathcal{B}}$ first and then using this context to construct $\phi_{\mathcal{B}}$. While this may seem roundabout it is simpler to construct $\bar{\phi}_{\mathcal{B}}$ than it is to construct $\phi_{\mathcal{B}}$. Furthermore

we have previously shown that it is possible to convert between the two contexts in reasoning problems.

When constructing $\bar{\phi}_B$ we start by first constructing a context that encodes all 0 probability valuations in each CPT of the Bayesian Network and then combining these contexts to create $\bar{\phi}_B$. We can do this in time linear on the number of CPTs (at most one per random variable) and the size of the largest CPT of the Bayesian Network. We describe this process in more detail below.

Constructing $\bar{\phi}_B$

For some random variable $X \in V$ with parents $Y_1, \dots, Y_n$ and for all $x \in \text{val}(X)$, $y_i \in \text{val}(Y_i)$ such that $P(X = x | Y_1 = y_1, \dots, Y_n = y_n) = 0$ we create a primitive context

$$\phi_{X_k} = \{(X, x), (Y_1, y_1), \dots, (Y_n, y_n)\}$$

where k is the number of this 0 probability in the CPT. We then construct the context that encodes all valuations of X and its parents that have 0 probability,

$$\phi_X = \vee_k(\phi_{X_k}) = \bigcup_k (\phi_{X_k})$$

Finally we use these contexts to construct $\bar{\phi}_B$,

$$\bar{\phi}_B = \vee_{X \in V}(\phi_X) = \bigcup_{X \in V} (\phi_X)$$

Once we have constructed $\bar{\phi}_B$ for some $\mathcal{BALC}$ knowledge base we can construct ϕ_B by using this context. We do this because it is more complicated to encode non-zero probability contexts as they must be inconsistent with **all** context with zero probability. That is for all $\alpha \in \phi_B$ there can exist no $\beta \in \bar{\phi}_B$ such that $\beta \subseteq \alpha$ or we would have that $P(\alpha) = 0$. This makes it more difficult to construct ϕ_B as it must consist of primitive contexts that are not consistent with any primitive contexts in $\bar{\phi}_B$ which mean far more variables must be specified for each context. However since we have already shown that $\bar{\phi}_B$ can be used instead of ϕ_B in our algorithm this is not a problem. We provide a way to construct ϕ_B for completeness reasons rather than necessity.

Constructing ϕ_B

For a $\mathcal{BALC}$ knowledge base $\mathcal{K}$ we construct ϕ_B by creating an empty set and then for each Bayesian World ω if $\omega \not\models \bar{\phi}_B$ then create a context ϕ such that $\phi = \omega$ and add ϕ to ϕ_B .

Complexity

While the construction of $\bar{\phi}_B$ was described as a three stage process it is actually just a scan over each CPT of the Bayesian Network where we add primitive context to $\bar{\phi}_B$ each time a 0 entry is found in a CPT. This gives us the previously stated result that constructing $\bar{\phi}_B$ takes time linear in the number of CPTs and the size of the largest CPT. While this result seems good it is misleading in that the size of the largest CPT is (in the worst case) exponential on the number of random variables. This occurs if we have a random variable that is dependent on all other random variables in the network.

In terms of space complexity we have a similar initially good seeming result with bad worst case complexity. We have seen in the construction of $\bar{\phi}_B$ that this complex context contains a primitive context for each 0 in a CPT of the Bayesian network. In the case where all worlds except one have probability 0 this would mean that $\bar{\phi}_B$ would have an primitive contexts for all but one valuation in all CPTs. Since the size of the biggest CPT is exponential on the number of random variables in the networks we again have the result that in the worst case $\bar{\phi}_B$ is exponential in size.

5.2.3 Using ϕ_K^C and $\bar{\phi}_B$ to determine total concept satisfiability

We have previously shown that for a given knowledge base $\mathcal{K}$ if we can find a Bayesian world ω such that $\omega \models \phi_K \wedge \phi_B$ or $\omega \models \phi_K$ and $\omega \not\models \bar{\phi}_B$ then $\mathcal{K}$ is not totally concept satisfiable for C . Since we have now provided methods for constructing both ϕ_K and $\bar{\phi}_B$ we now only have to find a method to find a Bayesian world that satisfies the requirements.

Naively we could do this by iterating over all possible worlds and then comparing each world to contexts in ϕ_K until we find a context that is satisfied by the world and then checking that no context in $\bar{\phi}_B$ is satisfied by the world (this is again a comparison to all contexts in $\bar{\phi}_B$). If we find such a world we can terminate and we know that the knowledge base is not totally concept satisfiable for C , and if no such world exists then the C is totally satisfiable in $\mathcal{K}$. However, although Bayesian inference is inherently exponential in complexity, what we are attempting to do give us the option of using an algorithm that is a slightly more elegant.

Note that if we have some primitive concept $\alpha \in \phi_K^C$ such that none of the contexts in $\bar{\phi}_B$ are entailed by α then we know that there exists some $\omega \models \alpha$ such that $P(\omega) > 0$. We can do this by simply comparing each $\alpha \in \phi_K^C$ to all primitive concepts in $\bar{\phi}_B$ until we find an alpha which does not entail any of the context in $\bar{\phi}_B$. Clearly this is still exponential in the worst case, however, if there are few contexts that lead to unsatisfiable restricted knowledge bases this approach is far more efficient than iterating over all possible worlds of the Bayesian Network.

Complexity of Total Concept Satisfiability: We have shown that we can test total concept satisfiability by finding a world that has probability greater than 0 that leads to an inconsistent restricted knowledge base. We split this problem into 3 sections: finding all worlds that lead to inconsistent restricted knowledge bases (constructing ϕ_K^C), finding all worlds that have 0 probability in the Bayesian Network (constructing $\bar{\phi}_B$), and finally using these sets of worlds to determine if a world exists that meets the requirements. We have shown that constructing ϕ_K^C is exponential in the size of the input knowledge base, that constructing $\bar{\phi}_B$ is in the worst case exponential in the size of the Bayesian Network, and that determining if there exists a world in both sets is exponential in this case (it is polynomial in the size of the contexts which are in this case exponential).

This gives the result that the complexity of determining whether a given concept is totally satisfiable in some knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ is exponential in the size of the input knowledge base and the number of random variables (V) in the Bayesian Network. That is total concept satisfiability is in the complexity class $O(2^{||\mathcal{K}||+|V|})$.

The algorithm we have provided for total concept satisfiability is a more general algorithm than is required for consistency. In the next section we show how we adapt this algorithm for consistency checking.

5.3 Consistency

Encoding some knowledge about a domain is only really useful if there exists some instance (model) of said domain. Otherwise an impossible domain has been encoded which we cannot reason about or apply. As such a reasoning service to check that a knowledge base is capable of modeling some domain of knowledge (that a knowledge base has a model) is critical. We provide a solution to this problem by first introducing the formal definition of consistency for $\mathcal{BALC}$ (we need to know what it means to have a model before we can determine if a knowledge base has one) and then present a consistency testing procedure for $\mathcal{BALC}$.

In defining the reasoning problems and Description Logic components of $\mathcal{BALC}$ we have tried to keep definitions consistent with that of classical Description Logics. Therefore it makes sense to define consistency for $\mathcal{BALC}$ analogously to consistency in other Description Logics. That is we say that a $\mathcal{BALC}$ knowledge base is consistent if, and only if, it has a (probabilistic) model.

Definition 5.3.1 (Consistency). A $\mathcal{BALC}$ knowledge base $\mathcal{K}$ is consistent iff it has a probabilistic model. We will often write $\mathcal{K} \models \mathcal{P}$ when a probabilistic interpretation $\mathcal{P}$ is a model of $\mathcal{K}$.

Recall that in $\mathcal{EL}$ (and in $\mathcal{BEL}$) the consistency problem is trivial since all KBs are consistent. This is not the case in $\mathcal{BALC}$ as can be seen with the following example. Consider the knowledge base with $P(\alpha) > 0$ in the BN and the TBox:

$$\mathcal{T} = \{(A \sqsubseteq B)^\top, (A \sqsubseteq \neg B)^\top\}$$

and the ABox:

$$\mathcal{A} = \{(a \in A)^\alpha\}$$

It is easy to see that this KB is inconsistent. In order for this KB to be consistent there would have to exist a probabilistic model in which the concept A is both empty in all worlds with positive probability **and** non-empty in all worlds that entail α . Since this is not possible the KB must be inconsistent (as it cannot have a probabilistic model).

Having now formally defined what is meant by consistency in $\mathcal{BALC}$ we can now check the consistency of a knowledge base by either attempting to find a probabilistic model for the KB or by showing that no such model can exist. Obviously if we follow the initial approach and attempt to find a probabilistic model the construction of the model must be guaranteed to find a model, if one exists, in order to be considered complete. For the latter method, when attempting to show that no model exists, the process should only fail (to show that no model exists) if the knowledge base is consistent. The algorithm we present in this section technically follows the second approach and attempts to show that no model exists for a knowledge base.

Recall that in our definition of a V -interpretation we require that the domain Δ^V be non-empty. This leads us to the obvious consequence that a $\mathcal{BALC}$ knowledge base is only consistent if $\top$ is totally satisfiable (if $\top$ is not totally satisfiable then there is some V -interpretation that would be empty). Thus if we show that $\top$ is not totally satisfiable we have shown that there can exist no probabilistic model for the knowledge base (conversely if $\top$ is totally satisfiable then the KB has a model). We state and then prove this formally in the following theorem:

Theorem 5.3.2. A $\mathcal{BALC}$ knowledge base $\mathcal{K}$ is *consistent* if, and only if, $\top$ is totally satisfiable in $\mathcal{K}$.

Proof. ($\Rightarrow$) If $\top$ is totally satisfiable in $\mathcal{K}$ then by Definition 5.2.1 we have that there exists a probabilistic model of $\mathcal{K}$ with $\top$ non-empty in all V -interpretations. Since we have a probabilistic model for $\mathcal{K}$ it is consistent.

