DBAI
DBAI

TECHNICAL REPORT

INSTITUT FÜR INFORMATIONSSYSTEME

ABTEILUNG DATENBANKEN UND ARTIFICIAL INTELLIGENCE

Contributions to the Complexity Analysis of Non-Monotonic Reasoning

DBAI-TR-2000-40

Robert Baumgartner

Institut für Informationssysteme
Abteilung Datenbanken und
Artificial Intelligence
Technische Universität Wien
Favoritenstr. 9
A-1040 Vienna, Austria
Tel: +43-1-58801-18403
Fax: +43-1-58801-18492
sekret@dbai.tuwien.ac.at
www.dbai.tuwien.ac.at

DBAI TECHNICAL REPORT
JUNE 2000

TECHNISCHE UNIVERSITÄT WIEN

DBAI TECHNICAL REPORT
DBAI TECHNICAL REPORT DBAI-TR-2000-40, JUNE 2000

Contributions to the Complexity Analysis of Non-Monotonic Reasoning

Abstract. Part One of this report is a presentation of known results in the areas of logic and complexity theory written in an intuitive style and presents a number of examples. Among other things several non-monotonic formalisms which can be used for knowledge representation in artificial intelligence and their relationship to each other are discussed.

The ample second part consists of a large variety of complexity results for model checking of non-monotonic formalisms and for determining a belief pair of a three-valued semantics of autoepistemic logic.

Model Checking is the question whether a given interpretation satisfies a theory (in this case a non-monotonic theory) and is a viable alternative to reasoning. A number of open complexity problems for model checking with default logic and with default logic with weak extensions is solved; for instance it is shown that model checking with normal default theories is Δ_2^P-complete, however in the case of weak extensions Σ_2^P-complete. Additionally, default theories in extended Horn normal form are introduced and favourable complexity results, in some cases even polynomial algorithms, for such theories are derived.

Moreover, the complexity of model checking with disjunctive default logic, stationary default logic and autoepistemic logic is proved, and a number of cases with an easier model checking problem than in the general cases is analysed. The discussion of these results allows a better understanding of complexity issues and to draw conclusions with respect to translatability of non-monotonic formalisms.

Finally, the complexity of determining a belief pair in three-valued fixed-point semantics of autoepistemic logic (resp. the question if a formula can be inferred from the belief pair of a given autoepistemic theory) is considered and solved, both for the general case (Δ_2^P-completeness) and if the theory contains only one top-level modality ($\Delta_2^P[O(\log n)]$-completeness).

Keywords: *autoepistemic logic; complexity theory; default logic; model checking; non-monotonic reasoning; polynomial hierarchy; variants of default reasoning.*

[1]Institut für Informationssysteme, Technische Universität Wien, Favoritenstr. 9, A-1040 Wien, Austria. E-mail: baumgart@dbai.tuwien.ac.at.

Acknowledgements: This work was supported by the Austrian Science Fund Project N Z29-INF

This technical report is based on the doctoral thesis of the author

Contents

I **1**

1 Introduction **3**

1.1 Setting . . . 3
1.2 Model Checking for Default Logic . . . 3
1.3 New Complexity Results . . . 4
1.4 About a three-valued Semantics for Autoepistemic Logic . . . 5
1.5 About the following Chapters . . . 5

2 Some Words about Logic **7**

2.1 Languages . . . 7
2.2 A short glimpse into Recursion Theory . . . 8
2.3 Classical Logic: Syntax and Semantics . . . 9
2.3.1 Propositional Logic . . . 9
2.3.2 Predicate Logic . . . 13
2.3.3 Herbrand models . . . 16
2.4 Modal and Epistemic Logics . . . 17
2.4.1 Modal Logic . . . 17
2.4.2 Epistemic Logic . . . 20
2.5 Non-monotonic Reasoning . . . 21
2.5.1 Closed World Assumption and Circumscription . . . 22
2.5.2 Logic Based Abduction . . . 23
2.5.3 Default Logic . . . 24
2.5.4 Variants of Default Logic . . . 29
2.5.5 Autoepistemic Logic . . . 32
2.5.6 Modal Non-monotonic Logics . . . 34

2.5.7 Logic Programming 34
2.5.8 General Properties of Non-monotonic Reasoning 36

3 Complexity Theory **39**
3.1 Introduction 39
3.2 The Turing Machine 41
3.3 Complexity Classes 49
3.4 Time and Space Bounds 50
3.5 Oracle machines 53
3.6 Search and Function Problems 54
3.7 Hardness 54
3.8 The Polynomial Hierarchy 57
3.9 Some Complete Problems 59
3.10 NP-trees 61
3.11 Logical Characterisations 61

4 Complexity of Non-monotonic Reasoning **63**
4.1 Model Checking vs. Reasoning 63
4.2 Complexity Results for Non-Monotonic Reasoning 64
4.3 Default Logic: Complexity 64
4.4 Complexity of Reasoning with Weak Extensions, Disjunctive Default Logic and Stationary Default Logic 67
4.5 Complexity of AEL and Non-monotonic Modal Logics 68
4.6 Closed World Assumption: Complexity 68
4.7 Complexity of Logic Based Abduction 69
4.8 Complexity of Logic Programming Formalisms 69
4.8.1 Stable Model Semantics 70
4.8.2 Partial stable model Semantics 72
4.8.3 Closed World Setting 72
4.8.4 Perfect Model Semantics 73
4.8.5 Answer Set Semantics 73
4.8.6 General Logic Programming and Datalog 74
4.9 Translatability Issues 74
4.9.1 Representational Succinctness 75

4.9.2 Expressive Power 75
4.10 Summary and Stable Class Semantics 76

II **79**

5 Complexity of Model Checking for Propositional Default Logics: New Results and Tractable Cases **81**
5.1 The Model Checking Approach 81
5.2 Some Useful Tools 83
5.3 Model checking for Normal Default Theories is $\Delta_2^P[O(\log n)]$-complete 86
5.4 Complexity of Model Checking for Disjunction-free and EHNF Default Theories . 93
5.4.1 General Case 94
5.4.2 Normal Default Theories 99
5.5 Complexity of Model Checking for Default Theories with Weak Extensions 103
5.5.1 General Case 103
5.5.2 Normal Default Theories 104
5.5.3 EHNF default theories 106
5.6 Tautology Checking 107
5.7 Complexity of Model Checking with AEL and **N** 108
5.8 About Translations of Default Logic into Disjunctive Logic Programming 109

6 Complexity of Model Checking for Disjunctive and Stationary Default Logics **117**
6.1 Complexity of Model Checking for Disjunctive Theories 117
6.1.1 General Disjunctive Default Theories 119
6.1.2 Disjunction-free and EHNF theories 121
6.1.3 Weak Model Checking 130
6.2 Some words on Reasoning for Disjunctive Default Logic 134
6.3 Model Checking with Stationary Default Theories 135

7 Complexity of Fixed-point 3-valued Semantics for Autoepistemic Logic **137**
7.1 Definitions: Semantics and Syntax 138
7.2 Some useful Lemmata 139
7.3 Some Examples 141

7.4 General Case: Δ_2^P-hardness 142
7.5 Restriction to a Single Top Level Modality 147

8 Overview of Results 153

List of Figures

3.1 *Scheme of a Turing Machine* . . . 43
3.2 *Finite Automaton for Example 3.2.7* . . . 46
3.3 *A Simple Directed Graph* . . . 47
3.4 *Complexity Classes* . . . 58

5.1 *Deterministic Algorithm of Theorem 5.3.1 using* $O(\log n)$ *Oracle-Calls* . . . 90
5.2 *Good and Bad Defaults of Example 5.3.6* . . . 92
5.3 *Converse Non-Deterministic Algorithm of Corollary 5.3.4* . . . 93
5.4 *Non-Deterministic Algorithm of Lemma 5.4.3* . . . 96
5.5 *Deterministic Algorithm of Lemma 5.4.7* . . . 101
5.6 *Comparing Models of DLP and Default Logic* . . . 112

List of Tables

2.1 *Properties of Relations corresponding to Modal Axioms* 19
2.2 *Syntactical characterisation of Some Closed World Assumptions* 23
2.3 *Semantical characterisation of Some Closed World Assumptions* 23
2.4 *Computing Extensions* 28

3.1 *Polynomial vs. Exponential Growth* 40
3.2 *Binary Numbers divisible by 3* 44
3.3 *Class-Membership for Satisfiability* 50

4.1 *Brave Default Reasoning in Disjunction-free Default Theories* 65
4.2 *Complexity of Closed World Reasoning* 69

8.1 *Complexity of Model Checking for Default Theories* 153
8.2 *Complexity of Weak Model Checking for Default Theories* 154
8.3 *Complexity of Model Checking for Disjunctive Default Theories* 154
8.4 *Complexity of Model Checking for Limited Disjunctive Default Theories* 154
8.5 *Complexity of Weak Model Checking for Disjunctive Default Theories* 155
8.6 *Complexity of Weak Model Checking for Limited Disjunctive Default Theories* . . . 155
8.7 *Complexity of Stationary Model Checking for Default Theories* 155

Part I

Chapter 1

Introduction

1.1 Setting

This report is settled in the areas of *non-monotonic reasoning* and *complexity theory*. Non-monotonic reasoning (from now on sometimes abbreviated as NMR) is a kind of logic dealing with incomplete and inconsistent information; in NMR new premises may invalidate previous conclusions. In several kinds of NMR, negated facts can be assumed as long as nothing is known about them. Moreover, an agent can reach several (contradicting) conclusions or several preferred models depending on a *context*. NMR hence proves to be a useful knowledge representation tool in artificial intelligence helping to formalise common-sense reasoning. Until today several kinds of non-monotonic reasoning have been introduced, most famous among them are *default logic* and *autoepistemic logic*.

In (structural) complexity theory, classes of equally hard problems are compared in an abstract way, based on a kind of worst-case complexity and without considering special algorithms for solving a problem. Computer scientists above all prefer polynomial algorithms, however many interesting applications need an exponential solution time and are hence settled in some higher classes. Among those are problems like satisfiability in classical propositional logic and many problems of NMR (which are a little bit more involved than problems in propositional logic). Of special interest in this report are the classes of the polynomial hierarchy, which are introduced based on $\mathbf{P}$ and $\mathbf{NP}$ allowing access to oracles.

1.2 Model Checking for Default Logic

Some of the main contributions and new results of this report are settled in the area of model checking for default logic. The complexity of default reasoning, i.e. if a formula can be inferred of at least one or of all extensions of a default theory, is already well understood, however in search for model-based representations, the complexity of the model checking problem (instead of the inference problem) needs to be analysed.

In general, reasoning is for most variants of non-monotonic reasoning located on the second level of the polynomial hierarchy and is hence a little bit more involved than reasoning in propositional logic.

Model checking is a beneficial alternative simplifying reasoning tasks and allowing for representing the agent's knowledge as a semantic structure instead of a collection of formulae and additionally, this approach introduces a kind of closed-world assumption. Furthermore, the complexity of model checking is closely related to the notion of *representational succinctness* of comparing non-monotonic formalisms.

An interpretation is a *(weak) model* of a default theory iff it satisfies at least one (weak) extension of the theory (for details cf. Chapter 5; the concept of extensions is explained in Section 2.5.3, the concept of weak extensions in Section 2.5.4).

1.3 New Complexity Results

Liberatore and Schaerf [1998] already derived some results for the model checking scenario. However, they left open the case of normal default theories. We solved this problem and considered several further cases – like the complexity of model checking in case of disjunction-free theories, i.e. theories consisting only of conjunction of literals.

Moreover, we carry the model checking approach over to variants of default logic, in particular disjunctive and stationary default logic and exhibit some interesting results helping to understand related complexity issues.

A summary of our derived complexity results can be found in Tables 8.1 to 8.7. At this stage, let us only present a list of some of the main results of Chapter 5 and 6.

Since model checking is defined via extensions it is in general not easier than reasoning, however in many restricted cases we get advantageous complexity results compared to the reasoning approach.

- We solve the open problem of the exact complexity of model checking for normal default theories ($\Delta_2^P[O(\log n)]$-completeness) with a very involved proof using data-structure guessing techniques in several steps of the proof.
- We introduce a new ample class of default theories (theories in extended Horn normal form) with a tractable model checking problem in case of normal default theories and an $\mathbf{NP}$-complete model checking problem else.
- We show that weak model checking is different to model checking even Σ_2^P-complete if restricted to normal default theories.
- We analyse weak model checking for default theories in extended Horn normal form and show $\mathbf{NP}$-completeness for all cases except the one where the default theory is both prerequisite-free and normal.

- We generalise our results to Σ_2^P-completeness of propositional model checking with AEL and $\mathbf{N}$.
- We use the complexity results to draw some interesting conclusions in translatability issues concerning default logic and disjunctive logic programming.
- We introduce the concept of limited and not limited disjunctive default theories to obtain two kind of definitions for normal and semi-normal disjunctive default theories.
- We derive the exact complexity classifications for standard and weak model checking for disjunctive default logic and a large number of restrictions (like limited normal theories in extended Horn normal form).
- We show that even model checking is like reasoning Σ_2^P-complete for disjunctive default theories containing only conjunctions of literals. However, we give favourable results in case of limited disjunctive default theories by proving $\mathbf{NP}$-completeness.
- We generalise the concept of weak extensions to disjunctive theories in a way which is advantageous for complexity issues allowing $\mathbf{NP}$-completeness for not limited disjunction-free disjunctive default theories and even allowing tractability in case of normal and prerequisite-free theories in extended Horn normal form.
- Finally, we prove complexity results for model checking in the case of stationary default logic.

1.4 About a three-valued Semantics for Autoepistemic Logic

Furthermore, in Chapter 7, we settle some open complexity issues w.r.t. three valued fixed-point semantics for autoepistemic logic. In this setting, a belief pair, a set containing two sets of interpretions, the first one being a subset of the second one, is improved in a fixed-point iteration until the best approximation is reached. A belief pair is kind of an approximation to the possible world characterisation of stable expansions of an autoepistemic theory.

Determining a belief pair B of a modal theory T was known to be in Δ_2^P [Denecker *et al.*, 1998]. We strengthened the result to Δ_2^P-completeness in case T has more then one top-level modality and to $\Delta_2^P[O(\log n)]$-completeness if T only has one top-level modality. For the second proof we use Carnap's modal logic and extend the concept of $\mathbf{NP}$-trees.

1.5 About the following Chapters

Part One of this report consists of Chapters 2 to 4 – it is a presentation of known results in the area of logic and complexity theory to make this report self-contained and is written in an informal and

(I hope) intuitive style inventing a large number of examples and providing explanations for better understanding of the main aspects which are necessary to read the second part.

Part Two contains enhanced results of our papers which have been published (or are being prepared for publication) during the time the report was created, and a number of some further new results. Chapter 5 deals with the complexity of model checking for default logic, some restricted cases of default logic and default logic with weak extensions, and contains some new favourable results illustrating that the model checking approach is a viable alternative to the usual reasoning approach in this setting. Chapter 6 furthermore contains new results (which are not available in our papers) about the complexity of model checking for several variants of default logic, namely disjunctive default logic and stationary default logic. Finally, Chapter 7 settles some open complexity issues w.r.t. a three valued fixed-point semantics for autoepistemic logic. Chapter 8 concludes with a summary and topics worth for some further study and future research.

Chapter 2

Some Words about Logic

2.1 Languages

According to Brachman and Levesque [1984] a knowledge-based agent can be described at three levels: The abstract *knowledge level* (to say what the agent knows), the *logical level* (description in terms of logical sentences) and the *implementation level* (how the logical sentences are implemented on the system architecture).

The goal of the representation language is to give easy access in describing knowledge (*declarative approach* of system designing)

A knowledge representation language is defined by its semantics and its syntax. Syntax describes the possible configurations while semantics determines the facts in the world to which the sentences refer.

A logic is defined by its syntax and semantics. Syntax uses axioms and rule schemata and the notion of formal entailment, and semantics uses the notion of inference and models. Semantics is based on the world, while syntax works on the representation knowledge.

A syntax needs to be sound, and it generates new true sentences based on axioms and inference rules. In proof theory, suitable calculi are closely inspected, and in model theory, the corresponding semantics is dealt with. An inference procedure is complete if we can find a proof for it.

The goal of a logic is to formalise the semantic notion of inference to be used in a (syntactic) calculus. More details about syntax and semantics will be provided in the following sections w.r.t. basics of classical (propositional and predicate) logic.

The advantage of natural languages is that they are very expressive, but they also suffer from ambiguity. A good formal representation language should combine two aspects: expressiveness and succinctness, as well as being not ambiguous.

A sentence has no meaning by itself, hence the concept of interpretations is used, which combines syntactical and semantical aspects. In the next sections we will analyse some logics, like first-order logic, propositional logic and modal logic, and very briefly sketch some more, like temporal logic and fuzzy logic.

The situation calculus and related calculi were introduced to represent change in the world, and to use default-like rules to express that unmentioned changes remain as before. It can be shown that any system that makes decisions on the basis of past perceptions can be rewritten to use instead a set of sentences about the current world state, provided that these sentences are updated as each new percept arrives and as each action is done [Russell and Norvig, 1995]. However, change representation is not the main goal of this report, although this is also inherently non-monotonic.

2.2 A short glimpse into Recursion Theory

In every theory of mathematics, proofs use a semantic inference relation. The goal of Hilbert and other mathematicians was to create a system where every mathematical truth could be decided by using a set of axioms and rules and just manipulate symbols without any semantics in mind [Oberschelp, 1993].

In semantics the basic notion is that of "interpretations": an interpretation maps basic linguistic notions onto individuals, relations or operations.

Hilbert's program died at the latest when Gödel's famous incompleteness results were published and after the notion of computability was analysed in detail.

Consider Peano arithmetics, the arithmetic for natural numbers based on some easy axioms and one first-order scheme for induction: even with the use of such an axiom scheme, i.e. using an infinite set of axioms (given in first order logic, usually referred to as predicate logic), there are true sentences in this structure of natural numbers which cannot be derived with the use of these axioms. One example is the Theorem of Goodstone [Oberschelp, 1993] about super-base representations of numbers. However, this theorem can easily be proved in a stronger theory, namely the theory of ordinal numbers.

In detail, the first incompleteness Theorem of Gödel tells us that if a theory is "rich" enough, then there are sentences which are true but neither can be proved nor disproved.

The second incompleteness theorem tells us that, assuming a theory is consistent, proving the consistency of the theory stronger methods than provided within the theory are needed. However, observe that these theorems do *not* imply that there will never be a proof that number theory is consistent. There are proofs of the consistency of number theory, however, only with using stronger theories.

Of course the question remains open whether the stronger theory is consistent, and finally whether the fundament of mathematics, namely axiomatic set theory is consistent. We cannot give a proof, only relative consistency proofs of several additional axioms. This is no disadvantage: Gödel claimed that set theory worked well after some inconsistencies like the Russell paradox had been eliminated, hence, we use it – we have no other choice. Further discussions of these topics would lead too far, but we hope that we have given some basic ideas about the difference between semantics and syntax: (First-order) Peano-arithmetics is based on syntactically manipulating symbols whereas full arithmetics is a semantical concept. Furthermore the notions of truth (semantic) and provability (syntactic) are sometimes different (second-order logic).

Out of a variety of interesting articles in the area of recursion theory let us mention one interesting article of Christian [1996]. In the next chapter we will discuss the notion of computability, another very important notion of recursion theory (the "big brother" of complexity theory).

For more restricted settings, it is possible that semantics and syntax coincide, as shown by Gödel's completeness theorem for predicate logic.

2.3 Classical Logic: Syntax and Semantics

2.3.1 Propositional Logic

A proposition is a sentence of which we can ask if it is true or false.

Assume the following proposition: "Tolkien wrote *The Lord of the Rings*" and another proposition: "England was involved in the second world war". Operators used to join together such propositions to complex formulae are called *junctors*.

The above sentences can be joined together using for instance "and" or "while". There is a difference between these two examples of junctors.

For "and" only the truth value of the two propositions need to be known to obtain the truth value of the conjunction. These types of junctors are called *extensional* and beside conjunction, disjunction, implication and equivalence fall into this category.

However, "while" is an *intensional* junctor. If we replace in the proposition containing the two true propositions "Tolkien wrote *The Lord of the Rings* while England was involved in the second world war" the first one with another true proposition "Tolkien wrote *The Hobbit*" then the whole proposition becomes wrong.

In logic usually only extensional junctors are implemented. Propositions are abbreviated with small letters p, q, a, b, x, y, $\ldots$ whereas formulae are abbreviated with capital letters F, G, $\ldots$.

In propositional logic a finite or denumerable set of propositions $\mathcal{F}$ is considered. We use capital letters A, B, C, $\ldots$ to represent syntactic variables ranging over the set of formulae (see the below definition for formulae).

Junctors are the unary operator $\neg$ (not) and the binary operators $\wedge$ (and), $\vee$ (or). Additionally we use $\Rightarrow$ (implies, $A \Rightarrow B = \neg A \vee B)$), $\Leftrightarrow$ (equivalent, $A \Leftrightarrow B = A \Rightarrow B \wedge B \Rightarrow A$), $\oplus$ (exclusive or, $A \oplus B = (A \vee B) \wedge (\neg A \vee \neg B)$).

The propositional language $\mathcal{L}_{Prop}$ is made up of $\mathcal{F}$ and the set of junctors (most times only a subset of the above introduced junctors is used).

Definition 2.3.1 (Formulae) *Let $\odot \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow, \oplus\}$. The set of formulae F_L is inductively defined as follows:*

- *Truthhood ($\mathbf{t}$) and falsehood ($\mathbf{f}$) are atomic formulae.*
- *Propositional variables $p, q, x, y, a, b, \ldots$ are atomic formulae.*
- *An atomic formula is a formula.*
- *If F is a formula, then $\neg F$ is a formula.*
- *If F and G are formulae, then $(F \odot G)$ is a formula.*

To allow dropping of parentheses the following strength of junctors are introduced: $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ (from strongest to weakest). Moreover, same kinds of junctors are evaluated from the right to the left.

Example 2.3.2 *Let $F = (p \wedge q) \Rightarrow (((p \vee \neg q) \wedge q) \Rightarrow p)$ can equivalently be displayed as $F = p \wedge q \Rightarrow (p \vee \neg q) \wedge q \Rightarrow p$.*

Definition 2.3.3 *Atomic formulae and negated atomic formulae are referred to as literals.*

The *semantics* of propositional logic is simply based on evaluating formulae using the concept of truth tables (boolean evaluations).