($\Leftarrow$) If $\mathcal{K}$ is consistent then we have at least one probabilistic model $\mathcal{P}$ for $\mathcal{K}$. Then by definition all V -interpretations in $\mathcal{P}$ have Δ^V non-empty. Therefore we have that $\top$ is non-empty in all V -interpretations in $\mathcal{P}$. Therefore $\top$ is totally satisfiable in $\mathcal{K}$. $\square$

This theorem shows that the $\mathcal{BALC}$ consistency problem can be reduced to an instance of the total concept satisfiability problem. In fact it shows that consistency is just a special case of the total concept satisfiability problem (the satisfiability of the $\top$ concept). This is useful as it allows the reuse of an existing algorithm. This in turn shows that the two problems are in the same complexity class leading to the following lemma:

Lemma 5.3.3 (Complexity of consistency). Checking the consistency of a $\mathcal{BALC}$ knowledge base is in the complexity class $O(2^{||\mathcal{K}||})$.

With this we have concluded the section on consistency for $\mathcal{BALC}$. We are now able to represent some domain using the syntax of $\mathcal{BALC}$ and can now check that our representation is valid in terms of the semantics of $\mathcal{BALC}$ (by showing that there exists some interpretation that is modeled by the knowledge base). We now move on to the first inference problem in $\mathcal{BALC}$: checking whether one concept subsumes another in some context.

5.4 Subsumption

We introduced subsumption previously in Section 2.3.2 where we said a concept C is subsumed by a concept D in some knowledge base if, and only if, all individuals that are instances of C are also necessarily instances of D . We now adapt the definition of subsumption to take contexts into account by saying that a concept C subsumes a concept D in context κ if, and only if, in all worlds where κ is satisfied C is necessarily subsumed by D . We next provide an example to illustrate and contrast contextual subsumption with classical subsumption.

In classical Description Logics we can only express, or infer, that a subsumption holds under all conditions. We used the example of all textbooks being instances of non-fiction books as a practical example of subsumption. However, in $\mathcal{BALC}$ we are able to express that a subsumption holds only under certain conditions. We do this through the use of contexts to indicate that a given subsumption only holds under conditions that satisfy the context. For example consider a situation where someone who is walking observes a cat.

Clearly if the person seeing this cat is walking in a suburb its probable that this cat is a pet (for the sake of the example we will assume this is always the case). We would represent this fact in $\mathcal{BALC}$ syntax as $(Cat \sqsubseteq Pet)^{suburb}$. Which is the statement that says that in a suburban area all cats are pets. However what if this person is currently on a wildlife safari? Then its possible that the cat in question is actually a lion rather than a house cat. Clearly since a lion is not a pet then the contextual subsumption $(Cat \sqsubseteq Pet)^{savannah}$ should not be present, or entailed, by our knowledge base. But our knowledge base could still support the information that suburban cats are pets! This is something that is not possible under classical semantics (as we can only claim that all cats are always pets). Before we move on

to show how subsumptions are proved in $\mathcal{BALC}$ we must first provide a formal definition. We provide this definition, adapted from $\mathcal{BEL}$ [8], below:

Definition 5.4.1 (Contextual subsumption). Given $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ a $\mathcal{BALC}$ knowledge base, C, D possibly complex concepts, and κ a context. We say that C is contextually subsumed by D in context κ w.r.t. $\mathcal{K}$, denoted as $\mathcal{K} \models (C \sqsubseteq D)^\kappa$, if every probabilistic model of $\mathcal{K}$ is a probabilistic model of $(C \sqsubseteq D)^\kappa$.

While a binary definition of subsumption (either a concept is subsumed by another concept or it is not) is sufficient for classical Description Logics it is a bit lacking for probabilistic Description Logics. If we were to use only the definition of contextual subsumption we would only be able to reason about subsumptions that are “certain” while the entire purpose of a probabilistic DL is to reason about things which entail a degree of uncertainty.

Once we have defined the probability of a subsumption we will be able to work out how likely it is that a particular subsumption holds. For example we could calculate how likely it is that a cat is someones pet (which would intuitively be the sum of all worlds where all cats are pets). In fact, We will go further than just working out things like how likely it is that a cat is someones pet and allow for conditional subsumption queries. For example we will provide formalisms that allow queries such as determining the likelihood of a cat being a pet based on what we know about the observers location.

Intuitively we would want a probability of 1 for a subsumption to mean that one concept necessarily subsumes the other while a probability of $0 < p < 1$ would mean that C is subsumed by D in some worlds but not in others. We formalize the probability of a subsumption in such a way that this intuition holds.

Definition 5.4.2 (Probability of a Subsumption). We define the probability of a subsumption in a particular probabilistic model $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of some knowledge base $\mathcal{K}$ as

$$P_{\mathcal{P}}((C \sqsubseteq D)^\kappa) = \sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models (C \sqsubseteq D)^\kappa} P_{\mathcal{J}}(\mathcal{V})$$

That is the probability of a subsumption in a specific model is the sum of the probabilities of the worlds in which C is subsumed by D in context κ (this trivially includes all worlds where κ does not hold).

We define the probability of $(C \sqsubseteq D)^\kappa$ with respect to a knowledge base $\mathcal{K}$ as:

$$P_{\mathcal{K}}((C \sqsubseteq D)^\kappa) = \inf_{\mathcal{P} \models \mathcal{K}} P_{\mathcal{P}}((C \sqsubseteq D)^\kappa)$$

In the case where $\mathcal{K}$ is inconsistent we define the probability of all subsumptions as 1 to ensure our definition is consistent with general probability theory ($\inf(\emptyset) = \infty$ in general).

Finally we say that C is *positively* subsumed by D in κ if $P((C \sqsubseteq D)^\kappa) > 0$, *p-subsumed* if $P((C \sqsubseteq D)^\kappa) \geq p$, *exactly* subsumed if $P((C \sqsubseteq D)^\kappa) = p$, and *certainly* subsumed if $P((C \sqsubseteq D)^\kappa) = 1$.

It should be noted that there is a difference between the probability of the subsumption $(C \sqsubseteq D)^\kappa$ (that is $P_{\mathcal{K}}((C \sqsubseteq D)^\kappa)$) and the probability of the subsumption $(C \sqsubseteq D)^\top$ holding in context κ (that is $P((C \sqsubseteq D)^\top | \kappa)$). The most obvious way to see this is to remember that the former holds in all contexts that are not consistent with κ while the latter is a conditional probability query asking how likely it is that the subsumption holds given that specific context holds. Or in the case of our cat example it is the difference between the probability of all city cats being pets

$P_{\mathcal{K}}((Cat \sqsubseteq Pet)^{city})$ and the probability of all cats being pets given that we are in the city $P((Cat \sqsubseteq Pet)^{\top} | city)$.

Having now defined both what is meant by subsumption and the probability of a subsumption in $\mathcal{BALC}$ we next show how these concepts are related. In fact we show that they are related in the intuitive way that if C is subsumed by D with probability 1 then C is necessarily subsumed by D in all probabilistic models of the knowledge base.

Theorem 5.4.3. Given a knowledge base $\mathcal{K}$, concepts C and D , and a context κ then we have that:

$$\mathcal{K} \models (C \sqsubseteq D)^{\phi} \text{ iff } P_{\mathcal{K}}((C \sqsubseteq D)^{\phi}) = 1$$

That is C is subsumed by D in concept κ if, and only if, the probability of the subsumption is 1.

Proof. $P_{\mathcal{K}}((C \sqsubseteq D)^{\phi}) = 1$ iff for all probabilistic models $\mathcal{P}$ of $\mathcal{K}$ we have that $P_{\mathcal{P}}((C \sqsubseteq D)^{\phi}) = 1$. Which holds iff in all probabilistic models $\mathcal{P}$ of $\mathcal{K}$ we have that for all $\mathcal{V} \in \mathcal{J}$, we have that $\mathcal{V} \models (C \sqsubseteq D)^{\phi}$, iff $\mathcal{P}$ of $\mathcal{K}$ we have that $\mathcal{P} \models (C \sqsubseteq D)^{\phi}$, iff $\mathcal{K} \models (C \sqsubseteq D)^{\phi}$. $\square$

We have mentioned that there is a difference between the probability of a contextual subsumption and a conditional probabilistic subsumption (all city cats being pets and all cats being pets in the city). While potentially informative this statement by itself provides no insight into what exactly a conditional probabilistic subsumption actually is. We address this by providing a formal definition for the probability of a conditional subsumption.

Definition 5.4.4 (Conditional probability of a Subsumption). We define the conditional probability of a subsumption in a particular probabilistic model $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ as

$$P_{\mathcal{P}}((C \sqsubseteq D)^{\phi} | \psi) = \frac{\sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models \psi, \mathcal{V} \models (C \sqsubseteq D)^{\phi}} P_{\mathcal{J}}(\mathcal{V})}{P(\psi)}$$

That is the sum of all worlds in which the contextual subsumption holds **and** the conditional context is also satisfied normalized by the probability of the conditional context.

We define the probability of the conditional subsumption in a knowledge base $\mathcal{K}$ as:

$$P_{\mathcal{K}}((C \sqsubseteq D)^{\phi} | \psi) = \inf_{\mathcal{P} \models \mathcal{K}} P_{\mathcal{P}}((C \sqsubseteq D)^{\phi} | \psi)$$

This definition is intentionally quite similar to the Bayes Conditioning (see Theorem 3.0.6) which expresses the probability of a conditional probability in terms of non-conditional probabilities. In general probability theory this is useful exactly because Bayes Conditioning is used to reduce conditional probability queries to a combination of regular probability queries. If we had a similar result for conditional subsumption we would only need a single algorithm for finding the probability of a subsumption. Fortunately it turns out that this is exactly the case!

We next present the useful result that the probability of a conditional contextual subsumption can be reduced to a contextual subsumption and a few probability calculations in the Bayesian Network. This result allows us to develop only a single algorithm for subsumption problems which can then be applied to both conditional and non conditional subsumption queries.