Definition 2.3.4 (Valuation) *A valuation v is a mapping from the set of atomic formulae $\mathcal{F}$ into $\{\mathbf{t}, \mathbf{f}\}$:*

- *$v(a)$ yields the truth value of atom a;*
- *$v(\neg A) = \mathbf{t}$ iff $v(A) = \mathbf{f}$;*
- *$v(A \wedge B) = \mathbf{t}$ iff $v(A) = \mathbf{t}$ and $v(B) = \mathbf{t}$;*
- *$v(A \vee B) = \mathbf{t}$ iff $v(A) = \mathbf{t}$ or $v(B) = \mathbf{t}$;*
- *$v(A \Rightarrow B) = \mathbf{t}$ iff $v(A) = \mathbf{f}$ or $v(B) = \mathbf{t}$.*

A formula is called *tautology* if it yields the value *true* regardless of the values chosen for any symbols (i.e. for each v). A propositional formula is *satisfiable* if there exists at least one valuation which assigns $\mathbf{t}$ to the formula.

Definition 2.3.5 (Model) *A valuation v is called a model of a formula F if it evaluates F to true.*

Usually we use instead of v the equivalent notation of Herbrand models (cf. Section 2.3.3) and write $M \models F$ if F is true in this Herbrand interpretation. Logical consequence is defined as follows (sometimes we use, as it is usually done, the junctors and quantors of logic also for metalanguage as in the definition below; however, it should be clear at all stages if meta-language or object-language is meant):

Definition 2.3.6 (Inference) $\{F_1, F_2, \ldots\} \models G$ *holds if*

$$\forall M\,(M \models F_1, \ldots, M \models F_n \Rightarrow M \models G).$$

Notions of truth, validity and inference are not particulary practical in working, hence mathematicians consider formal calculi where the notions of provability, derivability, entailment and consistency are the basic notions. In this syntactical way, symbols are just treated as symbol without any reference of meaning. Starting from a set of axioms and rules formulae can be derived.

Definition 2.3.7 (Derivability) *Formula G is called derivable from a set of formulae* $\{F_1, F_2, \ldots\}$*, denoted as* $\{F_1, F_2, \ldots\} \vdash G$ *if A is derivable in the formal system with adding all* F_i *as additional axioms.*

There are many known calculi which syntactically characterise propositional logic. For instance, Hilbert style calculi, natural inference, or Gentzen style calculi. Let us present one Hilbert style calculus characterising classical propositional logic using the junctors $\wedge$, $\vee$ and $\Rightarrow$. Its axiom schemata are as follows.

(A1) $A \Rightarrow B \Rightarrow A$

(A2) $(A \Rightarrow B \Rightarrow C) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

(A3) $A \wedge B \Rightarrow A \quad A \wedge B \Rightarrow C$

(A4) $(A \Rightarrow B) \Rightarrow (A \Rightarrow C) \Rightarrow (A \Rightarrow B \wedge C)$

(A5) $A \Rightarrow A \vee B \quad B \Rightarrow A \vee B$

(A6) $(A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow (A \vee B \Rightarrow C)$

(A7) $(A \Rightarrow \neg B) \Rightarrow (B \Rightarrow \neg A)$

(A8) $A \Rightarrow \neg A \Rightarrow B$

(A9) $((A \Rightarrow \neg A) \Rightarrow A) \Rightarrow A$

and the single rule *Modus Ponens*: $A, A \Rightarrow B \vdash B$ saying that from A and $A \Rightarrow B$ it is, in one step, possible to infer B.

Definition 2.3.8 (Consistency) *A set of formulae* $\{F_1, F_2, \dots\}$ *is consistent iff* $\{F_1, F_2, \dots\} \nvdash F_L$.

Furthermore, a nice property holds in propositional logic – videlicet, the deduction theorem:

Theorem 2.3.9 (Deduction Theorem) *Let* F *and* G *be arbitrary propositional formulae. Then* $F \vdash G$ *iff* $\vdash F \Rightarrow G$.

This allows for easier reasoning algorithms; in predicate logic a similar, somehow restricted version of the deduction theorem holds, too, whereas in modal logic, the deduction theorem fails in general.

If we simultaneously *substitute* all occurrences of an atom a with an atom (or a formula) b in a formula F, we abbreviate the resulting formula as $F_{a|b}$. Propositional logic is closed under substitution, i.e. tautologies are preserved.

Example 2.3.10 *We want to determine if the formula scheme* $F = \neg(A \Rightarrow A) \Rightarrow B$ *can be derived in propositional logic (without further premises). Due to (A8) it is known that* $\vdash A \Rightarrow \neg A \Rightarrow B$. *We are allowed to substitute* A *with* $A \Rightarrow A$ *to get* $\vdash (A \Rightarrow A) \Rightarrow \neg(A \Rightarrow A) \Rightarrow B$. *This can due to the deduction theorem be rewritten as* $A \Rightarrow A \vdash \neg(A \Rightarrow A) \Rightarrow B$. *The premise can be dropped since* $\vdash A \Rightarrow A$ *is true and finally we obtain* $\vdash \neg(A \Rightarrow A) \Rightarrow B$.

It is well known that there are sound (every syntactically inferred conclusion is true), and complete (every true sentence can be inferred) formal systems for propositional logic:

Theorem 2.3.11 $\{F_1, F_2, \dots\} \models G$ *iff* $\{F_1, F_2, \dots\} \vdash G$.

Moreover, the truth value of each formula is decidable, a fact which is no longer true in predicate logic (and higher order logics are not even complete as stated in the previous section).

Due to this theorem it also follows that consistency is the syntactical counterpart of satisfiability.

Propositional logic is monotonic in the sense that already derived formulae cannot be invalidated by adding further premises:

Example 2.3.12 $p \vdash q \Rightarrow p$ *remains valid if some prerequisites are added like* $p, \neg p \vee q \vdash q \Rightarrow p$*; even if these new premises are contradicting the old ones, since ex falso quodlibet holds.*

Definition 2.3.13 (Disjunction-free) *A formula* F *is called disjunction-free if* F *is a conjunction of literals.*

Definition 2.3.14 (Polarity) *In the formula* $F = p$, p *is of positive polarity in* F. *Given a formula* $F = G(p) \vee H$, *the selected occurrence of* p *in* G *is in* F *of the same polarity as in* G. *Given a formula* $F = \neg(G(p) \vee H)$, *the selected occurrence of* p *in* G *is in* F *of the converse polarity as in* $G(p)$.

In classical propositional logic it holds that each formula can equivalently be represented as disjunctive (disjunction of conjunctions) or conjunctive normal form (conjunction of disjunctions). However, with respect to transforming these normal forms into each other without introducing new variables, the time needed is exponential.

In computer science, Horn formulae and Horn theories have attracted much interest.

Definition 2.3.15 *A Horn formula is a disjunction of atoms with at most one positive literal. A Horn theory is a conjunction of Horn formulae.*

A disjunction of literals is also called a clause. There are two main intermingled reasons why Horn formulae play such an important role: First of all, a Horn clause $\neg a_1 \vee \neg a_2 \vee \ldots \vee \neg a_n \vee b$ can be equivalently rewritten as $a_1 \wedge a_2 \wedge \ldots \wedge a_n \Rightarrow b$, which is the program clause in logic programming. On the other hand, computational issues are far easier with Horn theories, since they involve no options (i.e. no disjunctions) after the implication. For details, cf. Chapter 3.

Given a formula in propositional logic, one can ask if there is at least one evaluation which makes the whole formula true. This problem, known as satisfiability (short: SAT), already exhibits some traits of a hard problem. To find out if there is such an evaluation, each interpretation needs to be considered, and for n involved (propositional) variables we deal with 2^n interpretations.

The dual problem is the question if a formula is a tautology, and for both problems so far no polynomial algorithm is known, and it is highly uncertain that one ever will be found (unless $\mathbf{P} = \mathbf{NP}$, cf. later chapters for more details).

Example 2.3.16 *Is the formula $F = x \Rightarrow y \Rightarrow x$ satisfiable? Is it a tautology? For the first problem, one just needs to find one interpretation. Maybe with some intuition and good guessing one can find do so very fast, if there is one. However, in the second case, one needs to consider all interpretations. Choosing the interpretation $\{x\}$ for instance (standing for x is true, and y is false) yields $\mathbf{t} \Rightarrow \mathbf{f} \Rightarrow \mathbf{t}$, which is equivalent to $\mathbf{t} \Rightarrow \mathbf{t}$, which is true. Indeed, testing all interpretations tells us that this formula is a tautology – this tautology is used as axiom in many calculi.*

2.3.2 Predicate Logic

For most of the time we deal with propositional logic in this report. For sake of completeness, however, we recall predicate logic (the reader can for instance look up details in [Ebbinghaus *et al.*, 1996]).

Informally, in predicate logic quantifiers "for all x" and "for some x" are introduced; for instance, for all or some x a particular relation is true. The atomic formulae are now not the propositional atoms, but the relations used on functions, variables and constants, which can be true or false.

Formally, the semantics of predicate logics is defined via a structure consisting of a signature including relation symbols, function symbols and variables, and a given individual domain, which can be either finite or infinite. An interpretation of a formula consists of an evaluation function of the variables.

Although predicate logic still has the advantages of being complete and sound, it is only semi-decidable, i.e. we can give no algorithm deciding, given an arbitrary formula, if the formula is a theorem or not. We can only "construct a machine" which enumerates all theorems of predicate logic (and runs infinitely long).

Second order logic allows quantification not only over individual variables, but also over relation symbols. This powerful logic, and the more powerful logics allowing arbitrary quantifying, are not complete any longer. This fact is closely tied to the fact that one can give an axiomatic characterisation of number theory in second order logic – hence, somewhere there must be a drawback, namely that second-order logic itself cannot be formalised.

Formally, in predicate logics semantics is defined with assignments. Model theory is the use of logic in algebra and provides the requirements for a semantical treating.

An algebraic structure $\mathcal{A}$ consists of an individual domain A, relations, functions and constants. The language $\mathcal{L}_{Pred}$ of predicate logic includes variables (usually abbreviated as v_i, x, y, $\ldots$), junctors, quantifiers $\forall$ and $\exists$, equality =, brackets and corresponding to a structure $\mathcal{A}$, the non-logical constants, i.e. relations (with given arities), functions (with given arities) and individual constants.

Equality is added as special relation to the predicate logic formalism; only in second order logic (i.e. allowing quantification over relation symbols) equality of x and y can be expressed within the formalism as $\forall P \; P(x) = P(y)$.

Definition 2.3.17 (Terms) *Terms are defined as the minimal set P which satisfies:*

1. $v_i \in P, c_k \in P$ *(variables and non-logical constants are terms).*
2. *If $f \in \mathcal{L}_{Pred}$ is an n-ary function symbol, and $t_1, \ldots t_n$ are terms, then also $f(t_1, \ldots, t_n) \in P$.*

Terms involving no variables are called ground terms. Ground terms are the elements of Herbrand interpretations (see below), for instance c, $f(g(c))$ are ground terms, but $f(v_3)$ with v_3 being a variable is no ground term.

Definition 2.3.18 (Formulae) *Formulae are defined as the minimal set F_L which satisfies:*

1. *If s, t are terms, then $s = t \in F_L$.*
2. *If $R \in L_{Pred}$ is an n-valued relation symbol, $t_1, \ldots, t_n$ terms, then $R(t_1, \ldots, t_n) \in F_L$.*
3. *If ϕ and $\psi \in F_L$, then $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \Rightarrow \psi \in F_L$.*
4. *If $\phi \in F_L$, then $\forall v_i \; \phi$, $\exists v_i \; \phi \in F_L$.*

Formulae of type (1) and (2) are called atomic formulae. Usually we abbreviate propositional formulae with capital letters and first-order formulae with either greek letters or capital letters; we denote sets of formulae with capital greek letters.

Using a structure $\mathcal{A}$ and an evaluation h, each expression is mapped to a value – terms to individuals and formulae to truth-values.

Definition 2.3.19 (Assignment) *A function h of the set of variables into the individual domain A is called assignment.*

Free variables are variables that are not affected by a quantor; a sentence has no free variables. Hence, for sentences no assignment is needed.

Definition 2.3.20 (Evaluation function) *An evaluation function is an extension of the assignment function: $Eval^h : T \to A$ (of the terms into the individual domain) with*

$$Eval^h(v_i) = h(v_i), Eval^h(c_k) = c_k^{\mathcal{A}},$$

$$Eval^h(f_j(t_1, \ldots, t_n)) = f_j^{\mathcal{A}}(Eval^h t_1, \ldots, Eval^h t_n),$$

with $c_k^{\mathcal{A}}$, $f_j^{\mathcal{A}}$ are the suitable denotations for c_k and f_j in the structure $\mathcal{A}$.

Definition 2.3.21 (Validity) *For an assignment h in $\mathcal{A}$ a validity relation, $\mathcal{A} \models_h \phi$ is inductively defined (with $\phi \in F_L$):*

- *For atomic formulae:*

  1. *$\mathcal{A} \models_h t_1 = t_2$ if $Eval^h(t_1) = Eval^h(t_2)$ (with t_1 and t_2 terms);*
  2. *$\mathcal{A} \models_h R_i(t_1, \ldots, t_n)$ if $R_i^{\mathcal{A}}(Eval^h t_1, \ldots, Eval^h t_n)$ holds.*

- *Logical connectiva and quantors:*

  3. *$\mathcal{A} \models_h \neg\phi$ if $\mathcal{A} \models_h \phi$ does not hold;*
     $\mathcal{A} \models_h \phi \wedge \psi$ if $\mathcal{A} \models_h \phi$ and $\mathcal{A} \models_h \psi$;
     $\mathcal{A} \models_h \phi \vee \psi$ if $\mathcal{A} \models_h \phi$ or $\mathcal{A} \models_h \psi$;
     $\mathcal{A} \models_h \phi \Rightarrow \psi$ if $\mathcal{A} \not\models_h \phi$ or $\mathcal{A} \models_h \psi$;
  4. *$\mathcal{A} \models_h \exists v_i \phi$ if an assignment g exists with $g(v_j) = h(v_j)$ for $j \neq i$ for which $\mathcal{A} \models_g \phi$ holds;*
     $\mathcal{A} \models_h \forall v_i \phi$ if for all assignments g with $g(v_j) = h(v_j)$ for $j \neq i$ holds that $\mathcal{A} \models_g \phi$.

Definition 2.3.22 (Model) *A structure $\mathcal{A}$ is a model of a set of formulae $\theta \subseteq F_L$, written as $\mathcal{A} \models \theta$, if for all assignments h and all formulae $\phi \in \theta$ it holds that $\mathcal{A} \models_h \phi$.*

Definition 2.3.23 (Satisfiability) *A set of formulae $\theta \subseteq F_L$ is satisfiable, if there exists a structure $\mathcal{A}$ and an assignment h in $\mathcal{A}$ such that for all formulae $\phi \in \theta$: $\mathcal{A} \models_h \phi$.*

Tautologies are formulae which are true in all models, and contradictions are formulae which do not have a model.

With respect to the syntactical way, many different formal systems have been introduced (one example would be the Hilbert type calculus of the previous section extended with some axioms for $\forall$ and $\exists$ and with one further rule, the generalisation rule $A \vdash \forall x\, A$). Syntactical notions of derivability and consistency, and the semantic notion of logical consequence are defined in a similar way as in propositional logic.

As stated above, in a suited calculus for predicate logic it holds that $\theta \vdash \phi$ implies that $\theta \models \phi$ (soundness). Additionally, the *completeness theorem of Gödel* reveals that $\theta \models \phi$ implies that $\theta \vdash \phi$:

Theorem 2.3.24 (Completeness Theorem) *There exist sound and complete calculi for $\mathcal{L}_{Pred}$.*

Hence, the syntactical and semantical way coincide in predicate logic. However, in higher order logics like the second-order logic completeness can no longer be achieved.

Based on logic, theories can developed using own axioms, like Peano arithmetics mentioned in Section 2.2.

Two nice theorems are easy corollaries of the above completeness theorem (for instance cf. [Ebbinghaus *et al.*, 1996]):

Theorem 2.3.25 (Löwenheim and Skolem) *Every satisfiable countable set of sentences is satisfiable over a countable set as model.*

Theorem 2.3.26 (Compactness Theorem) *$\Phi \models \phi$ iff there exists a finite $\Phi_0 \subset \Phi$ with $\Phi_0 \models \phi$.*

Due to the compactness theorem a formula is satisfiable if it is satisfiable in a finite structure.

2.3.3 Herbrand models

The set of all ground terms of the predicate logic language is called the Herbrand universe of $\mathcal{L}_{Pred}$, denoted by U_H. It depends on the set of function and constant symbols, but not on the relation symbols.

The set of all ground atomic formulae – these are relations only involving ground terms – form the Herbrand base of $\mathcal{L}_{Pred}$, which is denoted by B_H. The elements of B_H are called ground atoms.

Definition 2.3.27 *A Herbrand interpretation is an interpretation using the individual domain U_H, and mapping each constant symbol and each function symbol into the suited denotation of the algebraic structure.*

By a propositional grounding $ground(F)$ of a universal sentence

$$F = \forall v_1, \ldots \forall v_n \; \psi(v_1, \ldots, v_n)$$

we mean the collection of all formulae resulting from ψ where each variable v_i is substituted with a ground term, ranging over all ground terms in the Herbrand universe.

Universal theories are theories consisting only of universal sentences, i.e. universal quantors followed by a quantifier-free formula; such universal theories are used in logic programming. A one-to-one relationship between Herbrand models of a universal theory and models of the propositional grounding of a universal theory can be shown. Each subset $S_{\mathcal{M}}$ of the Herbrand base B_H determines a Herbrand interpretation $\mathcal{M}_S$ and vice versa:

Theorem 2.3.28 *For every universal theory T, a Herbrand interpretation $\mathcal{M}_S$ is a model of T iff $S_{\mathcal{M}}$ is a model of $ground(T)$.*

Hence, we will think of a subset S of the Herbrand base both as propositional valuation (the propositions are the ground terms) or as Herbrand interpretation.

Therefore, for valuations of propositional theories, we always use Herbrand interpretations, too, consisting of those propositional atoms which are true in the valuation. For instance, given the propositional language $\{p, q, r, x\}$, the interpretation $\{p, r\}$ refers to the valuation in which p and r are true and q and x are false.

2.4 Modal and Epistemic Logics

2.4.1 Modal Logic

There are several possible definitions when to classify a logic as non-classical. One of those is that non-classical logics are logics that do not obey truth-functionality, are multi-valued logics or use another distinguished set of formulae than tautologies. In this sense most kind of modal logics would qualify themselves as classical since they only extend the used language. Other logicians would disagree with this definition of non-classical logic and treat modal logics as non-classical, too.

Modal logic is an extension of classical logic adding operators of necessity and possibility, abbreviated as $\Box$ and $\Diamond$, respectively.

The semantics of modal logic is a kind of possible-world semantics and the models are usually defined in terms of Kripke-structures.

Definition 2.4.1 (Kripke-structure) *A Kripke-structure $\mathcal{S}$ is defined as triple $\langle S, R, v \rangle$ with S being a nonempty set of possible states (worlds), R a binary accessibility relation connecting the states and v an evaluation function defined as follows:*

- *If p is an atom, and s a world, then $v(p, s)$ is the truth value of p in the world s.*
- $v(\neg p, s) = \neg v(p, s)$
- $v(p \odot q, s) = v(p, s) \odot v(q, s)$ *(with $\odot \in \{\wedge, \vee, \Rightarrow\}$)*
- $v(\Box F, s) = \mathbf{t}$ *iff* $\forall t\, sRt : v(F, t) = \mathbf{t}$, *else* $\mathbf{f}$.
- $v(\Diamond F, s) = \mathbf{t}$ *iff* $\exists t\, sRt : v(F, t) = \mathbf{t}$, *else* $\mathbf{f}$.

Depending on the restrictions posed on the accessibility-relation, we distinguish a large number of different modal logics.

For the sake of simplicity, we just consider propositional modal logics. Depending on the requirements the accessibility relation shall fulfill, several modal logics are distinguished.

Informally, being in one of the worlds s, an agent holds $\Box A$ as true exactly if A is true in every world he can think of, i.e. in every world t with sRt. He holds $\Diamond A$ as true exactly if A is true in at least one of the worlds t with sRt. For instance, if the relation is reflexive, then $\Diamond A$ is always the case if A is true. Such ideas lead to a syntactical characterisation of modal logics and to some comparisons: To each requirement posed on a relation corresponds an axiom as given in Table 2.1.

Definition 2.4.2 (Validity) *A modal formula Φ is valid in a particular Kripke-structure $\langle S, R, v\rangle$ iff $v(\Phi, s) = \mathbf{t}$ in each world $s \in S$. A modal formula Φ is valid (is a tautology) in modal logic ML iff it is valid in each Kripke-structure with the accessibility relation R_{ML}.*

Generally, possibility is defined via necessity – as $\neg\Box\neg$. Each normal calculus fulfills the property

$$\Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B) \qquad (K)$$

This property can easily be verified to be valid in each Kripke structure. Usually, a modal logic is named depending which properties the accessibility relation fulfills, resp. which axioms hold. Hence, each axiom is given a unique number or letter. Well, the notations are not unique, but we state the most widely used notation.

For instance, KT5 is also known as S5; since euclidicity and reflexivity imply symmetry and hence transitivity, this relation is an equivalence relation.

In each of these normal calculi the necessitation rule $\vdash A$ implies $\vdash \Box A$ is sound.

Besides the mentioned axioms and calculi, there are many more - most interesting is the rich variety of logics between KT4 (=S4) and S5. For instance, S4.2 is S4 extended with the weak direction property. S4.2 is, due to Lenzen [1980] the most suited logic for representation of knowledge.

reflexive	$\forall s\ sRs$	$\Box A \Rightarrow A$	(T)
symmetric	$\forall s \forall t\ sRt \Rightarrow tRs$	$A \Rightarrow \Box\Diamond A$	(B)
serial	$\forall s \exists t\ sRt$	$\Box A \Rightarrow \Diamond A$	(D)
transitive	$\forall s \forall t \forall u\ sRt \wedge tRu \Rightarrow sRu$	$\Box A \Rightarrow \Box\Box A$	(4)
euclidian	$\forall s \forall t \forall u\ sRt \wedge sRu \Rightarrow tRu$	$\Diamond A \Rightarrow \Box\Diamond A$	(5)
partially functional	$\forall s \forall t \forall u\ sRt \wedge sRu \Rightarrow t = u$	$\Diamond A \Rightarrow \Box A$	
functional	$\forall s \exists! t\ sRt$	$\Box A \Leftrightarrow \Diamond A$	
weakly dense	$\forall s \forall t\ sRt \Rightarrow \exists u\ (sRu \wedge uRt)$	$\Box\Box A \Rightarrow \Box A$	
weakly connected	$\forall s \forall t\ sRt \wedge sRu \Rightarrow tRu \vee uRt \vee u = t$	$\Box(A \wedge \Box A \Rightarrow B) \vee \Box(B \wedge \Box B \Rightarrow A)$	
weakly directed	$\forall s \forall t\ sRt \wedge sRu \Rightarrow \exists v\ (tRv \wedge uRv)$	$\Diamond\Box A \Rightarrow \Box\Diamond A$	

Table 2.1: *Properties of Relations corresponding to Modal Axioms*

2.4.2 Epistemic Logic

In epistemic logic, operators like $G(a, p)$ are describing that agent a believes p. In one-agent-systems the operator can be treated identically to the modal operator of necessity. Observe that if the necessity operator is identified as knowledge operator, then the possibility operator is not identified as belief operator, but as *not knowing that not*. There are different calculi for knowing and believing, and if both notions are treated together two (or more in a multi-agent-system) different necessity modalities are distinguished.