Theorem 5.4.5.

$$P_{\mathcal{K}}((C \sqsubseteq D)^{\phi|\psi}) = \frac{P_{\mathcal{K}}((C \sqsubseteq D)^{\phi \wedge \psi}) + P(\phi) - P(\phi \wedge \psi)}{P(\psi)}$$

Proof.

$$\begin{aligned}
P_{\mathcal{K}}((C \sqsubseteq D)^{\phi|\psi}) &= \inf_{\mathcal{P} \models \mathcal{K}} \frac{\sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \psi, \mathcal{V} \models (C \sqsubseteq D)^{\phi}} P_{\mathcal{J}}(\mathcal{V})}{P(\psi)} \\
&= \frac{\inf_{\mathcal{P} \models \mathcal{K}} \sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \psi, \mathcal{V} \models (C \sqsubseteq D)^{\phi}} P_{\mathcal{J}}(\mathcal{V})}{P(\psi)} \\
&= \frac{\inf_{\mathcal{P} \models \mathcal{K}} (\sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \psi, v^{\mathcal{V}} \models \phi, C^{\mathcal{V}} \subseteq D^{\mathcal{V}}} P_{\mathcal{J}}(\mathcal{V}) + \sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \psi, v^{\mathcal{V}} \not\models \phi} P_{\mathcal{J}}(\mathcal{V}))}{P(\psi)} \\
&= \frac{\inf_{\mathcal{P} \models \mathcal{K}} (\sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \psi, v^{\mathcal{V}} \models \phi, C^{\mathcal{V}} \subseteq D^{\mathcal{V}}} P_{\mathcal{J}}(\mathcal{V}) + \sum_{\omega \models \psi, \omega \not\models \phi} P(\omega))}{P(\psi)} \\
&= \frac{\inf_{\mathcal{P} \models \mathcal{K}} (\sum_{\mathcal{V} \in \mathcal{J}, v^{\mathcal{V}} \models \psi, v^{\mathcal{V}} \models \phi, C^{\mathcal{V}} \subseteq D^{\mathcal{V}}} P_{\mathcal{J}}(\mathcal{V})) + \sum_{\omega \models \psi, \omega \not\models \phi} P(\omega)}{P(\psi)} \\
&= \frac{P_{\mathcal{K}}((C \sqsubseteq D)^{\phi \wedge \psi}) + \sum_{\omega \models \phi} P(\omega) - \sum_{\omega \models \phi, \omega \models \psi} P(\omega)}{P(\psi)} \\
&= \frac{P_{\mathcal{K}}((C \sqsubseteq D)^{\phi \wedge \psi}) + P(\psi) - P(\psi \wedge \phi)}{P(\psi)}
\end{aligned}$$

□

We have now proved all general results about probabilistic subsumption in $\mathcal{BALC}$ that we will require before moving on to specific instances of the subsumption problem. We will next present our findings on the positive subsumption, exact subsumption, and p -subsumption problems. We start by examining the positive subsumption problem in more detail in the next section followed by the exact subsumption and then the p -subsumption problem.

5.4.1 Positive Subsumption

We have previously described the positive subsumption problem as determining whether a contextual subsumption holds with non-zero probability in a given knowledge base. In our cat example this would be the problem of determining whether it is possible for cats to be pets in some location.

Intuitively the formal definition from the previous section can be rephrased as if C is positively subsumed by D in context κ for some knowledge base $\mathcal{K}$ then for all probabilistic models of $\mathcal{K}$ we know that the subsumption $(C \sqsubseteq D)^{\kappa}$ has probability greater than 0. Since the subsumption has probability greater than 0 in all probabilistic models of $\mathcal{K}$ we know that in each probabilistic model of $\mathcal{K}$ there is some

V -interpretation (that has probability greater than 0) such that $(C \sqsubseteq D)^\kappa$ holds. This leads us to the theorem stated below.

Theorem 5.4.6. Given a $\mathcal{BALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ and a subsumption $(C \sqsubseteq D)^\kappa$ then $P_{\mathcal{K}}((C \sqsubseteq D)^\kappa) > 0$ iff for all probabilistic models there exists a Bayesian world ω such with $P(\omega) > 0$ such that either $\omega \models \kappa$ and $(\mathcal{T}_\omega, \mathcal{A}_\omega) \models C \sqsubseteq D$ or $\omega \not\models \kappa$.

While the first part of this theorem seems obvious (we need some world that has probability greater than 0) the second part is a little more intricate. Obviously if we have a world with probability greater than 0 in which C is subsumed by D in all V -interpretations that match the world our subsumption has probability greater than 0. However, we also have that in all worlds which are not consistent with κ our subsumption trivially holds. Therefore if there is some world, with probability greater than 0, in which κ does not hold then the subsumption will also have probability greater than 0. We provide a more formal proof that follows this outline below:

Proof. ($\Leftarrow$) If in all probabilistic models of $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ there exists a Bayesian world ω such that $\omega \models \kappa$ with $(\mathcal{T}_\omega, \mathcal{A}_\omega) \models C \sqsubseteq D$ and $P(\omega) > 0$ then we have that $\inf_{\mathcal{P} \models \mathcal{K}} P((C \sqsubseteq D)^\kappa) > 0$. Therefore C is positively subsumed by D in context κ with respect to knowledge base $\mathcal{K}$.

If in all probabilistic models $\mathcal{P}$ of $\mathcal{K}$ we have a Bayesian world ω such that $P(\omega) > 0$ and $\omega \not\models \kappa$ then in all $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ we have a $\mathcal{V} \in \mathcal{J}$ such that $v^{\mathcal{V}} = \omega$ that trivially satisfies $(C \sqsubseteq D)^\kappa$ so $\inf_{\mathcal{P} \models \mathcal{K}} P_{\mathcal{P}}((C \sqsubseteq D)^\kappa) > 0$.

($\Rightarrow$) If $P_{\mathcal{K}}((C \sqsubseteq D)^\kappa) > 0$ then in all probabilistic models $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of $\mathcal{K}$ there exists a $\mathcal{V} \in \mathcal{J}$ such that $\mathcal{V} \models (C \sqsubseteq D)^\kappa$ and $P_{\mathcal{J}}(\mathcal{V}) > 0$. Now note that across all probabilistic models there must be a specific $v^{\mathcal{V}}$ such that the previous condition holds for its associated V -interpretation. If this was not the case a new probabilistic model could be constructed such that C is not positively subsumed by D in κ . Since this $v^{\mathcal{V}}$ corresponds to some ω we have found a ω with $P(\omega) > 0$ such that either $\omega \models \kappa$ and $(\mathcal{T}_\omega, \mathcal{A}_\omega) \models C \sqsubseteq D$ or $\omega \not\models \kappa$. $\square$

We have now reduced the positive subsumption problem to finding a Bayesian world in which the given subsumption holds with probability greater than 0 (either through not matching the context or by holding classically). However we have still not provided an algorithm that does this. Fortunately we can use the total concept satisfiability algorithm to solve part of this problem and regular inference in the Bayesian Network for the other. We show how to do this below:

Theorem 5.4.7. Given a contextual subsumption $(C \sqsubseteq D)^\kappa$ and a consistent knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ then $P((C \sqsubseteq D)^\kappa) > 0$ iff $\mathcal{K}' = (\mathcal{T}, \mathcal{A} \cup \{C(x)^\kappa, \neg D(x)^\kappa\}, \mathcal{B})$ is inconsistent or $P(\kappa) \neq 1$.

Intuitively this works because if we make a consistent knowledge base inconsistent by asserting that the subsumption does not hold then clearly it holds in the original knowledge base. Alternatively if $P(\kappa) < 1$ in the Bayesian Network then there exists some world ω with probability greater than 0 in which $(C \sqsubseteq D)^\kappa$ trivially holds (so the subsumption must have non-zero probability). Furthermore the requirement that the input knowledge base be consistent is reasonable as all inconsistent knowledge bases entail all subsumptions with probability 1 (by definition). We provide a formal proof for this theorem below:

Proof. ($\Leftarrow$) If $\mathcal{K}'$ is inconsistent then there exists some ω such that $\mathcal{K}'$ has $\top$ totally unsatisfiable and $P(\omega) > 0$. Since $\mathcal{K}$ is consistent we have now found a world such that $(\mathcal{T}_\omega, \mathcal{A}_\omega) \models C \sqsubseteq D$ with $P(\omega) > 0$. Therefore $P((C \sqsubseteq D)^\kappa) > 0$.

If $P(\kappa) \neq 1$ then there exists some Bayesian world ω such that $\omega \not\models \kappa$ with $P(\omega) > 0$. Therefore $P_{\mathcal{K}}((C \sqsubseteq D)^\kappa) > 0$.

($\Rightarrow$) If $(C \sqsubseteq D)^\kappa > 0$ then there exists a Bayesian world ω such that $P(\omega) > 0$ and either $\omega \models \kappa$ and $(\mathcal{T}_\omega, \mathcal{A}_\omega) \models C \sqsubseteq D$ so $(\mathcal{T}, (\mathcal{A} \cup \{C(x)^\kappa, \neg D(x)^\kappa\})_\omega)$ is not satisfiable so $\mathcal{K}'$ is inconsistent; or $\omega \not\models \kappa$ so $P(\kappa) \neq 1$. $\square$

We have now proved that the positive subsumption problem can be reduced to an instance of the $\mathcal{BALC}$ consistency problem as well as a single probability calculation in the Bayesian Network. Complexity wise the worst case scenario will occur when we have to do both the probability calculation in the Bayesian Network as well as the consistency check which would give us a worst case complexity of doing both. This gives us the following lemma:

Lemma 5.4.8. Given a contextual subsumption $(C \sqsubseteq D)^\kappa$ and a consistent knowledge base $\mathcal{K}$ we can determine if the subsumption has probability greater than 0 in $O(2^{||\mathcal{K}||} + \exp(V))$ time.

Note that in the best case we only have to do one of the operations. Since there exist multiple efficient practical reasoners for Bayesian Networks it would probably be more efficient to do the Bayesian inference first when developing a reasoner for $\mathcal{BALC}$.

We are now able to determine whether its possible that city cats are pets, or more interestingly whether its possible for Savannah cats to be pets. However, we still cannot determine whether its possible that the cat we saw while walking in the city was someones pet. Fortunately, as we have already shown that there exists a reduction from conditional subsumption to subsumption, we can immediately solve this problem.