Epistemic sentences can be treated in two different ways, namely *de re* and *de dicto*, based on the problem if you believe in theorems or if you believe that theorems are valid. In predicate logic this difference can be exhibited as follows: consider the proposition "All mathematicians are necessarily rational". How do you translate this sentence into modal logic? There is no unique translation – you could either translate it as:

$$\forall x\ M(x) \Rightarrow \Box R(x) \qquad \text{de re}$$
$$\text{or}$$
$$\Box(\forall x\ M(x) \Rightarrow R(x)) \qquad \text{de dicto}$$

The situation is analogous if we view an operator of belief instead of the possibility operator. These two possibilities refer to different treatments of constants – are they the same in each world? (This is closely related to the notion of rigid designators – for instance, "soccer world champion" is one, but "France" refers to it only in some of the worlds.)

In logic, usually the de dicto view is adopted; however, in epistemic predicate logic the operator of belief is no longer referential autark and looks like $G(a, p(x))$ which yields a lot of problems with substitution. Hence, Sosa and Lenzen [1980] argue that in epistemic predicate logic the de re view should be better explored – there, the operator looks like $G(a, p)(x)$ and could be used for substitution.

Other problems hampering a good syntactical and semantical treatment of epistemic predicate logic are that the operators need not necessarily be extensional. In fact, they even do not need to be intensional:

Extensionality: $G(a, p) \wedge (p \Rightarrow q) \Rightarrow G(a, q)$

Intensionality: $G(a, p) \wedge (\vdash_L p \Rightarrow q) \Rightarrow G(a, q)$

Hyperintensionality: $G(a, p) \wedge G(a, p \Rightarrow q) \Rightarrow G(a, q)$

Extensionality expresses that q follows from p regarding content, intensionality that q follows from p in some formal logic calculus L. Even worse is that hyper-intensionality does not need to be valid. An operator of knowledge certainly fulfills this property, but an operator of weak belief does not necessarily fulfill it. For instance, one may believe very weakly in p, and very weakly in the implication, but the overall belief is too weak (for instance below $\frac{1}{2}$ and the other two ones are above $\frac{1}{2}$ if you want to state it like this). Another cause is that an agent may have forgotten the modus ponens ("how stupid" you might think, but such things happen sometimes . . .), and cannot jump to this conclusion.

Hence, weak belief cannot be presented within a Kripke-structure, as it is easy to see that hyper-intensionality is an instance of axiom (K) with no requirements to the reachability relation and hence forces to use other semantics.

Philosophers ask the question which modal/epistemic system is best suited to represent knowledge and belief. Lenzen distinguished *strong knowledge* and *weak knowledge*: the first being defined as *I know* p means that p is true and I am convinced of p and the second one additionally requires that there is a justified base for p (which is difficult to define anyway).

Weak knowledge is, according to Lenzen and Gettier best represented in modal logic S4.4, this is S4 and the additional axiom of $A \Rightarrow (\Diamond A \Rightarrow \Box A)$ with $\Box$ being the weak knowledge operator.

Lenzen introduced a knowledge, conviction and belief calculus, respectively. The last one does not fulfill hyper-intensionality and requires a semantics different to the usual Kripke models. The calculus of conviction is KD45 – this form of strong belief is in AI mostly used, because it is easier to represent than weaker beliefs.

The property that distinguishes knowledge from belief is reflexity - someone can only know true things, hence $Kp \Rightarrow p$. Due to Lenzen, however, S5 is a too strong system for knowledge, and he proposes S4.2 as given above. Lenzen then combines this system with some axioms where all three operators are used and provides a completeness-proof for this logic [Lenzen, 1980].

For other semantics and other problems in the rich variety of epistemic logics we refer to [Hoek, 1993]; for complexity results refer to [Halpern and Moses, 1992].

In the field of AI, especially the works of Kraus and Lehmann [Kraus *et al.*, 1990], Halpern and Moses [Halpern and Moses, 1992], and Meyer and Hoek [Meyer and Hoek, 1995] are worth mentioning. Among other things, these articles introduce calculi, discuss topics like the operator of common knowledge, and furthermore focus on complexity results.

2.5 Non-monotonic Reasoning

Most logics are monotone in the sense that, given a set of formulae A, and formulae F and G, that:

$A \vdash F$ implies: $A, G \vdash F$.

i.e. adding premises, previous conclusions can never be falsified.

However, non-monotonic reasoning (often the word "reasoning" is used instead of "logic" since the principle of monotonicity is not fulfilled) tries to formalise the knowledge of an agent in a given situation. If new/updated information is gained, previous conclusions can be invalidated.

Example 2.5.1 *Usually the following example is given, using the poor bird Tweety: An agent knows that Tweety is a bird. He concludes that Tweety can fly because birds typically fly. However, he now gains the information that Tweety is a penguin. Since penguins do not fly he has to reevaluate his previous conclusions.*

The notion $\vdash$ is defined in different ways for particular kinds of non-monotonic reasoning (NMR). For instance, in default reasoning, two straightforward ways of reasoning are possible – brave reasoning and cautious reasoning, as defined later on.

Moreover, non-monotonic reasoning is a kind of context-based reasoning – an agent is allowed to reach several sets of conclusions depending on the context. For instance, in default logics, a default theory can admit multiple extensions.

In artificial intelligence, non-monotonic reasoning is widely used, as it implements classical logic for description, and, admits intended and preferred assumptions [Marek and Truszczyński, 1993]. Additionally, some kinds of non-monotonic reasoning allow closed-world reasoning: if nothing is known about f, then infer $\neg f$. Hence, both syntactically and semantically specified non-monotonic reasoning methods prove to be useful in AI to formalise common-sense reasoning (for a general introduction to AI cf. [Charniak and Mc Dermott, 1985]).

Some examples of non-monotonic reasoning are Reiter's default logic and variants thereof, modal non-monotonic logics (in particular, Moore's autoepistemic logic), circumscription, logic programming, closed world reasoning and abduction/diagnosis. We will provide some details about all these logics in the following subsections.

2.5.1 Closed World Assumption and Circumscription

The closed world assumption [Reiter, 1978] assumes that everything which cannot be deduced is assumed false. Say, given a propositional theory (usually we are interested in propositional theories), the closed world assumption is:

$$\text{CWA}(T) = T \cup \{\neg p \mid T \not\vdash p \ \wedge p \in PosLit\},$$

where $PosLit$ is the set of positive literals. However, consider the theory $T = p \vee q$. In this case, CWA(T) is inconsistent, since both $\neg p$ and $\neg q$ are added. Hence, not only reasoning whether CWA(T) $\models F$ holds is an important task, but also determining the complexity whether the CWA closure of a theory is consistent is worth exploring.

Since CWA is not always consistent, several other closed world rules have been introduced in the past, presented in Table 2.2 in a syntactical and in Table 2.3 in a semantical way.

An equivalent alternative characterisation of CWA and other variants is possible with Herbrand models:

Let $MM(T)$ be the minimal models of T with the usual subset-relation (i.e. models which are not a subset of another model of T) and let $MM(T; P, Z)$ be the minimal models of T with $\leq_{P,Z}$. $M \leq_{P,Z} M'$ iff $M \cap Q = M' \cap Q$ and $M \cap P \subseteq M' \cap P$; with $\langle P, Q, Z\rangle$ being a partition of the atoms.

The equivalent characterisation of CWA is: Add $\neg p$ if p is a positive literal and there is some M s.t. $M \not\models p$.

GCWA	GCWA$(T) = \{\neg p \mid T \not\models C \vee p, T \not\models C, p$ is atom$\}$
EGCWA	EGCWA$(T) = \{\neg G \mid T \not\models C \vee G, T \not\models C, G = \bigwedge_i p_i\}$
CCWA	CCWA$(T) = \{\neg p \mid T \not\models B \vee p, T \not\models B, p$ is atom$\}$
ECWA	ECWA$(T) = \{\neg F(P,Q) \mid T \not\models B \vee F(P,Q), T \not\models B\}$

Table 2.2: *Syntactical characterisation of Some Closed World Assumptions*

GCWA	$\forall M \in MM(T), M \not\models p$
EGCWA	$\forall M \in MM(T), M \not\models G$
CCWA	$\forall M \in MM(T; P, Z), M \not\models p$
ECWA	$\forall M \in MM(T; P, Z), M \not\models F(P,Q)$

Table 2.3: *Semantical characterisation of Some Closed World Assumptions*

In Table 2.2, C is a positive clause, whereas B is a clause with literals from $Q \cup P^+$ (P^+ is the set of positive literals of P), p is a positive literal, and $F(P,Q)$ is an arbitrary formula without literals from Z. GCWA is called Generalized CWA, EGCWA Extended GCWA, CCWA is called Careful CWA, ECCWA Extended Careful CWA. Table 2.3 shows the semantical counterpart in the definition of the respective closed world assumption.

Circumscription is based on minimality of models and is a second-order method [Mc Carthy, 1980].

Let $T[P, Z]$ be a first-order formula with P and Z being lists of (disjoint) predicates. Formally, the circumscription $CIRC(T; P; Z)$ of the predicates P with varying predicates Z is the following second-order formula:

$$T[P; Z] \wedge (\forall P'Z'(T[P', Z'] \Rightarrow \neg(P' < P)))$$

where $P' \leq P$ stands for the conjunction of the formulae $\forall x(p_i'(\mathbf{x})) \Rightarrow p_i(\mathbf{x}))$ with p_i being the predicates in P and $\mathbf{x}$ a list of variables. Predicates in P are minimized, predicates not in $P \cup Z$ are fixed.

2.5.2 Logic Based Abduction

Another kind of non-monotonic reasoning is abduction. We are given a theory and a set of hypotheses. Based on manifestations we need to create a solution as a subset of the hypotheses.

Example 2.5.2 *Let*

$$T = \{take\_bus \vee take\_train, take\_bus \Rightarrow too\_late, take\_train \Rightarrow far\_to\_go,$$

$$overslept \Rightarrow too\_late \wedge too\_tired, too\_tired \Rightarrow \neg far\_to\_go\},$$

$$M = \{too\_late\}, H = \{overslept, take\_bus, take\_train\}$$

Then, $S_1 = \{take\_bus\}$, $S_2 = \{overslept\}$ *and* $S_1 \cup S_2$ *are abductive explanations of the manifestations* M*. It makes sense only to allow minimal solutions in some settings to rule out over-specified explanations.*

Definition 2.5.3 *A propositional abduction problem* P *consists of a tuple* $\langle V, H, M, T\rangle$*, where* V *is a finite set of propositional variables,* $H \subseteq V$ *is the set of hypotheses,* $M \subseteq V$ *is the set of manifestations, and* T *is a consistent theory.*

Definition 2.5.4 *Let* P *be a propositional abduction problem. Then,* $S \subseteq H$ *is a solution of* P *iff* $T \cup S$ *is consistent and* $T \cup S \models M$*.*

Several preference/minimality criteria $\leq$ have been introduced and the complexity been analysed in [Eiter and Gottlob, 1995b]. We will present some of these results, as for the other presented logics, too, in Section 4.6. In general, the set of solutions depending on an arbitrary preference criterium $<$ can be defined as follows (where $Sol(P)$ is the set of solutions of P without a minimality criterium).

Definition 2.5.5 $Sol_{\leq}(P) = \{S \in Sol(P) : \neg\exists S' \in Sol(P) : S' < S\}$

The following decision problems deserve to be treated:

- Decide if a given problem has a solution.
- Decide if an hypotheses h occurs in at least one solution (is relevant).
- Decide if an hypotheses h occurs in all solutions (is necessary).

2.5.3 Default Logic

A default theory is a pair $\langle D, W\rangle$ with W being a set of classical (propositional) formulae and D a set of default rules [Reiter, 1980].

Default rules are similar to ordinary inference rules, however, additionally they require a justification to be consistent in order to infer the consequent. Moreover, this justification not only needs to be consistent with the current set of derived formulae, but with a whole set of conclusions an agent can derive.

Whereas usual inference rules look like this:

$$\frac{p(d)}{c(d)}$$

a default rule is of the following form:

$$\frac{p(d) : Mj(d)}{c(d)}$$

where $p(d)$ is called the *prerequisite* of the rule, which is a (propositional) formula, $j(d)$ is the set of *justifications* which is a set of (propositional) formulae β_i, and $c(d)$ the *consequent* of the default rule.

In general, default rules can also contain first-order formulae - a default rule is closed iff it contains no free variables.

The treatment of open default theories is a little bit more involved as the closure of T needs to be formed first.

Example 2.5.6 *Consider the following default theory $\langle D, W\rangle$:*

$$W = \{Hobbit(Frodo), \forall x\ Hobbit(x) \Rightarrow Hobbit((Father(x)))\}$$

$$D = \left\{ \frac{Hobbit(x) : Smokes\_Pipe(x)}{Smokes\_Pipe(x)} \right\}$$

We compute its closure as the infinite set

$$D = \left\{ \frac{Hobbit(Frodo) : Smokes\_Pipe(Frodo)}{Smokes\_Pipe(Frodo)}, \right.$$

$$\left. \frac{Hobbit(Father(Frodo)) : Smokes\_Pipe(Father(Frodo))}{Smokes\_Pipe(Father(Frodo))} \dots \right\}.$$

The semantics of default logic is based on *extensions*, these are sets of total knowledge or belief an agent can assume based on a given default theory. Without using priorities, each of these sets has equal justification to be the knowledge of a particular agent.

Informally, $c(d)$ can be inferred if $p(d)$ had been inferred and all $\neg\beta_i$ with $\beta_i \in j(d)$ cannot be inferred (i.e. $j(d)$ is consistent).

For convenience we define $c(H) = \{c(d) \mid d \in H\}$. If $j(d)$ is a singleton we identify it with its only element, and we normally omit the 'M' in front of the justifications.

In the literature several different equivalent characterisations of extensions have been introduced. Informally, an extension is a set where the prerequisites need to be inferred in an iterative way, the justifications need to be consistent with the extension, every possible default has to be used, and the extension is an infinite set of all consequences.

Another characterisation introduces a context which needs to coincide with the extension – this is the definition given by Reiter; see below for a formal characterisation.

An extension shall fulfill several properties: First of all, it certainly should contain W. Furthermore, it is an infinite set closed under logical consequence. All usable defaults should be used. No formula shall be included which cannot be inferred from W and the used consequences.

With these ideas in mind, let us present Reiter's definition:

Definition 2.5.7 *[Reiter, 1980] Let $\Delta = \langle D, W \rangle$ be a default theory. For any set S of propositional formulas, $\Gamma_\Delta(S)$ is the smallest subset U satisfying the following three properties:*

1. $W \subseteq U$.
2. *U is deductively closed.*
3. *If $(\alpha : M\beta_1, \ldots, M\beta_n)/\omega \in D$, $\alpha \in U$, and $\forall i : \neg\beta_i \notin S$, then $\omega \in U$.*

An extension of $\Delta = \langle D, W \rangle$ is a fixed-point of Γ_Δ, i.e. a set E of propositional formulae satisfying $\Gamma_\Delta(E) = E$.

This fixed-point construction allows to achieve that an extension is minimal in the sense that it does not contain unfounded formulae. Property (3) achieves that an extension is maximal in the sense that a maximum number of applicable defaults is used (therefore, if $F \subseteq E$, then at most one of F or E can be an extension).

There are various other equivalent characterisations of extensions. Here, we use a kind of quasi-inductive characterisation, based on the base operator B^D due to Marek and Truszczyński (see Theorem 3.11 and Examples 3.32 to 3.39 in [Marek and Truszczyński, 1993]; although we use this characterisation, we will not follow exactly the same syntax as Marek and Truszczyński.

Definition 2.5.8 *Let $\Delta = \langle D, W \rangle$ be a default-theory and S be a* context. *Then, the operator B^{D_S} is defined as*

$$B^{D_S}(U) = U \cup \{c(d) \mid p(d) : j(d)/c(d) \in D \;\; with$$
$$U \models p(d) \;\; and \;\; S \not\models \neg\beta_i \;\; for\, all \;\; \beta_i \in j(d)\}\,.$$

Let $B^{D_S} \uparrow \omega$ be the infinite application of the operator B^{D_S} on itself.

Proposition 2.5.9 *[Marek and Truszczyński, 1993] E is an extension of $\Delta = \langle D, W \rangle$ iff $E = B^{D_E} \uparrow \omega(W)$.*

We define $cons(A)$ as usual as $\{F \mid A \models F\}$ (sometimes we drop the brackets if the meaning is clear).

Instead of using the operator B_D we normally use an equivalent formulation based on an iterative process. Let H be a subset of D:

$$E_0 = W,$$

$$E_{k+1} = E_k \cup \left\{ c(d) \;\middle|\; \frac{p(d) : j(d)}{c(d)} \in D, E_k \models p(d), \right.$$
$$\left. \forall \beta_i \in j(d) :\; W \cup c(H) \not\models \neg\beta_i \right\}.$$

As we limit ourselves to finite default theories it can easily be seen that at latest after $|D|$ steps a fixed-point has been reached. $E = cons(E_{|D|})$ is an extension of $\langle D, W \rangle$ iff $E_{|D|} = W \wedge c(H)$.

The set of defaults which generate an extension E is called generating defaults of D, abbreviated as $GD(D, E)$. Every extension is of the form $E = cons(W \cup c(GD(D, E)))$. Additionally, each E_k is of the form $W \cup c(J_k)$ with J_k being subsets of D.

Since extensions are usually infinite sets of formulae, this finite characterisation using E_k is very handy: Let $E_{|D|} \equiv W \wedge c(J)$. All E_k are finite sets; starting with a set $H \subseteq D$, the check if $E_{|D|} \equiv W \wedge c(H)$ reduces to checking if $H = J$.

Definition 2.5.10 (normal/seminormal) *A normal default theory is a default-theory in which $j(d) = c(d)$ for all defaults (recall the convention that we identify $j(d)$ with its only element if $j(d)$ is a singleton). A semi-normal default theory is a theory in which each $j(d)$ is of the form $f(d) \wedge c(d)$ where $f(d)$ are arbitrary propositional formulae.*

Every normal default theory is guaranteed to have at least one extension.

For other undefined concepts w.r.t. to Reiter's default logic we refer the reader to [Reiter, 1980].

Example 2.5.11 *Again we use the poor bird Tweety – now to formalise the example given above:*

$$W = \{bird(Tweety), penguin(Tweety) \Rightarrow \neg flies(Tweety)\}$$

$$D = \left\{ \frac{bird(x) : flies(x)}{flies(x)} \right\}$$

After being provided with new information that Tweety is a penguin, you are no longer allowed to use the default rule and to infer that Tweety can fly.

Example 2.5.12 *Let $W = \emptyset$, $D = \left\{ \frac{:\neg a}{\neg a}, \frac{:a}{a \wedge b}, \frac{b : a \vee c}{\neg c}, \frac{:\neg b}{c} \right\}$*

In Table 2.4 we analyse each subset S' of $c(D)$ (W is empty); $S = cons(S')$ is then the context; eliminate the defaults where the justification is not consistent with the context and look if the remaining rule system allows to infer exactly the elements of S'. If yes, then this context is the base of an extension (marked with $\surd$), i.e. cons S' forms an extension. Of course, only sets of consequences need to be taken into account as base of contexts; moreover, inconsistent context bases can be disregarded, and subsets or supersets of already determined extensions do not need to be considered.

S'	Rules	$\Gamma_\Delta(S')$
$\emptyset$	D	$\mathbf{f}$
$\{\neg a\}$	$\frac{}{\neg a}, \frac{b}{\neg c}, \frac{}{c}$	$\{\neg a, c\}$
$\{a \wedge b\}$	$\frac{}{a \wedge b}, \frac{b}{\neg c}$	$\{a, b, \neg c\}$
$\{\neg c\}$	$\frac{}{\neg a}, \frac{}{a \wedge b}, \frac{b}{\neg c}, \frac{}{c}$	$\mathbf{f}$
$\{c\}$	$\frac{}{\neg a}, \frac{}{a \wedge b}, \frac{b}{\neg c}, \frac{}{c}$	$\mathbf{f}$
$\{\neg a, a \wedge b\}$	$\emptyset$	$\emptyset$
$\{\neg a, \neg c\}$	$\frac{}{\neg a}, \frac{b}{\neg c}, \frac{}{c}$	$\{\neg a, c\}$
$\{\neg a, c\}$	$\frac{}{\neg a}, \frac{b}{\neg c}, \frac{}{c}$	$\{\neg a, c\} \surd$
$\{a \wedge b, \neg c\}$	$\frac{}{a \wedge b}, \frac{b}{\neg c}$	$\{a \wedge b, \neg c\} \surd$
$\{a \wedge b, c\}$	$\frac{}{a \wedge b}, \frac{b}{\neg c},$	$\{a \wedge b, \neg c\}$
$\{\neg c, c\}$	$\emptyset$	$\emptyset$
$\{\neg a, \neg c, c\}$	$\emptyset$	$\emptyset$
$\{\neg a, a \wedge b, \neg c\}$	$\emptyset$	$\emptyset$
$\{\neg a, a \wedge b, c\}$	$\emptyset$	$\emptyset$
$\{a \wedge b, \neg c, c\}$	$\emptyset$	$\emptyset$
$c(D)$	$\emptyset$	$\emptyset$

Table 2.4: *Computing Extensions*

Example 2.5.13 *Consider the following default theory:* $W = \{computer\}$,

$$D = \left\{ \frac{computer : \neg intelligent}{\neg intelligent}, \right.$$

$$\left. \frac{computer : fast}{fast}, \frac{fast : \neg computer \wedge intelligent}{intelligent} \right\}$$

The easiest algorithm is to analyse all subsets of $c(D)$ *(conjuncted with* W*) and looks if the set of conclusions using this context coincides with the context. However, intuition allows us to drop some cases: The third default for instance will never fire, as computer is element of* W *and hence* $\neg computer$ *is never part of a context consistent with* W*. The first and second default both fire and it is easy to see that the context of* $\{computer, \neg intelligent, fast\}$ *creates the only extension of this default theory.*

Example 2.5.14 $W = \emptyset, D = \left\{\frac{:a}{a}\right\}$ *is an easy default theory which is normal. The default theory* $W = \emptyset$, $D = \left\{\frac{a:b}{b}, \frac{:\neg c}{c}\right\}$ *has no extensions since the second default has to be applied and forces a kind of a vicious circle.*

2.5.4 Variants of Default Logic

Several variants of default logic have been proposed; here we will introduce three of them: default theories with weak extensions, default theories with stationary extensions, and disjunctive default theories. Another interesting variant not presented here is cumulative default logic [Brewka, 1991] (complexity results for cumulative default logic can be found in [Gottlob and Mingyi, 1994]).