Theorem 5.4.9. Given some consistent knowledge base $\mathcal{K}$, concepts C, D , and primitive contexts ϕ, ψ we have the following:

$$P_{\mathcal{K}}((C \sqsubseteq D)^\phi | \psi) > 0 \text{ iff } P_{\mathcal{K}}((C \sqsubseteq D)^{\phi \wedge \psi}) \text{ or } P(\psi) - P(\psi \wedge \phi) > 0$$

Proof.

$$\begin{aligned} P_{\mathcal{K}}((C \sqsubseteq D)^\phi | \psi) &= \frac{P_{\mathcal{K}}((C \sqsubseteq D)^{\phi \wedge \psi}) + P(\psi) - P(\psi \wedge \phi)}{P(\psi)} \\ &\leq P_{\mathcal{K}}((C \sqsubseteq D)^{\phi \wedge \psi}) + P(\psi) - P(\psi \wedge \phi) \end{aligned}$$

and it is trivially true that $P(\psi) \geq P(\psi \wedge \phi)$. Therefore since we have that $P_{\mathcal{K}}((C \sqsubseteq D)^{\phi \wedge \psi}) \geq 0$ and $P(\psi) - P(\psi \wedge \phi) \geq 0$ we must have that $P_{\mathcal{K}}((C \sqsubseteq D)^\phi | \psi) \geq 0$ if either of the previous is greater than 0. $\square$

We have now shown that we can do a conditional probabilistic subsumption query by doing at most one probabilistic subsumption query and two Bayesian Network queries. This puts this problem in the same complexity class as probabilistic subsumption giving us the following lemma:

Lemma 5.4.10. Given some consistent knowledge base $\mathcal{K}$, concepts C, D , and primitive contexts ϕ, ψ we can determine if $P_{\mathcal{K}}((C \sqsubseteq D)^\phi | \psi) > 0$ in $O(2^{||\mathcal{K}||} + \exp(V))$ time.

Having now provided an algorithm for solving both the positive subsumption and conditional positive subsumption problem we conclude this subsection of subsumption. In the next subsection we present an algorithm for calculating the exact probability of a subsumption in a *BALC* knowledge base.

5.4.2 Exact Subsumption

In this section we will present the exact subsumption problem in more detail. We start by providing an intuitive explanation of the problem and then provide some results which allows us to use the tableaux algorithm again in order to calculate the exact probability contextual subsumption.

We mentioned previously that exact subsumption determines the precise probability of a given contextual subsumption in a knowledge base. Therefore, unlike positive subsumption which is a decision problem, exact subsumption does not have a binary answer. Instead this problem takes in a knowledge base and two concepts and attempts to determine how likely it is for the one concept to subsume the other in a certain context. Intuitively this is equal to the sum of the probabilities of all worlds in which the subsumption is necessary. We know that it only uses the necessary worlds because if the subsumption is not necessary then the infimum in the definition will prevent it from holding in the model that minimizes the probability.

Clearly we need some way to identify which probabilistic worlds lead to the knowledge base entailing some contextual subsumption. Recall that a contextual subsumption holds in two cases: firstly when both its context is satisfied and the subsumption holds classically, or secondly when its context does not hold. So clearly calculating the exact probability of a subsumption has two parts: calculating the probability of the context not holding, and working out the probability of the context and subsumption holding. The first portion of this is clearly just a probability calculation in the Bayesian Network (the probability of ϕ not holding is equal to $1 - P(\phi)$). The latter portion is more complicated as we need to determine which contexts entail the subsumption.

Luckily we already have an algorithm that traces which axioms (and hence contexts) lead to some knowledge base being inconsistent. We can assume that we start with a consistent knowledge base. This is reasonable as all inconsistent knowledge bases entail all subsumptions with probability 1. Now if we want to find the worlds that entail some subsumption $(C \sqsubseteq D)^\phi$ we can add in the statement that C is actually not subsumed by D in context ϕ and then determine if this information clashes with anything already in the knowledge base. If it does not then we have found a model in which the contextual subsumption does not hold and so have shown that the subsumption has probability 0. If we do find clashes then we have found contexts describing the worlds which must entail the subsumption in all models of this knowledge base. This line of reasoning leads us to the following theorem:

Theorem 5.4.11. For a consistent knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$, a contextual subsumption $(C \sqsubseteq D)^\phi$, and the extended KB $\mathcal{K}' = (\mathcal{T}, \mathcal{A} \cup \{C(x)^\phi, \neg D(x)^\phi\}, \mathcal{B})$ we have the following

$$P_{\mathcal{K}}((C \sqsubseteq D)^\phi) = \sum_{\omega \models \phi_{\mathcal{K}'}} P(\omega) + 1 - P(\phi)$$

Proof.

$$\begin{aligned}
 P_{\mathcal{K}}((C \sqsubseteq D)^\phi) &= \inf_{\mathcal{P} \models \mathcal{K}} P_{\mathcal{P}}(C \sqsubseteq D^\phi) \\
 &= \inf_{\mathcal{P} \models \mathcal{K}} \left(\sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models (C \sqsubseteq D)^\phi} P_{\mathcal{J}}(\mathcal{V}) \right) \\
 &= \inf_{\mathcal{P} \models \mathcal{K}} \left(\sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models \phi, C^\mathcal{V} \subseteq D^\mathcal{V}} P_{\mathcal{J}}(\mathcal{V}) + \sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \not\models \phi} P_{\mathcal{J}}(\mathcal{V}) \right)
 \end{aligned}$$

Since in all probabilistic models $\mathcal{P}$ of $\mathcal{K}$ we have that $\sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models \omega} P_{\mathcal{J}}(\mathcal{V}) = P(\omega)$ we have that

$$\begin{aligned}
 P_{\mathcal{K}}((C \sqsubseteq D)^\phi) &= \inf_{\mathcal{P} \models \mathcal{K}} \left(\sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models \phi, C^\mathcal{V} \subseteq D^\mathcal{V}} P_{\mathcal{J}}(\mathcal{V}) + \sum_{\omega \not\models \phi} P(\omega) \right) \\
 &= \inf_{\mathcal{P} \models \mathcal{K}} \left(\sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models \phi, C^\mathcal{V} \subseteq D^\mathcal{V}} P_{\mathcal{J}}(\mathcal{V}) \right) + \sum_{\omega \not\models \phi} P(\omega)
 \end{aligned}$$

For all ω such that $\omega \models \phi_{\mathcal{K}'}$ we have that $(\mathcal{T}_\omega, (\mathcal{A} \cup \{C(x)^\phi, \neg D(x)^\phi\})_\omega)$ is inconsistent. Since $\mathcal{K}$ is consistent then either $(\mathcal{T}_\omega, \mathcal{A}_\omega)$ is consistent so $(\mathcal{T}_\omega, \mathcal{A}_\omega) \models C \sqsubseteq D$ or $P(\omega) = 0$. So $\phi_{\mathcal{K}'}$ encodes all worlds that must entail $C \sqsubseteq D$. Since in the infimum model $C^\mathcal{V} \subseteq D^\mathcal{V}$ will only hold in Bayesian worlds such that $(\mathcal{T}_\omega, \mathcal{A}_\omega) \models C \sqsubseteq D$ we have that

$$\inf_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models \phi, C^\mathcal{V} \subseteq D^\mathcal{V}} P_{\mathcal{J}}(\mathcal{V}) = \sum_{\omega \models \phi_{\mathcal{K}'}} P(\omega)$$

Therefore

$$\begin{aligned}
 P_{\mathcal{K}}((C \sqsubseteq D)^\phi) &= \sum_{\omega \models \phi_{\mathcal{K}'}} P(\omega) + \sum_{\omega \not\models \phi} P(\omega) \\
 &= \sum_{\omega \models \phi_{\mathcal{K}'}} P(\omega) + 1 - P(\phi)
 \end{aligned}$$

□

We have now shown that $\phi_{\mathcal{K}'}$ encodes all worlds that lead to our given subsumption being entailed. Now we just need to sum over the probabilities of all worlds that entail $\phi_{\mathcal{K}'}$. Unfortunately this is not as trivial as it may seem. Some of the contexts in $\phi_{\mathcal{K}'}$ may be entailed by the same worlds. That is there may be some worlds which model more than one context in $\phi_{\mathcal{K}'}$. This means that we cannot simply work out the probability of each context (partial valuation) in $\phi_{\mathcal{K}'}$ and sum them together since we might count some worlds more than once.

The obvious way of addressing this problem is to iterate over all worlds in the Bayesian Network and then for each world ω to check if $\omega \models \phi_{\mathcal{K}'}$. If it does we then calculate its probability and add it to some running total. Once this process terminates we then have to calculate $1 - P(\phi)$ and add it to the running total to get the exact probability of the subsumption.

While this is not an elegant solution it does not change the complexity class of the algorithm. The algorithm for constructing $\phi_{\mathcal{K}'}$ is already exponential in the size of the input knowledge base and Bayesian Network inference is inherently exponential. Therefore following this approach gives us the following complexity result:

Theorem 5.4.12. Given a knowledge base $\mathcal{K}$ we can calculate the probability of a contextual subsumption in $O(\exp(|\mathcal{K}| + |V|))$ time.

Proof. When following this approach we have to iterate $\exp(|V|)$ times (once for each probabilistic world) and do one $2^{|\mathcal{K}|}$ operation each time (entailment check). We also have the initial $O(2^{|\mathcal{K}|})$ cost of constructing $\phi_{\mathcal{K}'}$ to take into account. This gives us an upper bound of $O(\exp(|V|) \cdot 2^{|\mathcal{K}|} + 2^{|\mathcal{K}|})$ which is equivalent to $O(\exp(|\mathcal{K}| + |V|))$. $\square$

While it is true that we cannot expect a better result than an exponential upper bound our current approach is quite inefficient if we have a small number of fairly specific contexts. For example if we have a set of contexts such that they only encode a small portion of the available worlds we will waste a lot of time iterating over unnecessary worlds. In this case it would be more efficient to use the contexts in $\phi_{\mathcal{K}'}$ to guide the iteration over the worlds. We could do this by picking one of the primitive contexts in $\phi_{\mathcal{K}'}$ and then iterating over its unfixed variables to create worlds and adding their probabilities to some running total. This way we know that all the worlds we create entail $\phi_{\mathcal{K}'}$ and we can skip the entailment check. However we do need to store which worlds we have already seen to avoid counting any worlds twice. We then choose a new primitive contexts and keep doing this until we have seen all worlds or we run out of primitive contexts in $\phi_{\mathcal{K}'}$. At this point we calculate $1 - P(\phi)$ and add it to the running total to get the final answer.