Whereas the first two kind of theories do not extend the syntax of the rules, but only use a different method to calculate extensions, the third kind modifies the rules, too, and uses an extended concept of standard extensions.

Definition 2.5.15 *[Marek and Truszczyński, 1993] Let $\langle D, W\rangle$ be a default theory and H be a subset of D.*

$$E = cons(W \cup \{c(d) \mid d \in D, W \wedge c(H) \models p(d),$$
$$W \wedge c(H) \not\models \neg\beta_i \quad \forall \beta_i \in j(d)\})$$

E is called a weak extension *of a default theory $\langle D, W\rangle$ iff $E = cons(W \wedge c(H))$.*

The base of an extension E is $W \wedge c(GD(D,E))$ where $GD(D,E)$ is the set of generating defaults of E. Hence, E is an extension if $H = GD(E,D)$.

Weak extensions are not minimal while extensions are minimal, i.e. obey that no extension is a subset of any other extension. However, also default theories with weak extensions may have no extension. As it is easy to see, in the prerequisite-free case, both notions coincide. Furthermore, each extension is a weak extension.

Of course there is also a large variety of alternative characterisations. One is that you initially start with a context $S \subseteq W \cup c(D)$, form the set of all rules consistent with this context and check for each rule if $S \models p(d)$ iff $c(d) \in S$, and furthermore check if each element of S occurs at least once as consequent of those rules where $S \models p(d)$. If both requirements are fulfilled, S forms a weak extension. Refer to Section 6.1.3 for a proof that these characterisations are equal (there discussed in a broader setting extending the concept of weak extensions to disjunctive default theories).

Weak extensions treat prerequisites similar to AEL (Section 2.5.5) and hence these two concepts are closely related.

Example 2.5.16 *Consider the theory $W = \emptyset$, $D = \left\{\frac{a:b}{b}, \frac{b:a}{a}, \frac{:\neg b}{b}\right\}$. This theory has no extension, since the last default forces a vicious circle, and none of the others can be applied instead. However, $\{a, b\}$ forms a weak extension as easily can be checked with the two characterisations of extensions we mentioned. This context blocks the final default, and allows a and b to be "inferred".*

However, in general, weak extensions still suffer the disadvantage that there exist theories which have no extensions. Hence, other concepts of default reasoning have been introduced. One of them is stationary default logic, which reflects the same ideas as well-founded logic programming (Section 2.5.7), whereas default logic is close to logic programming with the stable semantics. A default theory always has at least one stationary extension.

Definition 2.5.17 *[Przymusinska and Przymusinski, 1995] Let $\langle D, W\rangle$ be a default theory. S is called a stationary extension of $\langle D, W\rangle$ iff $E \subseteq \Gamma_\Delta(E)$ and $E = \Gamma_\Delta(\Gamma_\Delta(E))$ with Γ_Δ as in Definition 2.5.7.*

Gottlob [1995] gives an equivalent finitary characterisation (finitary characterisations are helpful for complexity theoretic issues as it can be seen in later chapters). Stationary extensions capture the concept of a smallest extension, which is contained in each extension. In Section 6.3 we will make implicit use of this finitary characterisation as it is more useful for complexity issues. Stationary theories have the advantage of always having an extension, regardless if they are normal or not. Each extension is furthermore a stationary extension, too. Additionally, stationary default logic is cumulatively monotonic (cf. Section 2.5.8).

We now illustrate the intuition behind stationary extensions with a few examples:

Example 2.5.18 $W = \emptyset$,

$$D = \left\{ \frac{: \neg b \wedge c}{c}, \frac{: \neg c \wedge b}{b}, \frac{: a}{a} \right\}.$$

This theory has two extensions, namely $cons\{a, b\}$ and $cons\{a, c\}$, respectively. However, it has one more stationary extension: $cons\{a\}$, because

$$\Gamma_\Delta(cons\{a\}) = cons\{a, b, c\},$$

and

$$\Gamma_\Delta(cons\{a, b, c\}) = cons\{a\}.$$

This is the smallest extension which contains exactly those consequents which are in each extension. $cons(\emptyset)$ is not a stationary extension, because $\Gamma_\Delta(cons\ \emptyset) = cons\{a, b, c\}$ and because $\Gamma_\Delta(cons\{a, b, c\}) = cons\{a\}$.

Example 2.5.19 *However, as soon as a default theory admits at least two extensions which are inconsistent with each other, the smallest stationary extension is $cons(W)$. Since each normal default theory with more than one extension always fulfills this requirement, each normal default theory with more than one extension has $cons(W)$ as smallest stationary extension.*

Consider the theory $W = \emptyset, D = \left\{\frac{:q}{q}, \frac{:p}{p}, \frac{:\neg p}{\neg p}\right\}$. Intuitively, maybe you would like to have $cons\{q\}$ as smallest stationary extension. However, this is not the right logic for this. In stationary default logic, as soon as there are extensions inconsistent with each other, the smallest extension is the empty set (which is the intersection of two inconsistent theories): $\Gamma_\Delta(cons\ \emptyset) = cons\ \mathbf{f}$, and $\Gamma_\Delta(cons\ \mathbf{f}) = cons\ \emptyset$.

However, some different semantics, the well-founded semantics of default logic due to Brewka and Gottlob [1997], allow to obtain $cons\{q\}$ as extension of this example (but suffer from a higher complexity of reasoning).

Another variant of default logic is disjunctive default logic [Gelfond *et al.*, 1991]. This logic does not change the principle of what are extensions, but extends the syntax and extends the concept of extensions to theories with an additional effective disjunction in the consequents. Defaults are allowed to be of the form

$$\frac{p(d) : j(d)}{c_1(d)|\ldots|c_n(d)}.$$

In this section we will only give one example for illustration, and define disjunctive default logic in a more formal way in Section 6.1. However, observe that this is an issue of representation. If you choose a bad representation within a logic which might be suited for representing the problem in a better way, then not the kind of logic you use is bad in all the cases. Hence, you might find representations of problems where disjunctive default logic is not very intuitive, too.

Although we present an example showing the advantages of disjunctive default logic observe that the real motivation came from generalising default logic like disjunctive logic programming was generalised to extended disjunctive logic programming.

Example 2.5.20 *Sometimes standard default logic behaves too weak, sometimes it is too strong: Let $W = \{\mathit{professor} \vee \mathit{student}\}$ and $D = \left\{\frac{\mathit{professor}:sleep\_long}{sleep\_long}, \frac{\mathit{student}:sleep\_long}{sleep\_long}\right\}$.*

We really would like to conclude $sleep\_long$ but it is not possible. Default logic has some difficulties with treating disjunctive information, here too weak, and in the following too strong:

$$W = \{broken(\mathit{leftarm}) \vee broken(\mathit{rightarm})\},$$

$$D = \left\{\frac{: usable(x) \wedge \neg broken(x)}{usable(x)}\right\}.$$

In this case the only extension contains both

$$\{usable(\mathit{leftarm}), usable(\mathit{rightarm})\}$$

which is not quite what we wanted.

In disjunctive default logic, in the first example professor $\vee$ student can be replaced with

$$\mathit{professor}|\mathit{student}$$

(to be exact, with the default $\frac{:\mathrm{t}}{\mathit{professor}|\mathit{student}}$), admitting due to minimality requirements cons$\{\mathit{professor}, \mathit{sleep_long}\}$ and cons$\{\mathit{student}, \mathit{sleep_long}\}$ as extensions.

In the second example,

$$cons\{broken(\mathit{leftarm}), usable(\mathit{rightarm})\}$$

and

$$cons\{broken(\mathit{rightarm}), usable(\mathit{leftarm})\}$$

are the two extensions when using

$$broken(\mathit{leftarm})|broken(\mathit{rightarm}).$$

2.5.5 Autoepistemic Logic

Instead of using justifications to express "possibility" (consistency) one can use modalities as Moore [1985] did. One modal operator K extends the propositional language in a natural way – the classical consequence relation $\models$ is used as usual with $\mathcal{L}_K$ being the extended language treating the surface occurrences of the modal operator as new atoms.

Formally, $\Delta = \{\alpha \mid T \cup K\Delta \cup \neg K\overline{\Delta} \models \alpha\}$ where T is a set of modal premises [Marek and Truszczyński, 1991]. A Δ fulfilling this equation is a stable expansion, with $K\Delta = \{K\beta \mid \beta \in \Delta\}$ and $\neg K\overline{\Delta} = \{\neg K\beta | \beta \notin \Delta\}$.

Informally, expansions are constructed due to some kind of non-monotonic positive and negative introspection. Niemelä [1991] gave an equivalent finitary characterisation of stable expansions useful for complexity considerations.

In Chapter 7 a multi-valued AEL logic based on three-valued propositional logic will be presented which was introduced by Denecker *et al.* [1998] and we derive new complexity results for it. In three-valued logic another truth value is introduced which is referred to as $\mathbf{u}$ (undefined).

Example 2.5.21 *[Konolige, 1988] Let $\Sigma = Kp \Rightarrow p$ be a modal theory. Then an agent can reach the following stable expansions, one including $\neg Kp$, and the other one including both Kp and p. "Including" because the expansion is an infinite set including all knowledge gained with negative and positive introspection. However, one may argue that the second expansion is counterintuitive, since the belief of p is solely based on the assumption that p is believed (Observe that K is rather interpreted as a belief operator than a knowledge operator).*

This drawback of counterintuitive expansions is solved for instance in the non-monotonic logic **N** *(Section 2.5.6). However, default logic can be embedded into* **N***, and we know from the previous sections that default logic also suffers some drawbacks, and some of these counterintuitive problems are not discovered in AEL. So none of them is in general "better" than the other one.*

Due to Moore [1990] stable expansions can be equally characterised in terms of possible world structures, that is, non-empty sets of two-valued interpretations.

All formulae in an expansion without occurrence of a K are called objective formulae. The union of all such formulae forms the objective part of an expansion.

Each stable expansion of a modal theory T has a unique objective part. For each propositional formula F, either KF or $\neg KF$ is in each expansion. This is no longer the case in the three valued variant of [Denecker *et al.*, 1998] (see Chapter 7). Furthermore, to each objective formula exactly one stable expansion refers, namely the stable set [Moore, 1985] of the formula, which is created with deductive closure, positive and negative introspection ($\phi \in T \Rightarrow K\phi \in T, \phi \notin T \Rightarrow \neg K\phi \in T$).

Example 2.5.22 *Let $T = \neg Kp \Rightarrow q \wedge \neg Kq \Rightarrow p \wedge \neg K\neg p \Rightarrow \neg q \wedge \neg K\neg q \Rightarrow \neg p$. For instance try $\{p \wedge q\}$ as base of an expansion. This yields*

$$cons\{p, q\} = cons\{\neg Kp \Rightarrow q \wedge \neg Kq \Rightarrow p \wedge \neg K\neg p \Rightarrow \neg q$$

$$\wedge \neg K \neg q \Rightarrow \neg p \wedge Kp \wedge Kq \wedge \neg K \neg p \wedge \neg K \neg q \wedge \ldots \}$$

Due to $\neg K \neg p$, $\neg q$ is implied, and with the same reasoning $\neg p$. Hence, $\{p, q\}$ cannot be an objective part of an expansion, and due to symmetry reasons no other set out of $\{p, \neg q\}$, $\{\neg p, q\}$ and $\{\neg p, \neg q\}$ can be one. Furthermore, the objective sets $\{p\}$, $\{q\}$, $\{\neg p\}$ and $\{\neg q\}$ are no objective base of an expansion, since for instance for p, $\neg q$ can be inferred. Moreover, the empty objective set forms no expansion, too, because $\neg K \overline{\Delta}$ for instance includes $\neg Kp$, and hence q. Therefore, only the inconsistent objective set remains, allowing for $Kp \wedge K \neg p \wedge Kq \wedge K \neg q$, implying $p \wedge q \wedge \neg p \wedge \neg q$, which is inconsistent, too. Hence, this autoepistemic theory has one stable expansion, namely the one with the inconsistent objective set, since it fulfills the fixed-point equation.

Due to this fixed-point equations autoepistemic theories may have inconsistent expansions. However, there are autoepistemic theories which have no expansions at all, like $T_1 = Kp$. Of course some autoepistemic theories also do not allow the inconsistent expansion – like $T_2 = p \Leftrightarrow Kp$. Consider $\Delta = cons(p \wedge \neg p)$: In both T_1 and T_2, due to positive introspection, one gets $Kp \wedge K \neg p$ on the right-hand side, however the left-hand side of the equation only allows to infer p in T_1, and no objective formula in T_2, hence is not propositionally inconsistent and these two theories both have no inconsistent expansion (indeed, T_1 does not have an expansion at all).

The objective part of stable expansions is closely related to weak extensions of a default theory, indeed each default theory can be embedded into an autoepistemic theory admitting exactly those stable expansions whose objective parts are weak extensions (Section 4.9). However, due to slightly other definitions (in weak extensions starting with an inconsistent set $\mathbf{f}$ never allows to use any default since $\mathbf{f} \not\models \neg j(d)$ never holds) only consistent expansions and weak extensions are compared with each other.

Due to our definition of extensions and weak extensions, default theories only have inconsistent extensions if W is inconsistent (starting with an inconsistent context never allows to use any default, since the justifications are never consistent with an inconsistent context.). However, some care is necessary. Some authors allow for justification-free defaults (= rules). These justification-free defaults are different from defaults with $\mathbf{t}$ as justification! Allowing justification-free defaults of course allows for inconsistent extensions even if W is not inconsistent since no justification checking occurs (the definitions of extensions have to be refined for justification-free defaults simply with dropping the justification checking part for these defaults).

Example 2.5.23 *Let $W = \{p\}$, $D_1 = \left\{\frac{:\mathbf{t}}{\neg p}\right\}$ and $D_2 = \left\{\frac{}{\neg p}\right\}$. Using the quasi-inductive base definition subsequent to Definition 2.5.7 the theory $\langle D_1, W\rangle$ has no extension, whereas $\langle D_2, W\rangle$ has the inconsistent extension.*

However, referring to Section 4.9, the corresponding AEL theories are $T_1 = p \wedge \neg p$ and $T_2 = p \wedge (\neg K\mathbf{f} \Rightarrow \neg p)$ (since these default theories are prerequisite-free extensions coincide with weak extensions, hence we can speak of corresponding AEL theory here). Both have the inconsistent stable expansion due to a different fixed-point construction. Although this is only a minor matter, one needs to take care of these special cases in translatability issues.

As objective parts of stable expansions correspond to weak extensions, objective parts are not minimal and may contain each other like weak extensions may contain other weak extensions. Therefore, the notion of parsimonious expansions was introduced; these are expansions whose objective part does not contain the objective part of any other expansion. However, as Eiter and Gottlob [1992] showed, reasoning in AEL with parsimonious expansions is one level harder than reasoning in AEL with stable expansions.

2.5.6 Modal Non-monotonic Logics

Using a modal inference relation $\models_S$ instead of the usual one like in AEL and using only negative non-monotonic introspection several non-monotonic modal logics had been defined. For details, refer to [Marek and Truszczyński, 1993] and for details concerning modal logic to Section 2.4.

Definition 2.5.24 *Let $T \subseteq \mathcal{L}_K$ be a set of premises where L_K is the propositional language extended with the modal operator K (treating formulae with K as new atoms) and S be a modal logic. A set $\Delta \subseteq \mathcal{L}_K$ is called an S-expansion iff*

$$\Delta = cons_S(T \cup \neg K\overline{\Delta})$$

where $cons_S$ is the consequence operator of the modal logic S.

In non-monotonic modal logics, non-monotonic introspection is just limited to negative introspection. The non-monotonic version of modal logic S is usually abbreviated as $NM(S)$.

- $S = N$, the propositional logic extended with the necessitation rule, forms $NM(S)$, which is the well known logic **N**.
- $S = PL$ means using just the propositional consequence, and equals the logic $NM1$ of [Mc Dermott and Doyle, 1980; 1982].
- S5 coincides with $NM(\text{S5})$.
- Autoepistemic logic, with the exception of some trivial cases, coincides with $NM(\text{KD45})$.

2.5.7 Logic Programming

The non-monotonicity of logic programming yields from the fact that negation is treated as negation as failure, i.e. if the fact cannot be derived then its negation can be inferred because nothing is known about this fact. This kind of negation is also used in Prolog, and of course in logic programming languages like Datalog. Moreover, extended semantics introduces the classical negation additionally (like in the answer set semantics).

Definition 2.5.25 *(Program Clause) A* logic program clause *is an expression of the form*

$$a \leftarrow b_1, b_2, \ldots, b_k,\ \textit{not}\ b_{k+1}, \ldots\ \textit{not}\ b_m$$

where a *and* b_j *are atoms.*

The intuitive meaning is that if the b_j, $j \leq k$ are derived and none of the b_j, $j > k$, can be derived, then a is true for all variables occurring in the predicates. Hence, this negation is not the classical negation. Moreover, in semantics like the answer set semantics this negation-by-failure appears together with the classical negation.

Everything left of the arrow is called the head of the clause, the conjunctions on the right-hand side are called the body.

A logic program P is a collection of program clauses. Usually, logic programming is a first-order formalism – with a and b_j atomic formulae like $R(v_3, f(c_1))$ allowing function symbols and variables. To get decidable logic programs one needs to work without function symbols. A logic program clause is implicitly universally quantified, i.e. the clause $p(x) \leftarrow q(x, c)$ with variable x means $\forall x\ p(x) \leftarrow q(x, c)$. For example, given the facts $q(d, c)$ and $q(e, c)$, infer $p(d)$ and $p(e)$.

Here we are mainly interested in the propositional grounding of a logic program (refer to Section 4.8 for some more details) – substituting each possible value of the variables in the clauses such that they only contain ground atomic relations which equally can be viewed as propositional symbols.

While default logic is usually used in a propositional way, logic programming is a first-order formalism, and circumscription a second order formalism [Marek and Truszczyński, 1993].

$ground(P)$ denotes the logic program where all clauses are transformed into ground clauses. Since ground clauses involve no variables, each atomic formula can be treated as own propositional symbol, hence the question of the complexity of reasoning in $ground(P)$ (which may be exponentially longer than P!) is the same as reasoning in a propositional logic program.

In *disjunctive logic programming* a disjunction $a_1 \vee \cdots \vee a_n$ is allowed to occur in each head of a rule. We mention some words about the respective complexity issues in Section 4.8 and refer to [Marek and Truszczyński, 1993; Dantsin *et al.*, 1997; Eiter and Gottlob, 1995b; Eiter and Veith, 1999] for overviews and several complexity results.

The *stable model semantics* for logic programs, and the disjunctive stable model semantics for disjunctive logic programs is defined as follows.

Definition 2.5.26 *Given a DLP* P *and an interpretation* I*, the Gelfond-Lifschitz transformation* P^I *of* P *w.r.t.* I *is defined as follows.*

1. *If* $a_1 \vee \ldots \vee a_n \leftarrow b_1, \ldots, b_k,\ \textit{not}\ b_{k+1}, \ldots,\ \textit{not}\ b_m \in P$ *and for all* $k < i \leq m$, $b_i \notin I$*, then* $a_1 \vee \ldots \vee a_n \leftarrow b_1, \ldots, b_k \in P^I$.
2. *Nothing else is in* P^I.

P^I is treated as a collection of classical Horn clauses, hence M is a model of P^I in the usual sense of propositional interpretations.

An interpretation I is called a disjunctive stable model of P iff $I \in MM(P^I)$, i.e. among the minimal models of P^I with the usual set-theoretic notion of minimality.

The reader probably detects some similarities to previous definitions for instance with default logic extensions - models of logic programs are context-based!

For a definite logic program, i.e. one where all the heads consist of single atoms, there is a unique minimal model, hence complexity tasks are easier due to the fact that P^I is a collection of Horn clauses.

The set of models of P, $S(P)$, is a subset of $M(P)$, of all models of P if *not* is treated in a classical way. P is called S-consistent iff $S(P) \neq \emptyset$. A DLP P entails a propositional formula f under semantics S (for instance under the disjunctive stable semantics where $S = \{I : I \in MM(P^I)\}$) iff $S(P) \vdash f$, i.e. if F is true in all models in $S(P)$ (cautious reasoning). See Section 4.1 for details on cautious reasoning and Section 4.8 for an introduction of other semantics presented together with some complexity results. For translation issues of non-monotonic formalisms into logic programming, refer to Section 4.9 and Section 5.8. For some details about the relationship of answer set semantics to disjunctive default logic, cf. [Eiter and Gottlob, 1995] and Section 6.1.

Whereas in autoepistemic logic and default logic we had an explicit concept of extensions, such a concept is missing in other non-monotonic formalisms. However, implicitely there is such a concept in almost every formalism – for instance, in logic programming, the Herbrand models $\mathcal{M}$ of a logic program can be viewed as extensions with $E = \{p \mid p \in \mathcal{M}\} \cup \{\neg p \mid p \notin \mathcal{M}\}$ – then a comparison to other formalisms is possible. An alternative approach is not to compare extensions/expansions, but models (and posing some additional requirements on a translation since just comparing models is a concept allowing to compare almost everything to everything).

2.5.8 General Properties of Non-monotonic Reasoning

To discuss general properties of non-monotonic reasoning an entailment relation $\models_{nm}$ has to be considered, as was already implied in the previous section and which will be dealt with again in Chapter 4. We use an entailment relation of *cautious reasoning* – a formula can be deduced exactly if it can be found in all extensions (Section 4.1).

Some of the requirements we would like to pose on such a relation are reflexivity, cut and cumulative monotonicity (see [Kraus *et al.*, 1990] for a discussion about this topic and about general properties of non-monotonic reasoning).

- Reflexivity: $f \models_{nm} f$
  For instance, in default logic this property is fulfilled: $\langle D, f\rangle \models f$, the same in AEL, since every subset of T is always contained in Δ (be careful if the theory has no extension/expansion: cautious reasoning is defined as $\forall E,\ E \models f$. In a theory with no extensions

this is trivially fulfilled, hence such a theory is like **f** in the classical case, that everything can be deduced, so reflexivity is also valid in these cases!)

- Cut: $f \wedge g \models_{nm} h$ and $f \models_{nm} g$ implies $f \models_{nm} h$
  Given a default theory $\langle D, W_1\rangle$ for which an extension W_2 is added to W_1 without inferring any new information forces that if $\langle D, W_1 \wedge W_2\rangle \models_{nm} h$ then $\langle D, W_1\rangle \models_{nm} h$; hence this property is fulfilled.

- Cumulative monotonicity: $f \models_{nm} g$ and $f \models_{nm} h$ implies $f \wedge g \models_{nm} h$
  Although this property of the meta-predicate $\models_{nm}$ sounds fine, it is not satisfied in default logic [Makinson, 1989] as the example below demonstrates. However, stationary default logic for instance exhibits this property!