While there are specific cases where this approach would be much faster than iterating over all worlds it is still in the exact same complexity class as the previous solution ($O(\exp(|\mathcal{K}| + |V|))$).

In concluding the presentation of our work in calculating exact probabilities of subsumptions we would like to point the reader back to Theorem 5.4.5. Since we have shown that we can reduce conditional subsumptions to probabilistic subsumption (plus a few Bayesian inferences) we already have a procedure for finding the exact probability of a conditional subsumption. Furthermore since Bayesian inference is $O(\exp(|V|))$ in Bayesian Networks we also have that calculating the probability of a conditional subsumption falls in the same complexity class as calculating the probability of a subsumption.

We have now provided algorithms that solve both the positive and exact subsumption problems. This leaves us only with the p -subsumption problem, which we briefly discuss in the next section.

5.4.3 p -Subsumption

In the previous two sections we have provided reasoning procedures that can either determine if some subsumption has positive probability or find the exact probability of the subsumption. We now move onto the p -subsumption problem. In contrast to the previous problems we are now instead interested in knowing if some contextual subsumption has probability of at least p .

Intuitively the p -subsumption problem falls somewhere in between positive and exact subsumption problem. We clearly need to do at least a partial calculation of the probability but do not necessarily have to do a full calculation. This means we only need to find worlds in which the contextual subsumption holds until we have a probability equal or greater than p . However, since positive and exact subsumption fall in the same complexity class we should not expect this problem to be more efficient than the previous problems. At least not until we can show that there exists a more efficient procedure for positive subsumption.

As such instead of developing an entirely new algorithm to solve this problem we instead adapt the exact subsumption algorithm. Obviously it is a good idea to calculate the $1 - P(\phi)$ portion of the probability first. Since if this value is greater or equal to p we can avoid doing any reasoning in the knowledge base (we do not have to construct $\phi_{\mathcal{K}'}$ or iterate over worlds entailed by $\phi_{\mathcal{K}'}$). However, if this probability is not equal, or greater, than p we start iterating over probabilistic worlds ω such that $\omega \models \phi_{\mathcal{K}'}$, and simply add a condition that checks if the running total equals or exceeds p . If this should happen the algorithm can terminate and return that the subsumption holds with probability at least p .

In the worst case this algorithm is identical to exact subsumption. This occurs if the probability of the subsumption is equal or less than p . In this case we would have to do all Bayesian inferences (one for regular p -subsumption and more for conditional p -subsumption) as well as iterate over all worlds entailed by $\phi_{\mathcal{K}'}$. This gives us the following lemma:

Lemma 5.4.13. The complexity of determining whether a (conditional) contextual subsumption holds with probability p in a knowledge base $\mathcal{K}$ is in the worst case $O(\exp(|\mathcal{K}| + |V|))$ time.

This concludes our work on probabilistic subsumption for $\mathcal{BALC}$. While working on the subsumption problem in $\mathcal{BALC}$ we have seen both a decision problem and probability calculation version for each subproblem. We now extend the concept of calculating the probability of a reasoning problem to concept satisfiability.

5.5 Partial Concept Satisfiability

Now that we have studied the subsumption problem in $\mathcal{BALC}$ we have seen that we can define not only an decision version of subsumption but also the probability of a subsumption. We now address the question if it is possible to define something similar for satisfiability. It turns out that this is entirely possible leading us to partial (or probabilistic) concept satisfiability.

Partial concept satisfiability is a weaker form of the satisfiability problem than total concept satisfiability. Total concept SAT deals with determining whether a probabilistic model exists for a knowledge base such that a concept is satisfiable in **all** of its V -interpretations. While this is an interesting problem, and a more general problem than $\mathcal{BALC}$ consistency as we have shown, it does not completely capture the probabilistic nature of $\mathcal{BALC}$. In particular total concept SAT does not address questions such as “how likely is it that a concept is satisfied in a knowledge base”. In order to answer this kind of question we introduce partial concept satisfiability.

Definition 5.5.1 (Partial Concept Satisfiability). A concept C is partially satisfiable with respect to a $\mathcal{BALC}$ Knowledge base $\mathcal{K}$ iff there exists a probabilistic model $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of $\mathcal{K}$ s.t. $\mathcal{P}$ has a V -interpretation $\mathcal{V} \in \mathcal{J}$, with $P_{\mathcal{J}}(\mathcal{V}) > 0$, that has at least one individual in C .

Intuitively a knowledge base having a concept that is partially satisfiable can be understood as there existing some model for the knowledge base such that it is possible for there to exist an individual in the concept (i.e. there is some V -interpretation that has an element in the concept). For example consider a lottery situation and in particular the concept $\exists won.Lottery$ (the set of winners of the lottery). The chance of winning a lottery is very low, however, it is not impossible. So we would probably expect probabilistic models that model this lottery to have no winners in many of

the possible worlds. Clearly we now have that the $\exists won.Lottery$ concept is not totally satisfiable. However we would expect it to be possible to win the lottery under some conditions (otherwise the lottery would be a fraudulent). So clearly we should expect $\exists won.Lottery$ to be partially satisfiable in any valid lottery! We also now provide a second more formal example. Consider again the example knowledge base that has $P(\alpha) > 0$ and consists of the following TBox:

$$\mathcal{T} = \{(A \sqsubseteq B)^\top, (A \sqsubseteq \neg B)^\alpha\}$$

Clearly this KB has that concept A is not totally satisfiable. However A is partially satisfiable if $P(\alpha) < 1$. If this is the case then there exists a world with probability greater than 0 where α does not hold. Therefore such a world can have A not empty. Next we show how to test a concept for partial satisfiability.

We draw inspiration from classical $\mathcal{ALC}$, in which many of the reasoning problems can be reduced to each other, and try to show that we can check if a concept is partially satisfiable using one of the other reason problems we have a solution for. If we think about what the definition of partial concept satisfiability requires from a knowledge base we see that it requires that in some model in some V -interpretation we have an element in concept C . Therefore in some model of our knowledge base $\mathcal{K}$ we have that C is not empty in all worlds, so clearly we have that $C \not\sqsubseteq \perp$ in all contexts! Therefore if we can show that the knowledge base does not entail $C \sqsubseteq \perp$, i.e. that $P((C \sqsubseteq \perp)^\top) \neq 1$, then we know that the concept C is partially satisfiable. This leads to the following theorem and the formal proof of this result.

Theorem 5.5.2. A concept C is partially satisfiable with respect to a $\mathcal{BALC}$ knowledge base $\mathcal{K}$ iff $\mathcal{K} \not\models C \sqsubseteq \perp$.

Proof. ($\Leftarrow$) If $\mathcal{K} \not\models C \sqsubseteq \perp$ then there exists a probabilistic model $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of $\mathcal{K}$ such that $\mathcal{P}$ is not a model of $(C \sqsubseteq \perp)^\top$. Which in turn means that there exists a $\mathcal{V} \in \mathcal{J}$ such that $\mathcal{V}$ is not a model of $(C \sqsubseteq \perp)^\top$. Since $\mathcal{V}$ is not a model of the subsumption we know that $v^\mathcal{V} \models \top$ and $C^\mathcal{V} \not\sqsubseteq \perp^\mathcal{V}$. Since all $P_{\mathcal{J}}(\mathcal{V}) > 0$ by definition we have a probabilistic model with a V -interpretation that has at least one individual in C so C is partially satisfiable.

($\Rightarrow$) If C is partially satisfiable in $\mathcal{K}$ then we have that there must exist a probabilistic model $\mathcal{P}$ with a V -interpretation $\mathcal{V}$ such that $\mathcal{V} \models C \sqsubseteq \perp$. Since we have a probabilistic model $\mathcal{P}$ s.t $\mathcal{P} \models C \sqsubseteq \perp$ we have that $\mathcal{K} \not\models C \sqsubseteq \perp$. $\square$

Now that we have a link between probabilistic subsumption and partial concept satisfiability we just need to show exactly how we reduce one problem to the other. We do this by using the fact that if a knowledge base has that the probability of $(C \sqsubseteq \perp)^\top$ is equal to 1 then there exist no model which has a V -interpretation where C is not empty. More formally we have that:

Theorem 5.5.3. The concept C is not partially satisfiable in knowledge base $\mathcal{K}$ iff $P((C \sqsubseteq \perp)^\top) = 1$.

Proof. ($\Leftarrow$) If $P((C \sqsubseteq \perp)^\top) = 1$ then in all probabilistic models $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of $\mathcal{K}$ we have that for all $\mathcal{V} \in \mathcal{J}$, $\mathcal{V} \models C \sqsubseteq \perp$. Therefore there is no V -interpretation with an individual in C so C cannot be partially satisfiable.

($\Rightarrow$) If C is not partially satisfiable then there exists no probabilistic model with a V -interpretation with $C^\mathcal{V} \neq \emptyset$. So in all V -interpretations of all probabilistic models we have that $\mathcal{V} \models C \sqsubseteq \perp$. Therefore $P((C \sqsubseteq \perp)^\top) = 1$ for all $\mathcal{P}$ so $\inf_{\mathcal{P} \models \mathcal{K}} (P((C \sqsubseteq \perp)^\top)) = 1$. $\square$

We have now formally shown how to reduce partial concept satisfiability to subsumption. Now when we are given some knowledge base $\mathcal{K}$ and a concept C we can determine if there exists some Bayesian world where this concept can have an element. In the context of our example we are now able to check whether it is possible to win the lottery.

In a certain sense this is a lot like the positive subsumption problem in that it tells us only if something is possible. While this is useful it is not as informative as we might like. Knowing that we can win the lottery and knowing how likely it is for someone to win the lottery are very different things after all. Ideally we would like to know exactly how likely it is for a concept to be satisfiable. However in order to do this we must first define what is meant by the probability of a concept being satisfiable.

We define the probability of partial satisfiability in a similar way to the probability of a subsumption. That is we first define the probability of partial satisfiability for a concept C in a probabilistic interpretation and then use this to define it in the context of a knowledge base.

Definition 5.5.4 (Probability of partial satisfiability). Given a concept C and a knowledge base $\mathcal{K}$ we define the probability of a C being partially satisfiable in a probabilistic model $\mathcal{P}$ of $\mathcal{K}$ as:

$$P_{\mathcal{P}}(C) := \sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models (C \sqsubseteq \perp)^\top} P_{\mathcal{J}}(\mathcal{V})$$

We define the probability of C being partially satisfiable in $\mathcal{K}$ as:

$$P_{\mathcal{K}}(C) := \sup_{\mathcal{P} \models \mathcal{K}} (P_{\mathcal{P}}(C))$$

Note that while this definition follows the same pattern as the definition of the probability of a subsumption we have changed the infimum into a supremum. We have done this as it makes more sense to work with the model which has C satisfiable in the most cases possible. If we instead used an infimum we could have models that have an element in C with far greater probability than is indicated by our calculation.

Now that we have fixed what it means for some concept to be satisfiable with probability p we have to provide a way to do this calculation. We again show that we can reduce this problem to subsumption, in particular the probability of a subsumption.

Theorem 5.5.5. The concept C is partially satisfiable in knowledge base $\mathcal{K}$ with probability $1 - P_{\mathcal{K}}((C \sqsubseteq \perp)^\top)$.

Proof. Note that in a probabilistic model $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of a knowledge base $\mathcal{K}$

$$\sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models C \sqsubseteq \perp} P_{\mathcal{J}}(\mathcal{V}) = 1 - \sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models (C \sqsubseteq \perp)^\top} P_{\mathcal{J}}(\mathcal{V})$$

We therefore have that

$$\sup_{\mathcal{P} \models \mathcal{K}} (P_{\mathcal{P}}(C)) = 1 - \inf_{\mathcal{P} \models \mathcal{K}} (P_{\mathcal{P}}((C \sqsubseteq \perp)^\top))$$

□

We can now work out how likely it is that someone will win a lottery. This is quite useful because this information could be used to determine if its worth playing in a lottery before buying a ticket. However, we should point out that because we use a supremum in the definition we would get the most favorable odds of actually winning. This is because the actual model we will consider (the model that maximizes the probability of winning being satisfiable) will have a winner in all worlds where it is possible to have a winner. So in order for our lottery example to work in practice we would have to set up the contexts (and the Bayesian Network) carefully in order for this result to be useful.

Finally note that concept satisfiability does not check whether a specific individual is a member of a concept. Therefore we are only able to check whether its possible or how likely it is that anyone will win the lottery. In order to check for a specific individual we would have to use instance checking instead. Fortunately we have studied this problem and present our results in the next section.

5.6 Instance Checking

We introduced instance checking for $\mathcal{ALC}$ in Section 2.3.4 where we described it as the problem of determining whether a specific named individual is necessarily a member of some concept. That is, is the named individual a member of the concept in all models of the knowledge base. We now show how the notion of instance checking can be extended to Bayesian Description Logics, and in particular to $\mathcal{BALC}$. We do so by defining a new reasoning problem: namely probabilistic instance checking.

Definition 5.6.1 (Instance). Given an individual name a , a concept C , a primitive context ϕ , and a knowledge base $\mathcal{K}$ we say that a is an instance of C in context ϕ for $\mathcal{K}$, written as $\mathcal{K} \models C(x)^\phi$, iff for all probabilistic models $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of $\mathcal{K}$ we have that $a^{\mathcal{V}} \in C^{\mathcal{V}}$ for all $\mathcal{V} \in \mathcal{J}$ with $v^{\mathcal{V}} \models \phi$.

This definition again uses contexts to denote in which worlds a given statement is required to hold. This allows instance queries in $\mathcal{BALC}$ to express that an individual is only required to be an instance of some concept under certain conditions. For example if we have an ontology representing a school we may want to check if some student has homework. We could do so by checking if this particular student is a member of the class of things that has homework:

$$\exists has.Homework(Dave)^{\top}$$

However this is not quite good enough because we do not know if its currently school holidays in which case Dave may already have finished his homework! Fortunately we can include this in our instance check:

$$\exists has.Homework(Dave)^{\neg holiday}$$

That is we would check whether *Dave* is an instance of something that has *Homework* with the condition that he is not currently on *holiday*. This seems sensible as we would probably expect Dave to have some homework when not on holiday (and maybe some even then!). Note that in the case where the context associated with the instance check is $\top$ this definition is very similar to the classical case. In this case it would be required that the named individual be a member of the concept in all cases (in all worlds with positive probability) which is similar to the classical case. We next show how we go about providing a procedure that solves this problem.

In the case of classical $\mathcal{ALC}$ we can reduce instance checking to consistency checking. However, in $\mathcal{BALC}$ this is not trivially possible. This is because if we find that that $\mathcal{K}' = (\mathcal{T}, \mathcal{A} \cup \{\neg C(a)^\phi\}, \mathcal{B})$ is inconsistent (i.e. the assertion that the knowledge base does not entail that a is an instance of C in ϕ) we have only shown that there exists *one* V -interpretation where $C(a)$ is entailed (one V -interpretation where $\neg C(a)^\phi$ cannot hold). This is not sufficient as we need to show that it holds in all cases where the context is matched. We can however reduce instance checking to exact subsumption fairly simply as we show below.

Theorem 5.6.2. Given an individual name a , a concept C , a context ϕ , and a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ then a is an instance of C in ϕ iff $P_{\mathcal{K}'}((D \sqsubseteq C)^\phi) = 1$ where D is a new concept name not in $\mathcal{T}$ and $\mathcal{K}' = (\mathcal{T}, \mathcal{A} \cup \{D(a)^\phi\}, \mathcal{B})$.

Proof. ($\Rightarrow$) If $P_{\mathcal{K}'}((D \sqsubseteq C)^\phi) = 1$ then in all probabilistic models $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of $\mathcal{K}'$ we have for all $\mathcal{V} \in \mathcal{J}$ that $\mathcal{V} \models (D \sqsubseteq C)^\phi$. Therefore for all $\mathcal{P}$ of $\mathcal{K}'$ we have that $a^\mathcal{V} \in C^\mathcal{V}$ for all $\mathcal{V} \in \mathcal{J}$ with $v^\mathcal{V} \models \phi$. Therefore a is an instance of C in ϕ for $\mathcal{K}'$ and is therefore an instance of C in ϕ for $\mathcal{K}$.

($\Leftarrow$) If a is an instance of C in ϕ in $\mathcal{K}$ then in all probabilistic models $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of $\mathcal{K}'$ we have for $\mathcal{V} \in \mathcal{J}$ such that $v^\mathcal{V} \models \phi$ that $\mathcal{V} \models D \sqsubseteq C$. Furthermore for all $\mathcal{V}$ such that $v^\mathcal{V} \not\models \phi$ we trivially have that $\mathcal{V} \models (D \sqsubseteq C)^\phi$. Therefore since in all probabilistic models we have that in all $\mathcal{V} \in \mathcal{J}$ that $\mathcal{V} \models D \sqsubseteq C^\phi$ then $P_{\mathcal{K}'}((D \sqsubseteq C)^\phi) = 1$. $\square$

Note that the second part of this proof works because we can never infer that there is another individual in D because D does not appear in the TBox (i.e. there is no terminological information that involves D). Furthermore, since we have reduced instance checking to probabilistic subsumption we have that instance checking is in the same complexity class as probabilistic subsumption. This gives us the following lemma.

Lemma 5.6.3. Probabilistic instance checking in a knowledge base $\mathcal{K}$ is in the complexity class $O(2^{|\mathcal{K}|+|V|})$.

Now that we have an algorithm for instance checking in $\mathcal{BALC}$ we can now check whether Dave has done his homework. However this only gives us a yes or no answer about his homework. There are many, hopefully unlikely, situations which we may not know about that would result in a student not doing their homework. We cannot represent this information while still keeping the context simple and retain the ability to check instances of students having done their work. However if we were to define a probability of a student having done their homework instead of just a binary answer we can include this information without complicating our instance check unnecessarily.

Continuing with the pattern we have established previously we first defined a decision problem (is an individual an instance of a concept) and now define the probability calculation for instance checking. We would like this probability to represent how likely it is for a given individual to be a member of some concept in a given context. We formalize the probability of an instance in a very similar way to the probability of a subsumption.

Definition 5.6.4 (probability of an instance). We define the probability of an instance in a probabilistic model $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of a knowledge base $\mathcal{K}$ as:

$$P_{\mathcal{P}}(C(x)^\phi) = \sum_{\mathcal{V} \in \mathcal{J}, \mathcal{V} \models C(x)^\phi} P_{\mathcal{J}}(\mathcal{V})$$

That is, the probability of the instance in a specific model is equal to the sum of all worlds in which the contextual instance holds. Note that the instance holds both when the context holds and the instance holds classically or trivially when the context does not hold.

We define the probability of the instance with regards to a knowledge base $\mathcal{K}$ as:

$$P_{\mathcal{K}}(C(x)^{\phi}) = \inf_{\mathcal{K} \models \mathcal{P}} P_{\mathcal{P}}(C(x)^{\phi})$$

As we did for subsumption we define the probability of all instance checks for an inconsistent knowledge base as 1 in order to keep our definitions consistent with probability theory.

Similar to the algorithm for solving the decision problem we now show that the probability calculation can be reduced to subsumption. In particular it is reduced to calculating the probability of a subsumption in the knowledge base. We do this by introducing a new concept D not in the knowledge base and asserting that the individual a is a member of this concept in the given context ($D(x)^{\phi}$) and then calculate the probability of D being subsumed by the given context. Stated more formally we have that:

Theorem 5.6.5. Given an individual name a , a concept C , a primitive context ϕ , and a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ then a is an instance of C in ϕ with probability $P_{\mathcal{K}'}((D \sqsubseteq C)^{\phi})$ where D is a new concept name not in $\mathcal{T}$ and $\mathcal{K}' = (\mathcal{T}, \mathcal{A} \cup \{D(a)^{\phi}\}, \mathcal{B})$.