Example 2.5.27 *Let $W_1 = \{c\}$, $W_2 = \{a \vee b\}$, $W_3 = \{a\}$. Let $D = \{\frac{c:a}{a}, \frac{a\vee b:\neg a}{\neg a}\}$.*
$\langle D, W_1\rangle \models_{nm} a \vee b$ (since the only extension is $cons\{c, a\}$) and $\langle D, W_1\rangle \models_{nm} a$, however $\langle D, W_1 \wedge W_2\rangle \not\models_{nm} a$, because this theory has two extensions, namely $cons\{c, a\}$ and $cons\{\neg a, b, c\}$.

Now consider the same example in weak and stationary default logic: In weak default logic the weak extensions are exactly the same, in the first case only $cons\{a, c\}$, in the second one another extension, $cons\{\neg a, b, c\}$, exists. Hence, weak default theory suffers the same drawback as default theory, and due to translatability issues the same holds true for autoepistemic logic.

However, in stationary default logic the theory $\langle D, W_1\rangle$ has cons $(\emptyset)$ as smallest extension (and everything in the smallest extension can be cautiously deduced and vice versa). Start with the empty set, Γ^2_Δ(cons $(\emptyset)$) = cons $\emptyset$. The theory $\langle D, W_2\rangle$ also admits cons $\emptyset$ as smallest extension, hence (at least this example) demonstrates that stationary default logic is cumulatively monotonic – and it is well known, that this does not hold only for this example [Przymusinska and Przymusinski, 1995].

Rational monotonicity is another nice property: $f \models_{nm} g$ and $f \not\models_{nm} \neg h$, then $f \cup h \models_{nm} g$. However, one usually does not treat this as a property that non-monotonic reasoning should fulfill. Reiter's logic does not fulfill it: Consider the theory $D = \left\{\frac{:\neg b}{a}\right\}$. It has the only extension $cons\{a\}$, and it cannot be inferred that $\{\neg b\}$ holds. However, adding to the empty W the formula b, a can no longer be inferred.

To conclude with one more property, let us state the property (Or) $f \cup g \models_{nm} h$, $f \cup j \models_{nm} h$, then $f \cup \{g \vee j\} \models_{nm} h$ (for more properties and abstract thoughts about non-monotonic, preferential and cumulative reasoning refer to [Kraus *et al.*, 1990]). Default logic has difficulties treating disjunctive information as already mentioned in Section 2.5.3 – there you can also find a counterexample of a theory not fulfilling (Or). Using standard disjunction, this problem remains in disjunctive default logic, too. However, using the effective disjunction instead (if you allow it in W as special case) disjunctive default logic satisfies (Or); see Section 6.1 for some details.

Before diving into the complexity of some forms of non-monotonic reasoning and discussing related issues in Chapter 4 and the subsequent chapters, let us proceed with some basics of complexity theory next.

Chapter 3

Complexity Theory

3.1 Introduction

Complexity theory [Garey and Johnson, 1979; Papadimitriou, 1993] is the little brother of recursion theory, however it is far from being cute like a child, rather hideous like a troll as many problems like the issue whether $\mathbf{P} = \mathbf{NP}$ still have remained open until today.

Logicians usually stop at the point when a problem turns out to be computable, are bored that it is that easy and not for instance in some higher class of the arithmetical hierarchy, and consider other problems instead. Computer scientists, however, are first of all happy, but after initial celebrations and closer looks they often are disappointed - finding a solution to their nice problem which is computable needs far more time than the age of the universe. Hence, they try to find special cases and limitations where this problem is easier to solve.

For such matters it is quite useful to introduce abstract notions of complexity classes with some goals in mind such as that the class should be independent of the used machine model (like one-tape Turing machine, many-tape Turing machine, RAM, . . .).

First of all, we need to distinguish problems and problem instances. An instance is a particular input to a generally stated problem. For instance, the problem may be: *Is a given propositional formula satisfiable* and the special instance is the formula $(a \wedge b) \vee (\neg a \wedge \neg c)$.

There are several abstraction levels how to classify problems w.r.t. complexity - in all of them complexity is treated as a measure of used computer resources, like space and time.

One can analyse the cost of a particular algorithm - like Quicksort, or derive properties of the best algorithm solving the problem. Instead of space and time which are usually used, *axiomatic complexity theory* uses abstract resources, like how much ink is needed to solve a problem.

One of the main goals in theoretical computer science is to find tractable cases. Which problems do we consider to be tractable? Consider a problem where the answer time grows linear or quadratically with the input length, and one where time grows exponentially - refer to Table 3.1 ([Garey and Johnson, 1979]).

	1	2	5	10	20	100
n	1	2	5	10	20	100
n^2	1	4	25	100	400	10000
2^n	1	4	32	1024	1048576	$1.26 \cdot 10^{30}$

Table 3.1: *Polynomial vs. Exponential Growth*

As it is easy to see, an exponential function supersedes a polynomial one, regardless how high the exponent of the polynomial is, sooner or later. Hence, if the input length is doubled things do not turn terrible for a polynomial, but they grow in a horrible way for large input lengths in case of exponential growth.

This distinction between polynomial and exponential growth leads to a classification of tractable and intractable problems. All polynomial problems are considered tractable, while the problems with higher growth, like exponential or worse, are intractable.

However, it may be the case, that for a particular problem, we only know an exponential algorithm, but there is maybe a polynomial one, which we do not know of.

Consider the traveling salesman problem (see below), or one of the most famous problems - satisfiability (SAT) of a propositional formula.

The straightforward way is to try all truth-values of all variables, however these are 2^n possibilities, where n is the number of variables.

Are there easier methods? Indeed, there are, however none that breaks through the exponential barrier. So far every algorithm is exponential, and the question if we can find an easier one is tied to one of the fundamental questions in theoretical computer science, namely if $\mathbf{P} = \mathbf{NP}$.

Initially, in the thirties, logicians were interested in computability, however, issues changed very fast after implementations on real computers were considered. Many tractable problems turned out to be very hard, hyper-exponential and worse.

Computer scientists started to classify problems w.r.t. their hardness. Many different types of such notions were introduced. The one we are interested is a kind of worst-case complexity on a very abstract level. However, this abstract level turns out to allow us to introduce classes of equivalent hard problems and gives us a mechanism to classify new problems. This field of complexity theory is also known as *structural complexity theory*. Problems are gathered into complexity classes by their inherent complexity.

We mainly focus on decision problems, i.e. problems only requiring a boolean answer (*yes/no*). However, search problems can be treated in a very similar way, or by reformulating them as questions if an answer lesser than or greater than some value is possible (and then applying binary search to get the exact value).

We need an abstract notion of input length and computation, and one of the best suited models is, as in recursion theory, the Turing machine [Turing, 1937]. It is old, it is slow, it is not a real machine, but an abstract model - however, it is a machine which can compute every computable function and is hence a model equal to the μ-recursive functions or the λ-calculus (cf. [Oberschelp, 1993]). Due to the Church-Turing thesis, even the best and fastest and most powerful computer today can at most compute the computable functions (hence the intuitive name thereof). This shows that the notion of computability is independent of the used model, unlike notions like provability and so on.

3.2 The Turing Machine

The Turing machine is a model of computation which has been proved to be equivalent to other models of computation (recall the Church-Turing thesis that all models of computation yield the same notion, i.e. computability is independent of definitions). Initially, the Turing machine was introduced by Alan Turing to simulate human reasoning processes.

A Turing Machine (from now on, TM) consists of a potential infinite tape (at least in one direction), a read/write head, an alphabet, and a number of states the machine can be. Formally:

Definition 3.2.1 (Alphabet) *An alphabet is a finite set of symbols. We usually denote it with Σ. Σ^* denotes the set of all finite strings of elements of Σ, including the empty string ϵ.*

Definition 3.2.2 (Language and Classes) *A set $L \subseteq \Sigma^*$ is called language. The complement language $\overline{L}$ consists of all strings not belonging to L. A class is a set of languages; given a class C, its complement class is $co - C = \{L \mid \overline{L} \in C\}$.*

Input data needs to be formally represented to be processed by a Turing machine. A representation is a suitable function, for instance an undirected graph is represented via its adjacency matrix. Any natural representation is reasonable, as they usually can be translated into each other in polynomial time, anyway. Numbers are usually represented in binary notation (observe that unary notation would be exponentially longer!)

Each problem has a set of possible *instances*. Each instance is a mathematical object (for instance, a formula or a graph), of which we ask a particular question.

Definition 3.2.3 (Representation) *A representation of a problem P is a suitable function f mapping each problem instance into a string Σ^*.*

Usually, a problem P is identified with the language L_P defined via f.

Definition 3.2.4 (Decision problem) *A decision problem Π consists of a set D_Π of instances and a subset Y_Π of yes-instances. The complementary problem has instances D_Π and complementary yes-instances $D_\Pi - Y_\Pi$.*

Example 3.2.5 *One of the most common problems encountered is satisfiability (SAT) and its variants:*

INSTANCE: A well-formed propositional formula F.

QUESTION: Is F satisfiable?

For instance, $F = (a \wedge \neg b) \vee (\neg b \wedge c \wedge \neg a)$. This formula is satisfiable, e.g. with interpretation $\mathcal{M} = \{c\}$. As it is easy to see, in disjunctive normal form, it is straightforward to recognize if a formula is satisfiable by looking if one of the constituents is satisfiable. However, the problem is harder if we consider arbitrary formulae, or even if we consider formulae in conjunctive normal form.

Hence, the question arises, why not translate a formula in its disjunctive normal form, because the problem is then easy to treat - namely in polynomial time, while in general the problem is intractable as each possible interpretation needs to be considered. However, it turns out that such a translation usually needs exponential time (there are ways to translate in polynomial time with introducing new variables, but that is not the point here).

Back to the definition of a Turing machine - at each time the machine head is in one particular position doing exactly what the corresponding order tells: If machine M is in state i and reads letter x_j, then do one of the following things: either jump left or right or write a new symbol and additionally enter new state k. Formally:

Definition 3.2.6 (Turing machine) *A deterministic TM (from now on abbreviated as DTM) with k tapes consists of a five-tuple $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ where:*

1. *Q is the finite set of internal states;*
2. *Σ is the tape alphabet (for instance $\{0, 1\}$);*
3. *$q_0 \in Q$ is the initial state;*
4. *$F \subseteq Q$ is the set of final states;*
5. *$\delta : Q \times \Sigma^k \to \Sigma^{k-1} \times Q \times \{-1, 0, +1\}^k$ is a partial function called the transition function of the machine M.*

To get a description of a particular Turing machine at some particular moment of time, one usually specifies its *configuration*, i.e. the contents of the tape with the current state written in front of the symbol scanned.

A machine table of a TM consists of a set of such five-tuples: given a pair (state, read symbol) there is a unique order what to do next (unlike in the nondeterministic model).

Other equivalent definitions of TM include machines with only one tape, or not allowing to step into a direction and simultaneously write a new symbol, however these turn out to be as powerful

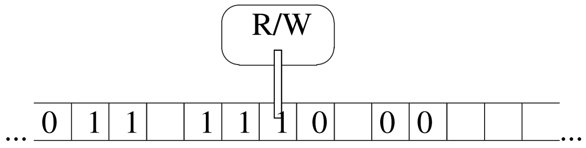

Figure 3.1: *Scheme of a Turing Machine*

as this more handy definition. Furthermore it is well-known that the binary alphabet $\{0, 1\}$ (two symbols) is a sufficient representation for each other alphabet.

As mentioned above, a deterministic Turing machine has a transition function for some pairs (i, x_j), i.e. at most one instruction. In a nondeterministic Turing machine, however, this function turns into a relation allowing to choose one of multiple instructions. nondeterminism is a powerful concept although nondeterministic Turing machines cannot compute anything which a deterministic Turing machine could not, but due to some weak requirements they can speed up computations.

The TM is a very basic model of computation, however, it is not only useful in a recursion-theoretic sense, but also in a complexity-theoretic one as it allows to treat problems in a rather abstract manner.

For complexity theoretic issues we usually use a three-tape Turing machine: the first tape is used as read-only tape with the input stated on it, the second tape as working tape, and on the third tape the output is written.

Now it is straightforward to classify problems using polynomial time, polynomial space and so on - namely how much time the Turing machine needs to compute the answer, and how much space *on the working tape* is needed and so on.

In the *Chomsky hierarchy*, several different grammars and machine models are compared with each other and classified due to their abilities of computability. In this hierarchy, the Turing machine is the most powerful machine model - other models include for instance the finitary automaton: these are machines which are only able to read and after each step move one step right. In Figure 3.2 of the example below the machine diagram of a finite automaton is given (because the computation of this example can be equally carried out by a finite automaton).

Example 3.2.7 *We state an easy example of a Turing machine that stops with* yes *if the input number is divisable by* 3*, and with* no *otherwise.*

Consider a one-tape Turing machine with the head above the first input value and the input given in binary notion.

Of course it is a little bit tougher to program a Turing machine than to program a normal computer, however, everything an ordinary computer can do can be computed with a Turing machine, too.

So how do we proceed? Let us think about when a number is divisible by 3 *(Table 3.2):*

Dec	Bin	Dec	Bin	Dec	Bin	Dec	Bin
0	0	3	11	6	110	9	1001
1	1	4	100	7	111	10	1010
2	10	5	101	8	1000	11	1011
Dec	Bin	Dec	Bin	Dec	Bin	Dec	Bin
12	1100	15	1111	18	10010	21	10101
13	1101	16	10000	19	10011	22	10110
14	1110	17	10001	20	10100	23	10111

Table 3.2: *Binary Numbers divisible by 3*

As one can see, the numbers divisible by 3 *are not straightforward to classify as language. It looks like an even number of* 1 *is required, separated by an even number of* 0 *with* 0 *allowed to be attached at the end of the number. However, it only appears so - 21 is an exception of this. Well, anyway, to encode a Turing machine we do not need to know the accepted language (or the language of yes-instances), but simply encode the division algorithm.*

A Turing machine can be easily encoded to accept this language (even a finite automaton can be programmed to solve it since no write operations, i.e. no memory is needed, and we never need to go left).

Consider three cases for the first numbers read:

$$\begin{array}{ll} 11 & 11 : 11 \textit{ leaves no remainder} \\ 100 & 100 : 11 \textit{ leaves remainder } 1 \\ 101 & 101 : 11 \textit{ leaves remainder } 10 \end{array}$$

So we need to program these three cases in the states of the machine and pass over the remainder through the states. M *starts in state* q_0. *If* M *reads a* 0, *then it just needs to go right till the number really starts.*

$$\langle q_0, 0, 0, q_0, +1 \rangle$$

if scanning a 0, *just go right till the number starts,*

$$\langle q_0, 1, 1, q_1, +1 \rangle$$

if reading a 1, *enter a new state and jump one step to the right.*

If another 1, *now in state* q_1, *is discovered,* M *can safely go back to* q_0 *to initialize the division algorithm again (as no remainder is left) - the* 0*'s it discovers in* q_0 *are unimportant. However, if* M *reads a* 0 *in* q_1, *then two different things can happen, hence we simply jump into a new state and deal with these options in the new state:*

$$\langle q_1, 0, 0, q_2, +1 \rangle$$

if M detects a 0 go to state q_2.

$$\langle q_1, 1, 0, q_0, +1 \rangle$$

if M scans a 1 in state 1 it can jump back to state 0 and if subsequently a 1 is discovered, launch the algorithm again.

In q_2, if a 0 is read, then the remainder is 1, hence we need to go into a state in which the machine believes that it recently simply read a 1, since the 1 is passed on in the algorithm. This state is q_1. If M reads a 1 instead, then the required history is 10, this is exactly what happened before M reached the state, hence M remains in q_2 (and as always moves one step to the right).

$$\langle q_2, 0, 0, q_1, +1 \rangle$$
$$\langle q_2, 1, 0, q_2, +1 \rangle$$

Additionally, we need states that deal with the case the number ends, i.e. an empty space is found. If an empty space is discovered in q_0, then the number is divisible by 3, if an empty space is discovered in q_1 or q_2, the number is not divisible by 3, hence

$$\langle q_0, *, *, yes, 0 \rangle$$
$$\langle q_1, *, *, no, 0 \rangle$$
$$\langle q_2, *, *, no, 0 \rangle$$

This completes the machine table. Alternatively, the machine could be coded to run forever if the input is not divisible by 3. Then the accepted language of the machine would be the numbers divisible by 3. Now the accepted language are all binary numbers, but the accepted language of yes-instances are the multiples of 3. However, for abbreviation, we usually call the language of yes-instances the accepted language.

In this machine table the transition function is a complete function, however it would be allowed that it has no value for some pairs of states and symbols.

As it easily can be seen such limited machine tables (step right, no writing) correspond to a rule system which generates the same language - namely to a linear grammar which is as powerful as a finite automaton. Several machine models and grammars lead to the previously mentioned Chomsky-hierarchy, however due to space constraints we cannot go into details here.

The more involved way is if we directly try to find out the structure of the the language - this is not that difficult for 3, but rather tough for 5 or other numbers, and needs a little bit of number theory. It is far easier to look at the machine table, as this machine is so easy - no write operations, only steps right, thus we easily find out that the language is $0^* \circ 1 \circ (01^*0) \circ 1)^* \circ 0$.

Figure 3.2 depicts the finite automaton in the usual notion.

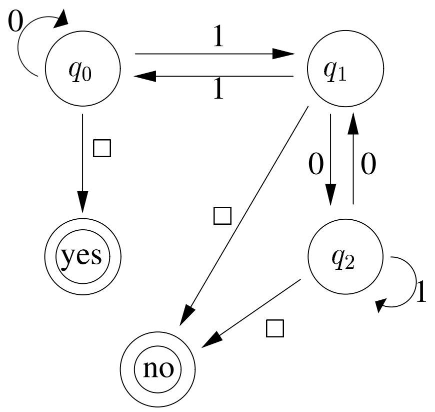

Figure 3.2: *Finite Automaton for Example 3.2.7*

Example 3.2.8 *As mentioned above, any suitable representation of a problem is reasonable. Consider the (directed) graph in Figure 3.3. G can (for instance) be represented by encoding it as*

- $G = (V, E)$, *where* $V = \{1, 2, 3, 4, 5\}$ *is the set of vertices and*

$$E = \{(1, 2), (1, 4), (2, 1), (2, 2), (2, 4), (2, 5), (3, 4), (4, 5), (5, 2)\}$$

  is the set of directed edges. This pair is coded in a suitable way.

- *The cardinality of vertices (5) and the adjacency matrix*

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

  with entry 1 *in* ij *if there exists a directed edge from* i *to* j *and otherwise entry* 0*; and the matrix is encoded as a string.*

- *describing* $|V|$, $|E|$, *and the vertex-edge incidence matrix.*

- *In an implicit way, where each entry of the adjacency matrix can be computed by the input data.*

- $\ldots$

One can verify that each representation can be transformed into each other efficiently, i.e. in polynomial time.

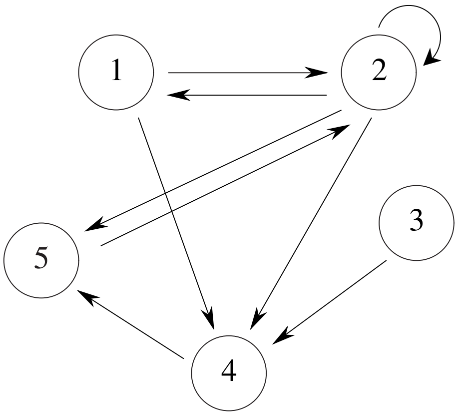

Figure 3.3: *A Simple Directed Graph*

The Invariance Theses: All common computation models simulate each other with polynomial time overhead. For instance, a Random Access Machine simulates a k-tape TM in quadratic time. Therefore, we are allowed to describe algorithms informally.

Definition 3.2.9 *A nondeterministic Turing machine (NDTM) with k tapes consists of a five-tuple $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ where*

1. *Q is the finite set of internal states;*
2. *Σ is the tape alphabet (for instance $\{0, 1\}$);*
3. *$q_0 \in Q$ is the initial state;*
4. *$F \subseteq Q$ is the set of final states;*
5. *$\delta : Q \times \Sigma^k \rightarrow P(\Sigma^{k-1} \times Q \times \{-1, 0, = 1\}^k)$ is a partial function called the transition function of the machine M with $P(A)$ denoting the power set of a set A.*

In each step the nondeterministic machine chooses one out of a set of possible transitions. Nondeterminism is a powerful concept, since it is possible to present some computations in polynomial time which otherwise probably need exponential time (depending whether $\mathbf{P} \neq \mathbf{NP}$). However, observe that nondeterminism, although it is a powerful and very basic theoretical tool, is far from being something like a "realistic model" of computation, due to the definition of accepting states given below: A nondeterministic computation is successful, exactly if one of the computation paths results in a final state *yes*.

Definition 3.2.10 *An input string $\omega \in \Sigma^*$ is accepted by a nondeterministic TM, if* there is *a computation which ends in an accepting configuration.*

The language accepted by M, denoted $L(M)$, is the set of words (strings) accepted by M.

Just one of the computation paths needs to lead to a success, nondeterminism can be displayed as a tree, where one path is "guessed and checked". This allows to define a nondeterministic machine equivalently as a *guess-and-check-machine*, namely initially guessing a sequence of 0 and 1, and afterwards starting a deterministic procedure.

This modification of the criteria of acceptance with different treating of yes/no-states leads to a fundamental asymmetry in nondeterministic complexity classes, as discussed in the next section. Sometimes, this asymmetry vanishes because the deterministic class corresponding to the deterministic TM solving a problem within given time or space constraints coincides with the nondeterministic class, or sometimes only the nondeterministic class and its complement coincide, but the deterministic counterpart is a proper subset.

Notice, that this is the same as in automata theory, where the language computable by nondeterministic automata for some automata is the same, for some a superset. However, we cannot go into details here and refer to [Hopcroft and Ullman, 1979].

Definition 3.2.11 (Landau symbol) *Let f and g be functions from $\mathbb{N}$ to $\mathbb{N}$. We write $f(n) = O(g(n))$ (= f is of the order of g) if there are positive integers c and n_0 such that for all $n \geq n_0$, $f(n) \leq c \cdot g(n)$.*

With this notation, every polynomial is of order $O(n^k)$ with k being the number of the highest degree occurring in this polynomial. Every polynomial grows strictly slower than every exponential, hence 2^n cannot be captured by $O(n^k)$, regardless of the degree of k.

Example 3.2.12 *Besides problems from logic, graph theoretical problems are par excellence suited for demonstration of complexity classes. Think about the following two questions:*

- *Determine if there exists a path from vertex i to vertex j.*
- *Determine if a graph has a clique of size k.*

The first problem is also known as REACHABILITY, and, as it is easy to see, this problem is solvable in polynomial time for both directed and undirected graphs: Starting with vertex i, create a set of all nodes which are connected to i, and in the subsequent steps continue this for all nodes in this set. At most after n steps, where n is the number of vertices, all possible nodes from i have been visited. Each step consists of at most n checks. Indeed, the problem is not among the hardest polynomial problems, but a little bit easier - for directed graphs it is $\mathbf{NL}$*-complete (refer to Section 3.3 for clarifying abbreviations like this one).*

CLIQUE is the problem if a graph of size n contains a clique of size k, that is a fully connected subgraph. Observe that k is like n part of the input, and not a fixed parameter. The straightforward algorithm checks for all subsets of cardinality k whether the graph is fully connected. Checking if *all nodes are connected can be done in polynomial time, and a number of $O(n^k)$ sets need to be considered.*

Since k is part of the input, this problem is exponential in the input length $O(n \cdot k)$. However, there is an easy algorithm for a nondeterministic machine - given a graph G with n vertices, it guesses one subset of G of the required cardinality k, corresponding to one path in the computation, and checks if this set forms a clique.