Proof. This can be seen to be true because for any probabilistic model $\mathcal{P} = (\mathcal{J}, P_{\mathcal{J}})$ of $\mathcal{K}$ for any V -interpretation $\mathcal{V}$ if we have that $\mathcal{V} \models C(a)^{\phi}$ then $\mathcal{V} \models (D \sqsubseteq C)^{\phi}$ and vice versa. $\square$

From this result we again see that calculating the probability of an instance can be reduced in constant time to probabilistic subsumption. Since we only add a single statement to the ABox of the knowledge base the input size does not change meaningfully post reduction giving us the following lemma:

Lemma 5.6.6. Calculating the probability of an instance in a knowledge base $\mathcal{K}$ can be done in $O(2^{||\mathcal{K}||+|V|})$ time.

This concludes our work on instance checking in $\mathcal{BEL}$. We have defined both a decision problem, which determines if some individual is necessarily an instance of a concept in some context, and a probability calculation, which determines with what probability a named individual is a member of a concept given a context. Furthermore we have provided algorithms, and their complexity results, for both of these problems.

5.7 Most Likely Context

In the field of Bayesian Networks we are often not only interested in knowing if, or with what probability, a consequence holds. We are also interested in finding the explanation of consequences, i.e. what conditions a consequence likely follows from. This kind of reasoning is useful, for example, whenever we observe a random event and are now interested in knowing what is the most likely cause for the observation. The most likely context problem extends this kind of reasoning to Bayesian Description Logics. In the context of Bayesian DLs (specifically $\mathcal{BEL}$) this problem

has been investigated for subsumption. The most likely context of a subsumptions allows, given a subsumption, a check which returns the context with highest probability that entails the subsumption. This reasoning service gives $\mathcal{BEL}$ the expressive power to provide insights on what conditions may result in subsumption relationships in models of a given knowledge base.

We will present all work we have done on this problem in this section. Finding the most likely context for a subsumption can also be described as finding the most likely explanation for a subsumption (given that the subsumption holds). This problem is closely related to classical axiom pinpointing in classical Description Logics. Classical axiom pinpointing finds the axioms that are responsible for a given entailment for a knowledge base [4]. These problems differ in that most likely context goes a step further than only finding the responsible axioms in that it also requires that the context representing these axioms must have maximal probability in the Bayesian Network. Formally a most likely context (MLC) for a subsumption is defined as,

Definition 5.7.1 (Most Likely Context of a Subsumption). Given a $\mathcal{BALC}$ knowledge base $\mathcal{K}$, C and D two concept names we say that a primitive context κ is a most likely context for the subsumption $C \sqsubseteq D$ if (i) $\mathcal{K} \models (C \sqsubseteq D)^\kappa$ and (ii) for all contexts κ' with $\mathcal{K} \models (C \sqsubseteq D)^{\kappa'}$ then $P(\kappa) \geq P(\kappa')$.

The usefulness of this reasoning problem can most easily be seen by looking at an example. Consider a Bayesian DL ontology that models information about different birds. We assume that this ontology uses contexts to represent meta information about the location of different birds. For example the ontology would represent the location of a migratory bird using the different seasons (or times of year) as a context. Then if we spot a bird while bird watching we can determine which context (reason) is most likely to be responsible for its current location.

In $\mathcal{BEL}$ the most likely context has previously been studied by Ceylan et al [9], and was shown to be $NP^{PP} - Complete$. For $\mathcal{BALC}$ at the time of writing we have no algorithm that solves this problem. We intend to address this in future work on the logic.

In the remainder of this section we will briefly outline an approach we are investigating, at the time of writing, in order to find an algorithm for the problem (most likely context of a subsumption). Furthermore we will also briefly discuss extending the most likely context problem to other reasoning services. In particular we will discuss how the most likely context for an instance and concept satisfiability can be defined.

From the work we have already presented in this thesis it is apparent that the special context $\phi_{\mathcal{K}}^C$ is very powerful. We have used it to provide solutions for multiple different reasoning problems. We believe that it has an application in the MLC problem for subsumption as well. Specifically we have already seen that, in the context of a subsumption, $\phi_{\mathcal{K}'}$ encodes all worlds that lead to the given subsumption holding. Remember that $\phi_{\mathcal{K}'}$ is the context that encodes all worlds in which the modified KB (modified such that the given subsumption does not hold) is inconsistent classically. If we exploit this property and construct $\phi_{\mathcal{K}'}$ for the subsumption given in the MLC problem we know that for the MLC κ we have that $\kappa \models \phi_{\mathcal{K}'}$. This means that we now only have to find a way to use $\phi_{\mathcal{K}'}$ to find κ in order to have an algorithm for the most likely context of a subsumption.

In the initial work on $\mathcal{BEL}$ the most likely context problem was only considered in the context of a subsumption. We believe that this is not the only problem for

which MLC context is applicable. In particular we think that the probabilistic instance problem as well as the probabilistic satisfiability problem would benefit from having a MLC version. We can motivate MLC instance checking using the bird example we gave previously. Currently we would have to assume that the entire class of the bird in question is in the same location in order to perform a MLC subsumption check. Clearly this is not the best solution to the problem. If this problem was defined for an instance then this problem could be solved in a more elegant way. We define the most likely context of an instance as,

Definition 5.7.2 (Most Likely Context of an Instance). Given a $\mathcal{BALC}$ knowledge base $\mathcal{K}$, a concept C , and a named individual x we say that the primitive context κ is the most likely context of the instance $C(x)$ iff (i) $\mathcal{K} \models C(x)^\kappa$ and (ii) for all contexts κ' with $\mathcal{K} \models C(x)^{\kappa'}$ then $P(\kappa) \geq P(\kappa')$.

For the most likely context to satisfy a concept we can show its usefulness by considering that it would provide the ability to check which context contributes the most to a concept being satisfiable. This could be used in the bird example as a way to check which conditions allow birds to live in certain locations (i.e. check that the location is not empty of birds). We define this problem as follows,

Definition 5.7.3 (Most Likely Context to Satisfy a Concept). Given a $\mathcal{BALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ and a concept C we say that the primitive concept κ is a most likely context to satisfy C iff (i) $(\mathcal{T}_\kappa, \mathcal{A}_\kappa)$ has C satisfiable and (ii) for all contexts κ' with $(\mathcal{T}_{\kappa'}, \mathcal{A}_{\kappa'})$ with C satisfiable then $P(\kappa) \geq P(\kappa')$.

With these two new definitions we now have three most likely contexts problems for $\mathcal{BALC}$. Once algorithms for these problems have been found $\mathcal{BALC}$ will have the expressive power to provide multiple insights into the causes behind inference. This will bring the existing work on Bayesian DLs closer to the expressivity of Bayesian Networks with regards to reasoning about the causes of consequences. We intend to find algorithms that solve each of the MLC problems in future work.

Chapter 6

Conclusion

6.1 Summary of Contribution

The main contribution of this work is extending the probabilistic framework developed by Ceylan et al. [4] for the lightweight Description Logic $\mathcal{EL}$ and applying it to the more feature rich Description Logic $\mathcal{ALC}$. This includes both adaption of the syntax and semantics, where necessary, as well as the development of algorithms. The algorithms presented in this work includes algorithms for most reasoning problems initially defined for $\mathcal{BEL}$ (with the exclusion of Most Likely Context) as well as the development of algorithms for new reasoning problems (satisfiability, consistency, conditional subsumption, instance checking). With regards to the changes in syntax and semantics, while we based the syntax and semantics of $\mathcal{BALC}$ on $\mathcal{EL}$ we extended the probabilistic framework (the probabilistic semantics) to support full knowledge bases instead of only terminologies. Of particular note, not only did we extend our logic to allow the inclusion classical assertional knowledge but we also incorporated the ability to represent contextual assertional knowledge, which allows probabilistic reasoning that involves both terminological and assertional information. As shown in the review of the background works this is not that common among probabilistic Description Logics.

6.1.1 Syntax and Semantics

We have used probabilistic world semantics for $\mathcal{BALC}$. In particular, because of the contextual nature of $\mathcal{BALC}$, each probabilistic world has specific terminological and assertional knowledge that applies in that world. This allows for making classical $\mathcal{ALC}$ statements about terminological or assertional information that only have to hold if some condition (some probabilistic world) matches their associated context.

6.1.2 Total concept satisfiability

Total concept satisfiability is a new reasoning problem that we have defined in $\mathcal{BALC}$. It is similar to the satisfiability problem in $\mathcal{ALC}$ that determines whether it is possible for some concept to be non empty. In $\mathcal{BALC}$ we extended this problem with probabilistic semantics by requiring that a concept be non-empty in all probabilistic worlds in order for it to be considered totally satisfiable. That is we say that some concept C is totally satisfiable if it is possible to have an element in C in all possible worlds.

We have provided a tableaux algorithm based on axiom pinpointing that solves this problem for a given knowledge base and concept in exponential time on the size of the knowledge base and size of the attached Bayesian Network, that is in $O(2^{||\mathcal{K}||+|V|})$ time. In contrast practical existing algorithms for solving satisfiability in

$\mathcal{ALC}$ are already exponential in the size of the input knowledge base, that is $O(2^{||\mathcal{K}||})$ time. This shows that the total concept satisfiability problem is not harder (not in a different complexity class) than the classical case. This is a good result as it means that our version of probabilistic semantics can be used in $\mathcal{ALC}$ at no additional cost to complexity.

6.1.3 Full knowledge base consistency

Knowledge base consistency is an existing reasoning problem in most Description Logics, although in some logics such as $\mathcal{EL}$ it is trivial since knowledge bases cannot be inconsistent. Classically a knowledge base is consistent if there exists some model for it. We have used this same definition for consistency for $\mathcal{BALC}$ and it is also used by $\mathcal{EL}$. However, the consistency problem has no algorithm in the existing work on $\mathcal{EL}$ as it has been shown that $\mathcal{EL}$ knowledge bases are always consistent.