In the guessing procedure one path is chosen which yields a yes-instance, if there is any. Hence, this problem can computed in a straightforward way by a nondeterministic Turing-machine in polynomial time.

3.3 Complexity Classes

Let us deal with the complexity classes defined upon the concept of Turing machine computation. One of the classes of main interest is $\mathbf{P}$, which is defined as the class of languages (problems) solved by a deterministic Turing- machine in polynomial time w.r.t. the input length.

The corresponding nondeterministic class is based on nondeterministic machines, introducing the well-known class $\mathbf{NP}$. This computation can be described with trying all the paths of computation or equivalently, that first a path is guessed and then in polynomial time checked if this path yields a success, as described in the previous section.

A problem having its converse problem in $\mathbf{NP}$ is in $\mathbf{coNP}$: nondeterministic classes are in general not closed under complementation, hence it is assumed, although not proven so far, that $\mathbf{NP} \neq \mathbf{coNP}$. If $\mathbf{P}$ were $\mathbf{NP}$, then of course $\mathbf{NP} = \mathbf{coNP}$ would be a direct consequence. However, the question if $\mathbf{P} = \mathbf{NP}$ is one of the most fundamental unsolved questions in theoretical computer science, and still, there is no solution in sight, and no real good results indicating the one or the other, just that there are oracles A for which $\mathbf{P}^A = \mathbf{NP}^A$ and oracles, for which this equation is not valid. Maybe, this question is even independent from axiomatic set theory, and hence cannot be decided (cf. eg., [Papadimitriou, 1993]).

Example 3.3.1 *Given a propositional formula f and assume it is asked if this formula is satisfiable - i.e. whether this formula has at least one model. Such existential questions are closely tied to* $\mathbf{NP}$*, since these problems can be formulated as "there is a computation path" and one possible model can be guessed and checked in polynomial time . However, the converse problem, if f is unsatisfiable, is in* $\mathbf{coNP}$*, since all models need to be considered.*

In particular, for formulae in conjunctive and disjunctive normal form (CNF, DNF), and for Horn formulae, the membership complexities of satisfiability and related problems are stated in Table 3.3.

	CNF	DNF	Horn
SAT	**NP**	**P**	**P**
$\neg$SAT	**coNP**	**P**	**P**
$\vdash$	**P**	**coNP**	**P**
$\neg\vdash$	**P**	**NP**	**P**

Table 3.3: *Class-Membership for Satisfiability*

3.4 Time and Space Bounds

Formally, the definition of time and space bounds is defined as follows.

Definition 3.4.1 *The size of a problem instance I is the number of symbols of the string x representing I, abbreviated as $|I|$.*

Definition 3.4.2 (Running time) *The running time of a deterministic TM M on input I, $time_M(I)$, is the number of steps until M halts, and undefined, if M does not halt. The running time of a NDTM M on input I is the maximum number of steps over all possible computations (i.e. the depth of the computation tree).*

The running time of a machine M on a problem is then defined as the maximum time needed for some problem instance with input length n. Time and space are measured with well-behaving functions.

Definition 3.4.3 (time/space constructibility) *A total recursive function f is time constructible (space constructible) if there is a TM which halts on every input on length n in exactly $f(n)$ steps (such that $f(n)$ cells have been used and are not blank and no other cell has been used.)*

Example 3.4.4 *Examples of time constructible functions are n^k, $n!$ and $n \cdot \lceil \log n \rceil^k$.*

Definition 3.4.5 *$DTIME(t)$ ($NTIME(t)$) is the class of all sets accepted by (non)deterministic Turing machines M with $time_M(n) \leq t(n)$, for $n \geq 0$ with a function $t(n) \geq n+1$ which is time-constructible.*

Definition 3.4.6

$$\mathbf{P} = \bigcup_{i \geq 0} DTIME(n^i)$$

$$\mathbf{NP} = \bigcup_{i \geq 0} NTIME(n^i)$$

$$\mathbf{EXPTIME} = \bigcup_{i \geq 0} DTIME(2^{n^i})$$
$$\mathbf{NEXPTIME} = \bigcup_{i \geq 0} NTIME(2^{n^i})$$

This is the formal definition of some time complexity classes as union of all problems that can be solved in a given time frame. Similarly, the space complexity classes can be defined. Moreover, there are many further natural and unnatural complexity classes defined with other bounds than space or time, or defined with other machine models, like alternating Turing machines, and so on, sometimes corresponding to already well known classes. Refer to [Papadimitriou, 1993] for more details on these topics.

Turing machines which are allowed to use a polynomial number of tape cells for their computation introduce the class $\mathbf{PSPACE}$.

Formally:

Definition 3.4.7 *The space used by a DTM M on input I, $space_M(I)$, is the number of different visited cells of the work-tape until M halts, and is undefined if M does not halt. The space used by a NDTM M on input I, $space_M(I)$, is the maximum number of different visited work-tape cells of all possible computations.*

The space used by machine M for n, $space_M(n)$ is the maximum over all $space_M(I)$ where $|I| = n$.

Definition 3.4.8 *$DSPACE(t)$ ($NSPACE(t)$) is the class of all sets accepted by (non)deterministic Turing machines M with $space_M(n) \leq t(n)$, for $n \geq 0$ with a function $t(n) \geq 1$ which is space-constructible.*

For space complexity classes, it is important to use the three-tape Turing machine model, and only measure the space used on the second tape, the working tape. Without this, classes like logarithmic space (for short: logspace) would not be possible to define, because it takes linear space even to read the input!

Definition 3.4.9

$$\mathbf{L} = \bigcup_{c \geq 1} DSPACE(c \cdot \log n)$$
$$\mathbf{NL} = \bigcup_{c \geq 1} NSPACE(c \cdot \log n)$$
$$\mathbf{PSPACE} = \bigcup_{i \geq 0} DSPACE(n^i)$$
$$\mathbf{NPSPACE} = \bigcup_{i \geq 0} NSPACE(n^i)$$

It turns out that the following inclusions and relations hold:

$$\begin{array}{ccc} \mathbf{P} \subseteq \mathbf{NP} & \mathbf{P} \subseteq \mathbf{PSPACE} & \mathbf{PSPACE} \subseteq \mathbf{EXPTIME} \\ \mathbf{L} \subseteq \mathbf{NL} & \mathbf{NL} \subset \mathbf{PSPACE} & \mathbf{NP} \subset \mathbf{NEXPTIME} \\ \mathbf{NL} \subseteq \mathbf{P} & \mathbf{NP} \subseteq \mathbf{PSPACE} & \mathbf{P} \subset \mathbf{EXPTIME} \\ & \mathbf{PSPACE} = \mathbf{NPSPACE} & \end{array}$$

Open are for all of these problems with $\subseteq$ whether the inclusion is strict; i.e. whether it holds that:

$$\begin{array}{ccc} \mathbf{L} \neq \mathbf{NL} & \mathbf{P} \neq \mathbf{NP} & \mathbf{NP} \neq \mathbf{PSPACE} \\ \mathbf{NL} \neq \mathbf{P} & \mathbf{P} \neq \mathbf{PSPACE} & \mathbf{PSPACE} \neq \mathbf{EXPTIME} \end{array}$$

and moreover, the questions if $\mathbf{NP}$ is a proper subset of $\mathbf{EXPTIME}$, and the relation between $\mathbf{NP}$ and $\mathbf{ETIME}$ are totally unclear, where

$$\mathbf{ETIME} = \bigcup_{i \geq 0} DTIME(2^{i^n}).$$

Additionally, it is not proved so far if $\mathbf{NP} \neq \mathbf{coNP}$ as stated above.

However, it is known that either $\mathbf{P} = \mathbf{NP}$, or there are problems in $\mathbf{NP} \backslash \mathbf{P}$ which are not $\mathbf{NP}$-hard (see also the following sections on hardness). One of these problems could be PRIMES, the problem to decide if a natural number is prime - there are algorithms showing that this problem is in $\mathbf{NP}$ and there are algorithms domiciling this problem in $\mathbf{coNP}$; hence it cannot be hard for one of these classes - nevertheless no polynomial algorithm is known.

It is known that $\mathbf{L} \subset \mathbf{PSPACE}$. Most interestingly, $\mathbf{NL}$ and $\mathbf{coNL}$ coincide. However, $\mathbf{L}$ is probably a proper subset of $\mathbf{NL}$. On the other hand, $\mathbf{PSPACE}$ and its nondeterministic counterpart $\mathbf{NPSPACE}$ coincide, implying that

$$\mathbf{co} - \mathbf{NPSPACE} = \mathbf{NPSPACE}.$$

Some other complexity classes include:

$$k - \mathbf{EXPTIME} = \bigcup_{i \geq 0} DTIME(g(k, n, i)),$$

$$\mathbf{ELEMENTARY} = \bigcup_{i \geq 0} i - \mathbf{EXPTIME}.$$

where $g(0, n, i) = n^i$ and $g(k + 1, n, i) = 2^{g(k,n,i)}$.

3.5 Oracle machines

Oracles are very fine things: you consult them and they tell you something which is hard to understand or to explain, problems where you would need a lot of time to find the answer yourself. Oracles are clever, although they are only allowed to answer closed questions, i.e. with *yes* or *no*.

Say, you want to solve a problem, for instance a problem which contains n satisfiability problems as subproblems. The problem itself is easy and you are sure you can solve it in polynomial time, however, these SAT instances are tough to solve. So you are glad that the old lady you have as a neighbour is a very wise oracle. You continue to solve the problem and each time you are confronted with a SAT instance you call her, tell her the instance and she answers immediately with yes or no. Hence, you have solved this problem in polynomial time with access to an oracle for satisfiability.

Based on this concept of wise old women, oracle Turing machines are defined. These are TM with access to an oracle, which returns *yes* or *no* in unit time. Formally an oracle is treated as a subprogram:

Definition 3.5.1 (Oracle Machine) *A (non)deterministic oracle Turing machine is a multi-tape TM with one designated tape, namely the oracle tape, with*

$$\{query, yes, no\} \subseteq Q.$$

If the machine is in the state query, a call to the oracle is performed: If the string ω on the query tape is in $O \subseteq \Sigma^*$ (oracle set) then change state to yes, else to no. Afterwards erase the contents of the query tape.

An oracle call counts as single step (unit time). Oracles are even far more clever than you believe: The old lady from next door, she cannot only deal with SAT problems, she can easily deal with CLIQUE problems, too. Maybe you need to explain it to her sitting together at a coffee break - and if this takes too long, you could instead drink your coffee at home while you encode CLIQUE in polynomial time as a SAT-problem and just give her this problem to solve.

And she can furthermore deal with UNSAT in the same manner (consider the symmetry in the construction of the query tape!) giving you the right answer. An oracle for A is as good as one for $\overline{A}$, and an oracle for a specific problem as good as one for a whole class of problems.

Definition 3.5.2 *The language accepted by an OTM M relative to an oracle set A, denoted by $L(M, A)$ is the set of all strings accepted by the OTM using the oracle set A.*

With these definitions, one can define complexity classes with access to oracles. For instance, $\mathbf{P}^{\mathbf{NP}}$ is the class of problems that are solvable in polynomial time with access to an oracle in $\mathbf{NP}$. As it can easily be seen that is the same as $\mathbf{P}^{\mathbf{coNP}}$ and $\mathbf{P}^{\mathbf{P}^{\mathbf{NP}}}$. Details about the polynomial hierarchy based on this concept are presented in Section 3.8.

Remark. If we allow the oracle space to be unrestricted, then also classes like $\mathbf{PSPACE}^{\mathbf{P}}$ create higher classes - this class then coincides with $\mathbf{EXPTIME}$. However, with restricted oracle space the class for instance coincides with $\mathbf{PSPACE}$. Compare for instance to [Eiter and Veith, 1999] for use of such concepts.

3.6 Search and Function Problems

Sometimes you are not happy with closed questions and want to formulate questions like "Tell me the shortest way from A to B" or, given a number of SAT-instances, you simply want to know the number of satisfiable formulae.

These problems can be re-formulated as decision-problems, and on these problems techniques like binary search can be applied, for instance starting with the question if the number of satisfiable formulae is lesser than $n/2$ with n being the number of SAT-instances and depending on the answer subsequently refine the interval.

Hence, for decision problems in $\mathbf{NP}$, related search problems are usually not much harder and are called $\mathbf{NP}$-easy problems. This class corresponds to Δ_2^P, which is equivalent to $\mathbf{P}^{\mathbf{NP}}$. Not all search problems are hard for this class, the ones solvable with binary search are a little bit easier - see also Proposition 5.2.4.

However, sometimes, given a propositional formula, you not only want to know if it is satisfiable, but the truth-assignments, too, or at least one truth-assignment. This problem can be solved like the maximum truth assignment problem (cf. Section 3.9) (since the maximum truth assignment is one truth assignment) and is a little bit harder than the problem of retrieving the number of satisfiable SAT instances out of n instances.

Search problems can be classified using own concepts of complexity classes and reductions. The class of all function problems associated with languages in $\mathbf{NP}$ is called $\mathbf{FNP}$. One further class of function problems is the class $\mathbf{TFNP}$ (which is believed to be a superset of $\mathbf{FP}$, the class of polynomial function problems), the class of total function problems based on languages in $\mathbf{NP}$. One of the problems in this class is factoring a given natural number - of course this problem always has a (unique) solution.

3.7 Hardness

The most important notion of complexity theory is the notion of *reduction*. To show that a problem is $\mathbf{Q}$-hard transform a $\mathbf{Q}$-hard problem into a special instance of this problem with a polynomial transformation (or instead a logspace-transformation, to take into account $\mathbf{P}$-complete problems; it is not known although assumed that all problems which are hard w.r.t. polynomial transformations are also hard for the respective classes using a logspace transformation). A formal definition is presented on the next page.

There are different notions of reductions. The standard reductions are polynomial time transformations, or logspace transformations. However, there are other transformations, for instance the Turing transformations, used for oracle computations, which could also serve as a hardness measure. W.r.t. function problems there are metric reductions, and there are many more reductions, for instance for problems with succinct input: the conversion lemma uses different kinds of notions of reductions to compare two problems and to compare the succinct version of two problems (like quantifier-free reductions). For more details on succinct problems refer to [Veith, 1996].

Consider the class $\mathbf{NP}$. We know that $\mathbf{P} \subseteq \mathbf{NP}$, but is it furthermore possible to classify the problems in $\mathbf{NP}$ in equivalence classes based on some kind of transformability into each other? Indeed, it is, although we only can directly tell something about the hardest of these classes! For instance, look at the problem if a propositional formula is satisfiable and at the problem if a graph has a vertex cover.

An undirected graph (V, E) has a vertex cover iff $\exists V' \subseteq V\ \forall (v, w) \in E :\ v \in V'$ or $w \in V'$.

Vertex Cover (VC) -
INSTANCE: Given an undirected graph $G = (E, V)$ and an integer $K \leq |V|$.
QUESTION: Is there a vertex cover of size $\leq K$?

Is this problem polynomially transformable into SAT? If yes, then it is straightforward to classify SAT as harder in some sense, or at least as hard as this problem, and we say that the problem is harder due to polynomial many-one (= not injective transformation, several instances of the first problem can be mapped into one instance of the second problem) transformation.

With a notion of reduction, i.e. transformation, one can structure $\mathbf{NP}$ into equivalence classes. The hardest problems in $\mathbf{NP}$ are called $\mathbf{NP}$-complete problems - completeness is defined as hardness and membership in the corresponding class. With this notion of transformation we can furthermore introduce complete problems for all other mentioned complexity classes. All these mentioned classes are closed under this reduction; and it is easy to recognize that the notion of polynomial many-one transformability is transitive.

However, with this notion of transformation all problems in the class $\mathbf{P}$ automatically are $\mathbf{P}$-hard because of course all of them can be translated into each other. If we want to distinguish $\mathbf{P}$-hard problems from problems that are **NL**-hard, or are in **L**, then we need to introduce logspace-reductions, i.e. reductions which can be carried out in logarithmic space. In Chapters 5 to 7 we generally apply polynomial transformations since they are sufficient for higher classes - and only apply logspace-reductions to show $\mathbf{P}$-hard problems. However, if you have a closer look at the proofs in these chapters you will recognize that they all are logspace transformations as they usually only need constant space depending on the length of the transformed problem.

In the next subsection we will show that VC is really reducible to SAT. Before that, let us formalise these thoughts:

Definition 3.7.1 (Polynomial/Logspace Reduction) *Given two languages L_1 and L_2, L_1 is polynomially (logspace) many-one reducible to L_2, formally $L_1 \leq_m^P L_2$ iff there exists a function*

$f : \Sigma^* \rightarrow \Sigma^*$ s.t. $f(w)$ is computable in time polynomial in $|\omega|$ (space logarithmic in $|\omega|$) and $x \in L_1$ iff $f(x) \in L_2$ for all $x \in \Sigma^*$.

Definition 3.7.2 (Hardness) *Given a class* $\mathbf{Q}$ *a problem (language) A is* $\mathbf{Q}$*-hard w.r.t polynomial reductions iff for every $B \in \boldsymbol{Q}$ it holds that $B \leq_m^P A$.*

Definition 3.7.3 (Completeness) *Given a class* $\mathbf{Q}$ *a problem (language) A is* $\mathbf{Q}$*-complete iff it is in* $\mathbf{Q}$ *and* $\mathbf{Q}$*-hard w.r.t polynomial reductions.*

So, all we need is one $\mathbf{NP}$-hard problem, i.e. one problem which is in the hardest equivalence class to easily derive which problems are in this class, too. The first problem shown to be $\mathbf{NP}$-complete was Satisfiability, proven by Cook with a generic reduction to a Turing machine which works in nondeterministic polynomial time.

After this, many problems have been shown to be $\mathbf{NP}$-complete by reducing SAT, or any other problem shown to be $\mathbf{NP}$-complete to some special instance of it. If a problem is $\mathbf{NP}$-hard, then it can be in some higher class, too - so for each problem it needs to be proved, too, that it is in $\mathbf{NP}$ to derive a completeness result.

For reduction, some problems are better suited than others, but in theory, every $\mathbf{NP}$-complete problem can be encoded into every other $\mathbf{NP}$-complete problem.

Today, there is a large catalogue of $\mathbf{NP}$-complete problems and of problems complete for other classes. The book [Garey and Johnson, 1979] started the whole thing off - with an appendix of lots of problems from different areas of computer science and logic.

Some trivial consequences:

1. If problem A is $\mathbf{NP}$-hard, then the converse problem $\overline{A}$ is $\mathbf{coNP}$-hard.
2. The notion of $\leq_m^P$ is reflexive and transitive, i.e. is a pre-order.
3. $A \leq_m^P B$ is equivalent to $\overline{A} \leq_m^P \overline{B}$

Example 3.7.4 *3SAT is the problem if a formula in CNF with at most three literals per clause is satisfiable. It is well known that this problem is* $\mathbf{NP}$*-complete, too. 2SAT however, is known to be easier, namely in* $\mathbf{NL}$*.*

Example 3.7.5 *As promised we now show that VC is* $\mathbf{NP}$*-hard. To show that a problem is* $\mathbf{NP}$*-hard we encode, using a polynomial transformation, another* $\mathbf{NP}$*-hard problem into it. For reduction, we use in this case 3SAT.*

An instance of 3SAT is given by

$$f = \bigwedge_{i=1}^{n} (a_{i1} \vee a_{i2} \vee a_{i3}).$$

Here n is the number of clauses, and a_{ij} are negated or unnegated literals out of the propositional variables $\{b_1, \ldots, b_m\}$.

The aim is to show that this formula is satisfiable exactly if a particular graph has a vertex cover of size $\leq k$. It is not necessary to reach all the possible graphs with transformations of formulae - it is sufficient to show that the problem is hard for some particular graphs.

Moreover, this translation needs to be fulfilled in polynomial time.

To get some first ideas, think about encoding each a_{ij} as node v_{ij} of the graph. Then, the edges somehow should represent the structure of the formula: Connect all nodes v_{ij} to v_{ik} for $j \neq k$ and connect all nodes with $a_{ij} = \neg a_{kl}$. The intuition is that the vertex cover corresponds to all literals that are "not needed" in a truth-assignment.

We need to show: F is satisfiable exactly if G has a vertex cover of size $\leq 2n$.

- *If F is satisfiable, then choose a satisfying truth assignment. In such an assignment at least one literal of each clause is true. Let V' be the set of vertices without the nodes corresponding to these literals. Since none of these literals are interconnected with each other, at least one vertex of each edge is part of V', hence the one direction is proved.*

- *If V has a vertex cover of size $\leq 2n$, then from each "triangle" at least one node does not belong to V'. The set of nodes $V \backslash V'$ has no edges due to the definition of vertex cover. Due to the definition of the graph, $V \backslash V'$ corresponds to a truth value assignment of F.*

The transformation is feasible in polynomial time, and therefore VC is $\mathbf{NP}$-hard. Moreover, VC is in $\mathbf{NP}$ - just apply the following guess-and-check-procedure: guess a subset $V' \subseteq V$ with $|V'| \leq k$ and check in polynomial time if it is a vertex cover. Hence, the problem is $\mathbf{NP}$-complete.

In essence, all hardness-proofs follow this schema; sometimes it is straightforward to prove membership like in this example, sometimes it is very hard to prove membership esp. for complexity classes using a limited number of oracle calls (like in our theorem in Section 5.3).

3.8 The Polynomial Hierarchy

A wide variety of classes between $\mathbf{NP}$ and $\mathbf{PSPACE}$ have been introduced by defining Turing machines with oracle computation - the *polynomial hierarchy*. Figure 3.4 shows some complexity classes and how they interrelate to each other.

Definition 3.8.1 (Turing reducibility) *A problem X is (nondeterministically) polynomial time Turing reducible to a problem Y, $X \leq_T Y (X \leq_{NT} Y)$, if there is a polynomial DTM (NDTM) for X with access to an oracle for Y.*

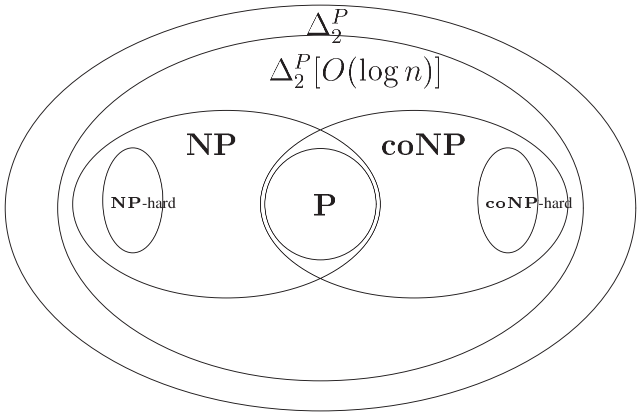

Figure 3.4: *Complexity Classes*

This definition introduces another notion of reduction, a reduction which distinguishes the levels of the polynomial hierarchy. Recall that for $\mathbf{NP}$-hardness the following is valid: A problem X is $\mathbf{NP}$-hard (under polynomial transformation!) if there is an $\mathbf{NP}$-complete problem Y s.t. $Y \leq_T X$.

To create some confusion: A problem X is called $\mathbf{NP}$-easy, if for some problem $Y \in \mathbf{NP}$, $X \leq_T Y$. This notion was introduced because $\mathbf{NP}$-easy problems are not much harder than problems in $\mathbf{NP}$. These are *not* easy problems in $\mathbf{NP}$ because this notion is based on another kind of reducibility! The class of $\mathbf{NP}$-easy problems is the class Δ_2^P (see below).

This means that X can use an oracle for Y, this forms the class $\mathbf{P}^{\mathbf{NP}}$, a superset of $\mathbf{NP}$ and $\mathbf{coNP}$. With extension of this notion to whole classes, this is formalised:

$$\mathbf{P}^C = \{L \mid \exists L' \in C : \ L \leq_T L'\}$$

$$\mathbf{NP}^C = \{L \mid \exists L' \in C : \ L \leq_{NT} L'\}$$

Based on these facts and with a motivated notation from the arithmetical hierarchy of recursion theory [Oberschelp, 1993], the polynomial hierarchy is formally introduced. The classes Δ_k^P, Σ_k^P and Π_k^P of the polynomial time hierarchy (PH) are defined as

$$\Delta_0^P = \Sigma_0^P = \Pi_0^P = \mathbf{P}$$

and for all $k \geq 0$,

$$\Delta_{k+1}^P = \mathbf{P}^{\Sigma_k^P}, \Sigma_{k+1}^P = \mathbf{NP}^{\Sigma_k^P}, \Pi_{k+1}^P = \mathbf{co}{-}\Sigma_{k+1}^P.$$

In particular, $\mathbf{NP} = \Sigma_1^P, \mathbf{coNP} = \Pi_1^P$ and $\Delta_2^P = \mathbf{P}^{\mathbf{NP}}$. Thus Δ_2^P is the class of decision problems that are solvable in polynomial time on a deterministic oracle Turing machine that makes a polynomial number of calls to an $\mathbf{NP}$ oracle.

The classes Δ_k^P have been refined depending on how many oracle calls are needed [Wagner, 1990]: Of special interest is the class $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$, also called $\Delta_2^P[O(\log n)]$, of decision problems solvable with a logarithmic number of calls to an $\mathbf{NP}$-oracle.

A problem is on the l-th level of the PH if it is Δ^P_{l+1}-complete under Turing reductions with completeness defined as being Σ^P_l-easy and Σ^P_l-hard or Π^P_l-hard with the notions given above.

3.9 Some Complete Problems

It is an open question whether the polynomial hierarchy is infinite, or collapses at some level. The most well known $\mathbf{PSPACE}$-complete problem is QBF (Quantified Boolean Formula): A propositional formula pre-switched with second order quantification: $\exists$ stands for *there is a truth evaluation for these variables* and $\forall$ stands for *for all truth evaluations of these variables*. Allowing an arbitrary number of quantification - with no restriction where which quantifier is allowed to occur - this problem is $\mathbf{PSPACE}$-complete. However, restricting the number of quantifiers, and the starting quantifier, the problem is complete for the respective classes of the polynomial hierarchy:

QBF$_{i,Q}$ with i is the number of quantifier alternations and $Q \in \{\forall, \exists\}$, is Σ^P_i-complete if $Q = \exists$ and Π^P_i-complete, respectively if $Q = \forall$.

Example 3.9.1 *It is Σ^P_2-complete to determine if the formula*

$$\exists\{a_1, a_2, \ldots, a_n\}\, \forall\{b_1, b_2, \ldots, b_m\}\, (a_1 \vee b_2 \vee \cdots) \wedge (\neg a_2 \vee b_1 \vee \cdots) \Rightarrow \cdots$$

is valid, i.e. if there exists a truth assignment for the a_i s.t. for all truth assignments of the b_j the propositional formula is true.

The following definition is needed afterwards:

Definition 3.9.2 *$\chi_{SAT}(F)$ is the characteristic function of satisfiability: If the formula F is satisfiable, then its value is 1 else 0.*

The deterministic classes of the polynomial hierarchy can be refined depending on the number of oracle calls needed: Δ^P_2 does not limit the calls, while $\Delta^P_2[O(\log n)]$ limits them to a logarithmic number.

The most used complete problems for these classes are MSA-odd for Δ^P_2-completeness, and SAT$^n_{odd}$ for $\Delta^P_2[O(\log n)]$-completeness (see below).

Many problems in Δ^P_2, although not directly apparent, can use a refined algorithm using only a logarithmic number of oracle calls like we use in Section 5.3. Two fundamental tools are binary search and the concept of $\mathbf{NP}$-trees [Gottlob, 1995c]. As soon as it is possible to display a problem as $\mathbf{NP}$-tree, it only needs a logarithmic number of oracle calls (cf. Section 3.10). In Chapter 7 we make use of a concept very similar to $\mathbf{NP}$-trees, namely that of $3val$-trees.

Furthermore, the class $\Delta^P_2[O(\log n)]$ coincides with the class $\mathbf{P}^{\mathbf{NP}}_{\|}$ of problems solvable with one (or even with a constant number of rounds) parallel evaluation of n independent queries to $\mathbf{NP}$-oracles.

However, some problems in Δ_2^P have some kind of inherent linearity, and cannot be logarithmized. Most fixed-point constructions fall into this category, since these step-by-step procedures cannot be simplified. In Chapter 7 we will deal with the complexity of such a fixed-point iteration problem and the reader will get an impression of what makes a problem Δ_2^P-hard, and what allows for a $\Delta_2^P[O(\log n)]$-hard problem.

MSA-odd is complete for Δ_2^P: given a propositional formula F in propositional variables $p_1, \ldots, p_n$, decide whether the lexicographically maximum truth assignment satisfying F w.r.t. the sequence $\langle p_1, \ldots, p_n \rangle$ is odd, i.e. assigns 1 to p_n. This truth assignment v is recursively defined as

$$v(p_1) = \chi_{SAT}(F(\mathbf{t}, p_2, \ldots, p_n))$$

$$v(p_{i+1}) = \chi_{SAT}(F(v(p_1), v(p_2), \ldots, v(p_n), \mathbf{t}, p_{i+2}, \ldots, p_n))$$

for $1 \leq i \leq n$ with χ_{SAT} is the characteristic function of satisfiability. Do not be confused by "is odd", this is just because it is usual in computer science to classify problems in such a way - observe that this is a decision problem, answering *yes* if the maximum assignment assigns true to p_n and *no* else.

This is a step-by-step construction and hence can only be linearized. A quite different problem is SAT$^n_{odd}$: An instance of this problem is given by a number of n independent SAT instances and the instance is positively solved iff the number of SAT instances with a positive solution is odd.

First of all this problem is not in $\mathbf{NP}$, although your intuition might first hint it were: Just guess a structure of n truth-assignments ... - however, this is the problem! If one instance is not satisfiable there will not be such an assignment and $\mathbf{NP}$ and $\mathbf{coNP}$-problems are mixed in some way if everything is guessed as a structure. Hence subsequent oracle calls are needed, or equivalently, n independent parallel calls. In this example it is also easy to see that the calls can be logarithmized (hinting that $\Delta_2^P[O(\log n)]$ and $\mathbf{P}_{\|}^{\mathbf{NP}}$ are equivalent):

This problem can easily be solved with binary search: First, prove that the problem to determine if the number of instances is greater or equal to i is in $\mathbf{NP}$: Namely, guess a data structure $\langle \mathcal{F}, \mathcal{M}_1, \ldots, \mathcal{M}_i \rangle$ (guessing data structures is a very important tool, also used in our proof in Section 5.3 and in [Gottlob, 1995]), where $\mathcal{F}$ is a subset with cardinality i of all given SAT-instances and $\mathcal{M}_j$ are interpretations (to be exact, interpretations in all variables of all SAT instances, even the ones not occurring in a specific instance); now the answer of this problem is *yes* if there is some subset of SAT-instances for which the first one has $\mathcal{M}_1$ as model, and so on ... (i.e. for at least one guessed data-structure). Hence, the cardinality of satisfiable formulae is at least i. With binary search the exact number can be determined, and since we deal with decision problems the answer *yes* is given if this number is odd, and otherwise the answer *no* (even).

We will not present more problems here, especially because it is possible to fill books with $\mathbf{NP}$-complete problems and there are many for higher classes, too. These problems will be sufficient for our purpose and we will get to know some more problems in the next chapter where we deal with complexity results for non-monotonic logics.

3.10 NP-trees

$\mathbf{NP}$-trees are trees where at each node an oracle in $\mathbf{NP}$ is called for evaluating an instance of a problem. The solution is passed to its father node and used in the problem instance given in that node. Querying an $\mathbf{NP}$-tree (i.e. the root of this tree) is a decision problem which is cleary settled in Δ_2^P. However, Gottlob [1995c] proved that it is possible to show membership in $\Delta_2^P[O(\log n)]$ with a rather involved proof. Some intuition of this proof is stated in Section 7.5; in this section we restrict ourselves to presenting a formal definition of $\mathbf{NP}$-trees:

Definition 3.10.1 ($\mathbf{NP}$-Tree) *An $\mathbf{NP}$-tree is a triple $\langle Var, G, F_R\rangle$ consisting of*

- *a set of propositional variables $Var = \{v_1, \ldots, v_n\}$, called the linking variables;*
- *a tree $G = \langle V, E\rangle$, the nodes $V = \{F_1, \ldots, F_n\}$ representing propositional formulae. We identify the name F_i of a node with the name of the formula represented by the node. Each three-valued propositional formula F_i, in addition to some "private" variables not appearing in any other node, contains the following linking variables: $\{v_j | \langle F_j, F_i\rangle \in E\}$.*
- *A distinguished terminal node (root) F_R, for some $R \in \{1, \ldots, n\}$, called the result node.*

For each $\mathbf{NP}$-tree the truth value assignment σ to the propositional linking variables $v_1, \ldots, v_n$ is defined as follows. $\sigma(v_i)$ is true if F_i' is satisfiable, where F_i' is a formula where each linking variable v_j is replaced by $\mathbf{t}$ iff $\sigma(v_j) = true$ and by $\mathbf{f}$ iff $\sigma(v_j) = false$; otherwise $\sigma(v_i)$ is false. The result value of an $\mathbf{NP}$-tree is $\sigma(v_R)$.

3.11 Logical Characterisations

The subdiscipline of database theory and finite model theory dealing with the description of the expressive power of query languages and related logical formalisms via complexity classes is called *descriptive complexity theory*. One of the early results was Fagin's theorem stated below.

By the expressive power of a query language one means the set of all queries expressible in this language. Query languages can be languages used in database systems or formalisms such as first-order or second-order logic over finite structures as used in finite model theory.

Fagin [1974] showed that hard problems of complexity classes can be captured by some kind of logic. For instance, as we already mentioned, $\mathbf{NP}$ captures exactly all kind of (graph-theoretical) problems which are expressible in second-order existential logic. The class of all graph-theoretical properties expressible in Horn existential second-order logic with successor is precisely $\mathbf{P}$.

Any property ϕ expressible in Horn existential second order logic obeys a powerful zero-one law: If all graphs on n nodes have equal probability, then the probability that a graph with n nodes satisfies ϕ is either asymptotically zero or one as n grows to infinity. Hence, trivial problems like determining if a graph has an even number of edges (probability $1/2$) are not expressible in this logic unless we introduce the successor relation!

In the next chapter we highlight some known complexity results for non-monotonic reasoning and give some intuition thereof before we present our new results in the subsequent chapters.

Chapter 4

Complexity of Non-monotonic Reasoning

4.1 Model Checking vs. Reasoning

Definition 4.1.1 *Model Checking: Given an interpretation $\mathcal{M}$ and a theory T, check whether $\mathcal{M} \models T$.*

Definition 4.1.2 *Reasoning: Given a theory T and a formula F, check whether $T \models f$.*

The second problem often is more complex than the first one. Consider classical propositional logic. Given a propositional Herbrand interpretation, one can easily verify in linear time by substituting the variables with truth values, if the given interpretation indeed is a model of F (this is even in $\mathbf{NL}$).

However, the propositional reasoning problem, which is due to the deduction theorem, equivalent to $\models T \Rightarrow F$, is $\mathbf{coNP}$-complete (it is the "dual" problem of *not satisfiable*; dual problems are often, but not always, in the same complexity class!)

In other formalisms like default logic one has to introduce a notion of inference and a model relation, respectively, to study and to compare the complexity of these problems.

In default reasoning (and many other non-monotonic formalisms) two kinds of inference are distinguished: Brave reasoning and cautious reasoning (as introduced in Chapter 2). Let $E_{\langle D,W\rangle}$ denote an extension of a default theory.

$$\textit{Brave reasoning: } W \models_D F \text{ iff } \exists E_{\langle D,W\rangle} \text{ with } E_{\langle D,W\rangle} \models F$$
$$\textit{Cautious reasoning: } W \models_D F \text{ iff } \forall E_{\langle D,W\rangle} \text{ with } E_{\langle D,W\rangle} \models F$$

Cautious reasoning is not the converse problem of brave reasoning, but the dual problem to the converse problem of brave reasoning. Hence, in general cautious reasoning does not need to be on the same level of the hierarchy and can be in the converse class. For instance, for instance stationary default theory, cautious reasoning is one level easier than brave reasoning [Gottlob, 1995].

4.2 Complexity Results for Non-Monotonic Reasoning

Classical propositional reasoning is $\mathbf{coNP}$-complete, the same holds for single agent monotonic epistemic logics in case of S5 and KD45, but not for other epistemic logics which are $\mathbf{PSPACE}$-complete (for instance for S4). On the other hand, in the multi-agent case, things get worse for S5 and KD45, too, which also turn into $\mathbf{PSPACE}$-complete problems [Halpern and Moses, 1992].

So, what happens in case of non-monotonic logics – do complexity issues get more involved than in monotonic cases? In general, it holds, due to a context-based way of reasoning, that non-monotonic reasoning is one level harder than the classical one, i.e. settled on the second level of the polynomial hierarchy.

This, on the one hand, has generated interest in finding limitations which are one level easier or even tractable, and on the other hand, trying to find translations between several formalisms on the same complexity level.

However, this is not always straightforward; for example default logic treats prerequisites in a very different way than autoepistemic logic does. Hence, as Gottlob [1995b] proved, these two cannot be translated into each other in a modular way, i.e. that the set W can be translated independently, or in a strong sense, that each default rule can be translated independently into a modal premise.

In some settings it was possible to show that, in some sense a formalism is more expressive because it is possible to embed another one into it. For instance, logic programming with the stable models semantics can be embedded into default logic, or logic programming with the answer set semantics into disjunctive default logic. However, in both cases a vice-versa-translation is not possible although reasoning is in both cases of the same complexity (but model checking for instance is not!).

Some variants of well-known formalisms have been explored which are of a slightly easier complexity, like stationary default logic with an easier brave reasoning problem, or three-valued AEL for which we derive some complexity results in this report.

Moreover, model checking is sometimes a viable alternative for reasoning. We derive, based on a paper of Liberatore and Schaerf [1998], new complexity results for model checking for default logic. Rosati [1999] introduced a different notion of model checking and derived favourable results, too. His notion is based on *extension checking* (like incomplete models in answer set semantics of logic programming) while ours is based on *extension finding*. More attention to the comparison of both notions will be paid in Section 5.8.

4.3 Default Logic: Complexity

We only treat the case of propositional default theories here w.r.t. complexity issues. In default logic, there are two different sources of complexity. Firstly, in general it is necessary (say, for brave reasoning) to guess a set of generating defaults and to test if it forms an extension – which can be done in polynomial time, but uses an $\mathbf{NP}$-oracle for satisfiability and inference checks.

	General	Semi-Normal	Normal
General	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete
Prerequisite-free	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{P}$-complete

Table 4.1: *Brave Default Reasoning in Disjunction-free Default Theories*

From an historical point of view, initially limited cases of default logics w.r.t. computational complexity have been treated [Selman, 1990; Kautz and Selman, 1991; Stillman, 1990] and embeddings of some restrictions into propositional logic tried [Ben-Eliyahu and Dechter, 1991]. Selman and Kautz [1990] also proposed a related concept of model-preference defaults. The main results of these papers were that default reasoning is at least as hard as propositional reasoning unless one puts very grave constraints upon a default theory.

The cited authors analysed several very special cases of default theories. However, the restrictions we are interested in mostly are disjunction-free default theories (and an ample superclass, EHNF theories) as defined in Section 5.4. Disjunction-free default theories are theories where only the junctors of negation and conjunction appear, and furthermore allowing negation just to be inflicted on atoms.

Table 4.1 summarizes the complexity of brave reasoning in disjunction-free default logics which are arbitrary, semi-normal or normal and with or without prerequisites. Of course, if we limit default theories to disjunction-free theories, we also need to limit which formulae are allowed as input (namely disjunction-free formulae, too!). Cautious reasoning is for Reiter's default logic in the converse class, i.e. $\mathbf{coNP}$-complete in the general disjunction-free case.

Consider an arbitrary disjunction-free default theory. To show membership in $\mathbf{NP}$, a Turing machine initially guesses a set of generating defaults. Since no disjunctions are involved, all steps in the quasi-inductive fixpoint computations do not need an oracle for satisfiability, hence all checks can be done in polynomial time. Hardness is in the above mentioned papers shown for prerequisite-free semi-normal default theories, and a normal default theory with prerequisites. However, a reader who carefully studies the proofs in Chapters 5 and 6 w.r.t. model checking, which are somehow more involved proofs, can easily find similar (easier) proofs for reasoning.

Selman [1990] additionally exhibited some tractable cases, and found out that brave reasoning with disjunctive default theories being both normal and prerequisite-free can be performed in polynomial time. However, Stillman proved that even if W is only allowed to be a Horn theory, instead of a conjunctive formula, reasoning gets $\mathbf{NP}$-hard, too.

Selman also presented some other interesting cases, like unary default theories (prerequisite limited to a single literal, consequence limited to a single literal, justification either to the consequence in case of a negative literal, and to a conjunction of the consequence and a negated literal else), disjunction-free ordered default theories, where a partial order is introduced, unary ordered theories, Horn default theories, which have only a conjunction of positive literals as prerequisites and justification and consequence are single literals (i.e. the form is like a Horn clause, but not to be confused with our ample class of default theories in extended Horn normal form).

Recall that we apply the usual notion of completeness w.r.t. is polynomial many-one transformability, however, for proving $\mathbf{P}$-completeness we use logarithmic space reductions.

Σ_2^P-completeness of brave default reasoning was finally proved by Gottlob [1992], and independently by Stillman [1992]. For the reduction they used the well known problem of $\mathrm{QBF}_{2,\exists}$ as presented in Section 3.9.

In this problem, one needs to guess an evaluation of the p_i such that for all evaluations of the q_j the formula $F(p_1, \ldots, p_i, \ldots, p_n, q_1, \ldots, q_j, \ldots, q_m)$ is true. Guessing such an evaluation can easily be represented with two prerequisite-free normal defaults, for each propositional variable:

$$\frac{: p_i}{p_i}, \frac{: \neg p_i}{\neg p_i}$$

With an additional normal default

$$\frac{F : x}{x}$$

(and an empty set W) one can easily see that the literal x is in at least one extension exactly if such a truth assignment exists which makes F true (otherwise the prerequisite cannot be applied).

Indeed, this proof shows that the problem is Σ_2^P-hard even if a single literal is derived and even in a normal default theory, because even there the machine has exponential choices to construct a set of generating defaults. It is even possible to modify the proof to show that if the theory is additionally prerequisite-free, the problem remains Σ_2^P-hard. Just drop the final default! It is not needed – and just ask the question if F is in at least one extension (i.e. the consequence of the base which is a collection of p_i). Or, if you want to show that it holds even then for a single literal consider $W = \{F \Rightarrow x\}$.

With similar hardness proofs it is shown that cautious reasoning is Π_2^P-complete and that the problem to decide if a (not-normal, since normal always have an extension) default theory has an extension is Σ_2^P-hard, too.

Membership of this problem is, with the use of the quasi-iterative characterisation of extensions, straightforward: just guess a set of generating defaults and in each step use two oracle calls, one for satisfiability and one for inference and a final oracle call if $E \models f$, where f is an arbitrary given propositional formula. Hence the problem is in Σ_2^P.

Since default logic can be considered as a fragment of the non-monotonic modal logic $\mathbf{N}$ as introduced earlier, hardness results immediately carry over to $\mathbf{N}$. Membership of this problem in Σ_2^P was shown by Gottlob by modifying a finitary characterisation of expansions of AEL due to Niemelä [1991].

As cautious reasoning is Π_2^P-complete and brave reasoning is Σ_2^P-complete, an interested reader could think about giving a proof of the exact complexity of reasoning in the case someone provides the additional information that a given default theory has exactly one extension.

4.4 Complexity of Reasoning with Weak Extensions, Disjunctive Default Logic and Stationary Default Logic

Many variants of default logic have been introduced, some because of the arguments given in Section 2.5.8 that default logic does not fulfill some of the principles a non-monotonic logic should fulfill, some with the goal in mind to make computational tasks like cautious reasoning easier.

One of the most famous variants of default logic is default logic with weak extensions [Marek and Truszczyński, 1993] as presented in Section 2.5.4. Weak extensions coincide with the objective parts of stable expansions of a corresponding autoepistemic theory as explained later.

Since for prerequisite-free default theories extensions and weak extensions coincide, we get all the hardness-results, too, for weak extensions. Furthermore, membership can be proved in a straightforward way, too – just guess a set of generating defaults and check with two oracle calls if it fulfills the equation.

Stationary default extensions were introduced by [Przymusinska and Przymusinski, 1995] and a complexity characterisation was given by Gottlob [Gottlob, 1995]. Stationary default logic uses a different concept to generate extensions. It is based on a fixpoint iteration, too, however, extensions are fixpoints of the operator $\Gamma^2_{\langle D,W\rangle}$ instead of $\Gamma_{\langle D,W\rangle}$ and being a subset of $\Gamma_{\langle D,W\rangle}$ (remember that the operator Γ is anti-monotonic).

Stationary default logic has some desirable properties: every default theory has at least one stationary extension, there is a smallest stationary extension (although often not the one one would desire but as soon as contradictory defaults are introduced, this is $cons(W)$), it is cumulatively monotonic, it has close relationship to logic programming with the well-founded semantics, and the cautious semantics coincides with the least extension.