We showed that in $\mathcal{BALC}$ consistency checking can be reduced to an instance of the total concept satisfiability problem, without changing the input size at all. As such we have that checking the consistency of a knowledge base is again in the exponential complexity class of $O(2^{||\mathcal{K}||+|V|})$

6.1.4 Subsumption of concepts

Concept subsumption is an existing problem in both classical $\mathcal{ALC}$ and $\mathcal{EL}$, and has been studied in $\mathcal{BEL}$. As such we have defined subsumption consistently with the definition in $\mathcal{EL}$, that is a concept C subsumes a concept D in context κ if, and only if, in all probabilistic worlds such that κ holds then C is subsumed by D in the classical sense (every individual in C must be in D). Since this definition does not have the desired probabilistic information we would like we next defined the probability of a subsumption.

We defined the probability of a subsumption in the same way as in $\mathcal{BEL}$. That is we first defined the probability of a contextual subsumption $(C \sqsubseteq D)^\kappa$ with respect to a probabilistic model as being the sum of the probabilities of all worlds in which either $C \sqsubseteq D$ holds classically or the context κ does not hold. We then defined the probability of a subsumption in a knowledge base as the infimum of the probability of the subsumption across all models of the knowledge base. We then defined conditional subsumptions which involve calculating the probability of a contextual subsumption when given some evidence about the context. We showed that conditional contextual subsumptions can be converted into a single contextual subsumption and three probability calculations in the Bayesian Network.

We then went on to define 3 subclasses of the subsumption problem:

- positive subsumption - determining whether some contextual subsumption holds with probability greater than 0 in a knowledge base.
- exact subsumption - determining the exact probability with which a subsumption holds in a knowledge base.
- p -subsumption - determining if a contextual subsumption holds with probability greater or equal to p in some knowledge base.

Note that in each case these subsumption problems have both a non-conditional and conditional version, but since we know how these two problems are related we need only a single algorithm for each problem.

We showed that the positive subsumption problem can be reduced to an instance of the consistency problem in constant time (we add two new assertions into the ABox before running the consistency algorithm). This also gives us the result that the positive subsumption problem is exponential time ($O(2^{||\mathcal{K}||+|V|})$).

For the exact and p -subsumption problem we provided a single algorithm that solves both problems. With the only difference being that we allow early termination of the algorithm in the case of p -subsumption when cumulative probability reaches the required value. We showed that this algorithm is again exponential with big O of $O(2^{||\mathcal{K}||+|V|})$.

6.1.5 Partial (probabilistic) concept satisfiability

The partial concept satisfiability problem, like the total concept satisfiability problem, is a new problem that we studied. It is similar to the working out the probability of a subsumption in that it focuses on working out how likely it is that a concept is satisfiable in some knowledge base instead of whether it is satisfiable in all worlds.

Importantly this means that total concept satisfiability does not require the concept to be satisfiable in all worlds and instead works out a probability based on the probabilities of the worlds where this concept is satisfiable. Intuitively the probability of a concept being satisfiable is equal to the sum of the probabilities of all worlds in which the concept is classically satisfiable. Note that this definition has the result that a concept that is satisfiable with probability 1 is totally satisfiable.

We showed that partial concept satisfiability can be reduced to an instance of the subsumption problem and that the probability of a concept being satisfiable is equal $1 - P_{\mathcal{K}}((C \sqsubseteq \perp)^\top)$. This allows us to both check whether a concept is partially satisfiable and to calculate its probability using the subsumption algorithm. This gives the result that partial concept satisfiability is in exponential with complexity $O(2^{||\mathcal{K}||+|V|})$.

6.1.6 Instance Checking

Probabilistic instance checking is the final problem that we studied for $\mathcal{BALC}$. Similar to many of the other reason problems instance checking in $\mathcal{BALC}$ has both a decision and a probability calculating version. We have defined the decision problem version to be determining whether a given contextual assertion (of the form $C(x)^\phi$) holds in all probabilistic models of the knowledge base. While the probability calculation is similar to probabilistic subsumption in that it involves the summation of the probabilities of worlds in which $C(x)^\phi$ holds in the worst case (the infimum). We have shown that we can reduce probabilistic instance checking to an instance of the subsumption problem in both cases and as such we have a complexity of $O(2^{||\mathcal{K}||+|V|})$.

6.2 Open Issues and Future Work

There are three clear avenues of future work open for $\mathcal{BALC}$ at this point. Firstly the remaining reasoning problems that have been defined for $\mathcal{BEL}$ should be extended to $\mathcal{BALC}$ where possible. This includes the most likely context of a subsumption as well as the more recent work on conjunctive query answering.

The most likely context of a subsumption is one of the reasoning problems presented in the founding work for $\mathcal{BEL}$ as such until this problem is solved for $\mathcal{BALC}$ we have not fully extended the probabilistic framework of $\mathcal{BEL}$ to $\mathcal{ALC}$. Completing

this work is a priority for future research in $\mathcal{BALC}$. Furthermore, since we have defined additional problems (probabilistic instance checking and partial satisfiability) that have a probability calculation the most likely context problem could further be extended to these problems. To do this we will define a most likely context for a contextual assertion (as the context with maximal probability that entails the assertion) and the satisfiability of a context (the context with maximal probability that entails that a concept is satisfiable).

Secondly the tableaux algorithm used in this work is lacking in optimizations that would lead to good practical complexity results. This is largely because the algorithm does not make use of any existing optimizations techniques for the $\mathcal{ALC}$ tableaux. As such a review of existing literature on reasoners for $\mathcal{ALC}$ would likely lead to several practical optimizations to our tableaux algorithm.

Finally, while theoretical results about $\mathcal{BALC}$ are valuable they cannot be directly applied to practical ontologies. In order to do this a reasoner of some form would have to be implemented. Once a reasoner is developed it should be used to perform benchmarks so that practical complexity results can be achieved. In some sense this overlaps with the development of optimizations to the tableaux and as such will probably be studied in parallel in the future.

Bibliography

- [1] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.
- [2] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, and Daniele Nardi. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2010.
- [3] İl Ceylan, Julian Mendez, and Rafael Peñaloza. “The Bayesian ontology reasoner is BORN”. In: *CEUR Workshop Proceedings* (2015).
- [4] İsmail İlkan Ceylan. “Context-sensitive bayesian description logics”. MA thesis. Technische Universität Dresden, 2013.
- [5] İsmail İlkan Ceylan. “Query Answering in Bayesian Description Logics.” In: *Proceedings of the Description Logics Workshop*. 2015.
- [6] İsmail İlkan Ceylan and Rafael Peñaloza. “Dynamic Bayesian Description Logics.” In: *Description Logics*. 2015.
- [7] İsmail İlkan Ceylan and Rafael Peñaloza. “Reasoning in the Description Logic $\mathcal{BEL}$ Using Bayesian Networks.” In: *AAAI Workshop: Statistical Relational Artificial Intelligence*. 2014.
- [8] İsmail İlkan Ceylan and Rafael Peñaloza. “The Bayesian Description Logic $\mathcal{BEL}$ ”. In: *International Joint Conference on Automated Reasoning*. Springer. 2014, pp. 480–494.
- [9] İsmail İlkan Ceylan and Rafael Peñaloza. “The Bayesian Ontology Language $\mathcal{BEL}$ ”. In: *Journal of Automated Reasoning* 58.1 (2017), pp. 67–95.
- [10] Gregory F Cooper. “The computational complexity of probabilistic inference using Bayesian belief networks”. In: *Artificial intelligence* 42.2-3 (1990), pp. 393–405.
- [11] Adnan Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009.
- [12] Michael Dürig and Thomas Studer. “Probabilistic ABox Reasoning: Preliminary Results.” In: *Proceedings of the Description Logics Workshop*. 2005, pp. 104–111.
- [13] Jochen Heinsohn. “Probabilistic description logics”. In: *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 1994, pp. 311–318.
- [14] Ian Horrocks and Peter Patel-Schneider. “Reducing OWL entailment to description logic satisfiability”. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 1.4 (2004), pp. 345–357.
- [15] Manfred Jaeger. “Probabilistic Reasoning in Terminological Logics.” In: *Proceedings of Principles of Knowledge Representation and Reasoning* 94 (1994), pp. 305–316.

- [16] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [17] Daphne Koller, Alon Levy, and Avi Pfeffer. "P-CLASSIC: a tractable probabilistic description logic". In: *AAAI/IAAI 1997* (1997), pp. 390–397.
- [18] Kevin Lee. "Inconsistency Management for Description Logics". PhD thesis. The University of New South Wales, 2012.
- [19] Kevin Lee, Thomas Meyer, and Jeff Z Pan. "Computing maximally satisfiable terminologies for the description logic ALC with GCIs". In: *Proceedings of the Description Logics Workshop*. 2006, p. 253.
- [20] Thomas Lukasiewicz and Umberto Straccia. "Managing uncertainty and vagueness in description logics for the semantic web". In: *Web Semantics: Science, Services and Agents on the World Wide Web 6.4* (2008), pp. 291–308.
- [21] Thomas Meyer, Kodylan Moodley, and Ivan Varzinczak. "First steps in the computation of root justifications". In: *Proceedings of Automated Reasoning about Context and Ontology Evolution*. Aug. 2010.
- [22] Thomas Meyer, Kevin Lee, Richard Booth, and Jeff Z Pan. "Finding maximally satisfiable terminologies for the description logic ALC". In: *Proceedings of the National Conference on Artificial Intelligence*. Vol. 21. 1. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 2006, p. 269.
- [23] Kodylan Moodley. "Debugging and repair of Description Logic ontologies". PhD thesis. University of KwaZulu-Natal, 2017.
- [24] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.
- [25] Manfred Schmidt-Schauß and Gert Smolka. "Attributive concept descriptions with complements". In: *Artificial intelligence* 48.1 (1991), pp. 1–26.
- [26] Kent A Spackman. "Normal forms for description logic expressions of clinical concepts in SNOMED RT." In: *Proceedings of the American Medical Informatics Association Symposium*. 2001, p. 627.
- [27] Philipp M Yelland. "An alternative combination of Bayesian networks and description logics". In: *International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers; 1998. 2000, pp. 225–234.