Hence, for cautious reasoning it is just necessary to compute the smallest extension, which is a subset of all other stationary extensions. Therefore, cautious reasoning gets easier and can be shown to be Δ_2^P-complete and in the case of normal default theories, $\Delta_2^P[O(\log n)]$-complete. The membership proof for normal default theories is rather involved, as it uses the techniques of data structure guessing - we will not go into details here, as we use such techniques in one proof in Section 5.3, too, and improve it there due to applying data structure guessing in several steps.

Disjunctive default logic [Gelfond *et al.*, 1991] changes the syntax of the rules, allowing an effective disjunction like in logic programming with the answer set semantics. Based on the definition of a reduct we get the usual extension characterisation for the case of no effective disjunctions (as explained in Section 2.5.4)

Hence, hardness results immediately carry over. Membership can easily be seen to remain in Σ_2^P due to the characterisation of extensions. However, disjunctive default logic is different w.r.t. limiting it to disjunction-free theories or literals only ([Eiter and Gottlob, 1995]). For this variant of default logic reasoning does not get one level easier, but remains Σ_2^P-complete and Π_2^P-complete, respectively. Hence, effective disjunction introduces a third kind of complexity, which is, however, closely connected to one of the other kinds since in the general case it is "parallel" to it and does not raise the complexity.

4.5 Complexity of AEL and Non-monotonic Modal Logics

Since default logic with weak extensions can be considered as fragment of AEL, hardness results immediately carry over to AEL. Membership in Σ_2^P for brave reasoning can be shown with the finite characterisation of Niemelä [1991] as presented in [Gottlob, 1992]. In case of AEL with parsimonious expansions brave reasoning is Σ_3^P-complete [Eiter and Gottlob, 1992].

As explained in Section 2.5.6, non-monotonic modal expansions are similar to autoepistemic ones, however they do not use positive introspection in the fixpoint equation, but a different consequence operator, based on a consequence relation of some modal logic.

$$\Delta = cons_S(\Sigma \cup \neg L\overline{\Delta}).$$

Gottlob [1992] proved that if S is a modal logic with N $\subseteq$ S $\subseteq$ S4F (with F $= A \wedge \Diamond\Box B \Rightarrow \Box(\Diamond A \vee B)$ added to S4) then brave reasoning and determining the existence of an extension is Σ_2^P-hard, and cautious reasoning Π_2^P-hard. Upper bounds depend on the the complexity of the underlying monotonic logic.

Although the complexity of reasoning in $S4$ is complete for $\mathbf{PSPACE}$, surprisingly, the complexity of cautious reasoning in non-monotonic $S4$ is only Π_2^P-complete [Schwarz and Truszczyński, 1992].

The non-monotonic variant of $S5$ coincides with S5, i.e. a formula can be inferred in $S5$ with a given theory if it is a consequence of the single extension of the theory.

4.6 Closed World Assumption: Complexity

We only consider the propositional variant of closed world reasoning in this chapter as discussed in Section 2.5.1. Closed world means that everything we cannot derive is assumed to be false.

The standard closed world assumption CWA$(T) = T \cup \{\neg p | T \not\models p\}$ is not always consistent (p is a ground atom), since if $T = q \vee r$ we run into inconsistency.

Therefore, an additional task to reasoning is the task if, given a theory T, CWA(T) is consistent. Since CWA can be equally characterised semantically it holds that CWA(T) is consistent exactly if T has a unique minimal model (with minimality defined as the usual subset relation) [Eiter and Gottlob, 1993]. This problem is known to be $\mathbf{coNP}$-hard and in $\Delta_2^P[O(\log n)]$, however, the exact complexity is still unknown.

Table 4.2 presents the complexity of reasoning with the in Section 2.5.1 considered closed world assumptions [Eiter and Gottlob, 1993].

CWR-rule	Complexity	Lower Bound	Upper Bound
CWA(T) $\models F$		$\mathbf{NP}$-hard	in $\Delta_2^P[O(\log n)]$
GCWA(T) $\models F$		Π_2^P-hard	in $\Delta_3^P[O(\log n)]$
EGCWA(T) $\models F$	Π_2^P-complete		
CCWA($T; P, Z$) $\models F$		Π_2^P-hard	in $\Delta_3^P[O(\log n)]$
ECWA($T; P, Z$) $\models F$	Π_2^P-complete		

Table 4.2: *Complexity of Closed World Reasoning*

Remember that ECWA in the propositional case coincides with circumscription, hence they admit the same complexity results (Section 2.5.1).

For more complexity results on other variants like *Iterated CWA* and *Disjunctive Database Rule DDR* we refer to [Eiter and Gottlob, 1995].

4.7 Complexity of Logic Based Abduction

In Section 2.5.2 we considered three tasks: consistency, relevance and necessity of an hypotheses. Eiter and Gottlob [1995b] exhibited complexity results for several $\leq$ preference relations and several restrictions to a theory (clausal form, Horn theory, definite Horn theory).

Without an additional preference criterion, and for arbitrary T the task of deciding if the set of solutions is non-empty is Σ_2^P-complete. Deciding if a hypotheses is relevant in this case is Σ_2^P-complete, too, and deciding necessity is Π_2^P-complete. Introducing other preference criteria in some cases slightly increases the complexity sometimes even up to the third level of the polynomial hierarchy.

Syntactically restricting T to clausal form does not result in a decrease of complexity. However, if T is Horn, one source of complexity is successfully attacked and the problem gets one level easier. If T is definite Horn (exactly one positive literal per clause), then for some cases the problem is even polynomial and the symmetry between necessity and relevance can no longer be maintained – in case of definite Horn theories with the subset-relation, relevance is still $\mathbf{NP}$-complete while deciding necessity is polynomial.

4.8 Complexity of Logic Programming Formalisms

In logic programming formalisms (like in the database query language Datalog) four interesting complexity issues can be discussed: *data-, expression (program-) -, combined -, and ground complexity*.

A (disjunctive) database is a tuple $DB = (\Pi, E)$ where Π is a program and E is a set of (disjunctive) ground facts. E represents the extensional part, whereas Π are the inference rules (remember that logic programming usually is a first-order formalism), called the intensional part of the database DB. Of course, E, too, is a program, and this is just introduced for distinctive complexity issues.

The data complexity is the complexity of (cautious) reasoning with a fixed program Π, the expression complexity the complexity of reasoning with E fixed, and in the combined complexity both E and Π are allowed to vary freely. Finally, the propositional (ground) complexity is the complexity of a propositional program (this coincides with programs without variables). However, observe that a ground instantiation of a program DB is usually exponentially longer (as input).

We only discussed logic programming without function symbols; allowing function symbols would introduce undecidability in the general case [Dantsin *et al.*, 1997].

Usually, expression complexity is exponential in data complexity and hardness can be proved with complexity upgrading techniques (conversion lemma, succinct inputs - for details, for instance refer to [Veith, 1996]).

4.8.1 Stable Model Semantics

In the following, the term *logic program* normally labels *disjunctive propositional logic program* and *plain* program a non-disjunctive program. We call a program *normal* if it is not-free. The following theorems are well-known and only presented with some illustrative proofs for your convenience.

Lemma 4.8.1 *Model Checking with a given propositional interpretation M and a logic program P ($M \models P$) for disjunctive logic programming with stable model semantics is in* $\mathbf{coNP}$.

Proof. Consider the following non-deterministic Turing machine T: First it checks in polynomial time if $M \models P^M$ (or in the usual guess-and-check procedure do this check after the guessing of another model). If yes, then all T needs to check is if the given model M belongs to the set of minimal models of P^M, i.e. the question if $\forall N. N < M \Rightarrow\ N \not\models P^M$. This is the complementary problem of the question if $\exists N. N < M \wedge\ N \models P^M$ – hence the NDTM guesses an interpretation N and checks that it is a smaller model than M. This problem in $\mathbf{NP}$, hence the converse problem is in $\mathbf{coNP}$. $\square$

Lemma 4.8.2 *Model Checking with a given propositional interpretation M and a non-disjunctive logic program P ($M \models P$) with the stable model semantics is in* $\mathbf{P}$.

Proof. All clauses of P^M have at most one literal in the head and are due to the definition of P^M *not*-free; such clauses can be represented as disjunctions of literals with at most one positive literal. The question if $\exists N < M. N \models P^M$. can be decided in polynomial time if all clauses are Horn-clauses since then P^M is a Horn theory and determining the smallest model of a Horn theory is polynomial. $\square$

Lemma 4.8.3 *Model Checking with a given model M and logic program P ($M \models P$) for disjunctive logic programming with stable model semantics is* **coNP***-hard, even if P is* not*-free.*

Proof. We use a reduction of the problem if a formula F in conjunctive normal form is not satisfiable (this problem is **coNP**-hard): $F(x_1, \ldots, x_l) = F_1 \wedge \ldots \wedge F_n \wedge (\neg x_1 \vee \cdots \vee \neg x_l)$ where F_k are disjunctions of literals $F_k = a_{k,0} \vee a_{k,1} \vee \ldots \vee a_{k,i} \vee \neg a_{k,i+1} \vee \ldots \neg a_{k+m}$ with all $a_{k,i}$ are positive literals, each positive literal is one of the variables $x_1, \ldots, x_l$.

Let P be the logic program consisting of $n \cdot l$ clauses of the form: $x_j \vee a_{k,0} \vee a_{k,1} \vee \ldots \vee a_{k,i} \leftarrow a_{k,i+1} \wedge \ldots \wedge a_{k,m} \quad (= F_k \vee x_j)$. This transformation can be done in polynomial time. We now only need to show that $M = \{x_1, \ldots, x_l\}$ is a model iff F is unsatisfiable:

The following statements are equivalent:

- M is a minimal model of P
- M is a minimal model of the conjunction of all $F_k \vee x_j$
- M is a model of $F \vee (x_1 \wedge \ldots \wedge x_l)$
- F is unsatisfiable (since the only other possibility that F is only satisfiable with $\{x_1, \ldots, x_n\}$ is ruled out due to the construction of F).

Hence, M is a minimal model of P exactly if F is not satisfiable. □

In general, even approximating the stable model semantics is a hard problem (cf. [Gottlob and Truszczyński, 1996]).

Brave reasoning (i.e. if a given formula follows at least of one model of a given logic program) with a non-disjunctive program is **NP**-complete - this is easy to see, since if no particular model is given, the minimal model needs to be constructed. Simply guess one interpretation M and check in polynomial time (since model checking is polynomial) that it is the minimal one for P^M and that this interpretation is a model of the given arbitrary formula f which can be decided in polynomial time, too. If a non-disjunctive program is not-free, too, then brave reasoning is even polynomial since minimal model finding is polynomial due to $P^M = P$.

Brave reasoning with disjunctive but not-free programs is already complete for Σ_2^P. However, determining the consistency of a disjunctive not-free program is still **NP**-complete since this question is equivalent to the question if the program rewritten as SAT clause is satisfiable (if it has a model, it has a minimal one).

Hence, in disjunctive programs with or without negation, brave reasoning is Σ_2^P-complete; and with similar thoughts cautious reasoning is Π_2^P-complete. The problem becomes harder since there is now no unique model. To view the converse problem, if a formula cannot be inferred of such a program: Guess an interpretation, check that it is minimal by calling an oracle in **coNP** that for all other interpretations none is a smaller one, and then check in polynomial time that this model is not a model of the given formula f. For detailed proofs of membership and hardness, we refer to [Eiter and Gottlob, 1995].

As soon as negation occurs, also consistency checking of a disjunctive program turns Σ_2^P-hard since, due to the necessary construction of P^M, several different SAT instances need to be distinguished.

4.8.2 Partial stable model Semantics

Partial stable model semantics is a three-valued semantics containing, like the answer set semantics, undefined literals, distinguishing a positive and a negative set of atoms. For details, for instance, refer to [Eiter and Gottlob, 1995].

Lemma 4.8.4 *Model Checking with a given interpretation M and logic program P ($M \models P$) for disjunctive logic programming with partial stable model semantics is in* $\mathbf{coNP}$.

Proof. For partial stable semantics P^M can be formed in polynomial time (*not* p is substituted with 0, 0.5 or 1 depending of the truth value of p in M – M is a partial Herbrand model leaving some variables undefined). $M \models P^M$ can be done in polynomial time (just substitute the truth values of all variables). Now the same question as above remains, namely if M is a minimal model of P^M. M is a minimal model of P if its positive set is minimal and its negative set is maximal. The problem if a smaller model exists is in $\mathbf{NP}$: just guess a model with smaller positive part and bigger negative part and prove in polynomial time if it is a model of P^M. So, the complementary problem is in $\mathbf{coNP}$ and therefore the whole task is in $\mathbf{coNP}$. □

Lemma 4.8.5 *Model Checking with a given model M and a logic program P ($M \models P$) for disjunctive logic programming with partial stable model semantics is* $\mathbf{coNP}$*-hard.*

Proof. In the hardness-proof of model checking for stable models semantics (Lemma 4.8.3)just add the clauses: $\leftarrow x_j$, *not* x_j. Then all partial models are total models and coincide with the models of the stable model semantics. Deciding whether a given model is a minimal model of P^M is $\mathbf{coNP}$-hard. Hence, there is a logic program for which model checking with the partial stable model semantic is $\mathbf{coNP}$-hard. □

4.8.3 Closed World Setting

$EGCWA(P) = \{I \in MM(P)\}$, P can be viewed as *not*-free. Hence the question if $M \models_{EGCWA} P$ is equivalent to the question if $M \in MM(P)$. This is for disjunctive logic programs shown as above to be $\mathbf{coNP}$-complete, hence this problem is $\mathbf{coNP}$-complete, too.

The same holds for model checking with ECWA(P)=CIRC(P) which is in the general case, and in the case of disjunctive logic programs $\mathbf{coNP}$-complete. Cadoli [1992] shows for which cases model checking is polynomial and for which it is $\mathbf{coNP}$-complete. Refer to Section 4.6 for complexity issues about reasoning.

Model Checking with $GCWA$ is probably more complex, because

$$GCWA(P) = \{M \in M(P) \mid \forall x \in A(P)\; MM(P) \models \neg x \;\Rightarrow\; M \models \neg x\}.$$

Here one needs to know the set of *all* minimal models to test if M is a model.

4.8.4 Perfect Model Semantics

To name another semantics, we mention the perfect stable semantics for not stratifiable programs which introduces another preference relation on the models.

A priority relation on atoms is introduced, informally $x < y$ means that y has higher priority than x. The preference order $\ll$ is defined by $M_1 \ll M_2$ iff $M_1 \neq M_2$ and for each $x \in M_1 - M_2$ there exists $y \in M_2 - M_1$ such that $x < y$.

$PERF(P) = \{I \in M(P) \mid \forall N \in M(P)\; N \not\ll I\}$. W.r.t. given I, P, the test with the stable model semantics is $\mathbf{coNP}$-complete. If $N \not\ll I$ can be tested in polynomial time, the analogous problem if a smaller N exists is therefore also $\mathbf{coNP}$-complete. Hence, membership in $\mathbf{coNP}$ can be shown with guessing a data-structure $\langle M, N \rangle$. Guessing data structures is a convenient and sometimes powerful method explained in more detail in Chapter 5.

4.8.5 Answer Set Semantics

Answer set semantics is a semantics for programs which allow, additionally to negation-by-failure, a classical negation (Extended Disjunctive Logic Programs, short EDLP). EDLPs furthermore treat disjunctions in a somehow different way (called effective disjunctions) – for more detailed information, see Chapter 6 where the close relationship between EDLPs and disjunctive default logic is explored.

The answer sets of a logic program are defined as follows - $ANSW(P) = \{S \mid S$ is a consistent answer set of $Red(P, S)\}$ where $Red(P, S)$ is the EDLP obtained from P by deleting each rule that has $(\textit{not } l)$ in its body for some literal $l \in S$ and deleting all subformulae of the form $(\textit{not } l)$ in the bodies of the remaining rules. Any context S which is an answer set of $Red(P, S)$ is an answer set of S. Answer sets are defined as minimal and closed sets, given a context S which is a subset of all literals (i.e. an incomplete model containing positive, negative or no information at all about a literal). An answer set is closed if for each rule of the program it holds that if the body is in S, then at least one element of the head is in S.

It is well known that M is a stable model of Q iff M is a consistent answer set of the translation of Q where each disjunction is replaced with an effective disjunction. Therefore, answer set checking is $\mathbf{coNP}$-hard.

A proof of membership of answer set checking can be illustrated as follows: given S, P, first calculate $Red(P, S)$. This can be done in polynomial time. Check that there exists no subset of S which is closed under P. This is the complementary question of the question whether $\exists S'\, S' \subseteq$

$S \wedge S'$ is closed under P, which is clearly in $\mathbf{NP}$ (guess a S' and test in polynomial time), therefore the original question is in $\mathbf{coNP}$. Hence, the complete procedure is in $\mathbf{coNP}$.

Observe that answer set checking can be both seen as kind of incomplete model checking or as kind of extension checking. If carrying over this notion to default logic, answer set checking is the same as extension checking (see [Rosati, 1999] for complexity results; he calls this notion model checking); however, in later chapters we use model checking for default logic in that sense that we consider total Herbrand interpretations (like Liberatore and Schaerf [1998]). For more details about these matters, refer to the later chapters – of course, complexity results for both notions of model checking are interesting and worth exploring.

4.8.6 General Logic Programming and Datalog

Usually logic programming is a first order formalism as mentioned above – the complexity results for the propositional case carry over to the issue of data complexity of datalog.

Furthermore, some special cases are worth mentioning: It has been proved that for special formulae reasoning gets easier – in the case of conjunctive queries to a disjunctive program, the problem is only $\mathbf{coNP}$-complete [Imielinski and Vadaparty, 1989]. In [Eiter and Veith, 1999] it is shown that the complexity is $\Delta_2^P[O(\log n)]$-complete in the case of disjunctive queries.

In general, the combined complexity of reasoning in a non-disjunctive datalog program is **DEXPTIME**-complete. Expression complexity allows for a slightly sharper upper bound than combined complexity (**DETIME** vs. **DEXPTIME**). See [Dantsin *et al.*, 1997] for details.

4.9 Translatability Issues

We have presented a lot of formalisms, and you may be asking yourself which one is more powerful, and if these formalisms could be interrelated and compared with each other – maybe one can be embedded into an other one? If such an embedding is possible, is it possible in polynomial time? How to compare extensions with for instance, expansions? And can each default or program clause be independently added in a translation by preserving the result?

One example of such an embedding is to translate disjunctive logic programming into disjunctive default logic - for details see the next subsection. Another example is prerequisite-free default logic which can be embedded in a very straightforward way into autoepistemic logic by translating $(: f/g)$ as $\neg K \neg f \Rightarrow g$. In this case each default can be independently translated - which is called modularity. However, in general, default logic can only be embedded into autoepistemic logic in a non-modular way [Gottlob, 1995b].

There are different possibilities to compare formalisms with each other – one is the concept of representational succinctness, another one that of pfm-translations. Both notions allow to create hierarchies of formalisms, in the first option of succinctness, in the second one of expressiveness.

For some results on translatability issues, refer to [Bidoit and Froidevaux, 1991; Inoue and Sakama, 1993].

4.9.1 Representational Succinctness

Definition 4.9.1 *[Gogic* et al.*, 1995] Let A and B be formalisms for knowledge representation, Φ_A, Φ_B be knowledge bases, i.e. specific theories like default theories: We say that B is at least as (equally or more) succinct as A, symbolically $A \rightsquigarrow B$ iff $\forall \Phi_A \in A$, Φ_B has an equivalent knowledge base in B which is only polynomially longer. (Equivalent here means that it has the same variables and the same set of models.). B is more succinct than A iff $A \rightsquigarrow B$ and $B \not\rightsquigarrow A$, i.e. there is a knowledge base Φ_B in B for which there is only an exponentially longer knowledge base in A.*

The size of Φ_B has to be polynomially bounded in the size of Φ_A, but we have no provision that Φ_B is efficiently computable, given Φ_A. That means, we just have to show that there is a polynomial bound in the *size of of the theory*, but we do not need a polynomial time way to construct it.

Succinctness is independent of the computational complexity of the inference problem. However, it is closely connected to the model checking problem (and in the case of default logic it uses the notion of model checking we analyse in Chapter 5).

Hence, if for two formalisms A and B, *model checking* which is at different complexity levels (succinctness is independent of computational complexity as far as reasoning is concerned, but not as far as model checking is concerned; various formalisms may have the same reasoning complexity, but different model checking complexity), then $A \rightsquigarrow B$ if model checking in B is more complex.

Therefore we can directly argue, based on the results of [Liberatore and Schaerf, 1998] and our results, that $DLP \rightsquigarrow DL$, even for normal default theories (DLP is disjunctive logic programming, DL is default logic; model checking for DLP is $\mathbf{coNP}$-complete with several semantics; model checking for default logic is Σ_2^P-complete in the general case, and $\Delta_2^P[O(\log n)]$-complete if restricted to normal theories).

If both formalisms are on the same complexity level w.r.t. reasoning a proof needs to be given – it could even be possible that they are not comparable with each other as far as succinctness is concerned [Gogic *et al.*, 1995].

4.9.2 Expressive Power

Expressive power is measured by the notion of a polynomial, faithful and modular translation function. Depending on the definition of *faithful* and *modular*, it is possible to get different results.

Polynomiality: The time required to translate a theory into another one is polynomial in the length of the theory.

Faithfulness: The propositionally consistent extensions of the theory and the translated theory are in one-to-one correspondence such that extensions coincide up to the propositional language.

Modularity: The translation function is independent of the propositional subtheory $T \subseteq L$ of a non-monotonic theory.

Definition 4.9.2 *B is at least as (equally or more) expressive as A if a translation function exists mapping each Φ_A into a Φ_B which is polynomial, faithful and modular (pfm-function).*

Gottlob [1995b] disallows introduction of new variables when extensions are compared, whereas Janhunen allows it in his notion of faithfulness [Janhunen, 1998]. Hence, they both get different results: In Gottlob's notion autoepistemic logic is more expressive than default logic, in Janhunen's notion it is the other way round.

Expressiveness is not directly related with the problem of model checking as succinctness is. It is instead related with the problem of inference, in the following sense:

If reasoning in B is computationally harder (and not equally hard) than reasoning in A, then there is no polynomial transformation from A into B such that formula f can be inferred in Φ_A iff it can be inferred in Φ_B. Therefore there can be no polynomial translation function and hence no pfm-translation function. Hence: A is not at least as expressive as B. To find out if B is at least as expressive as A, we need to find a faithful and modular function (out of the polynomial functions we already know exists).

If reasoning in B is equally hard as in A, then anything can happen: B can be at least as expressive than A or/and A can be at least as expressive than B.

Due to these two arguments we can say: If A is at least as expressive as B and B is at least as expressive as A, then the reasoning tasks in A and B are exactly of the same complexity.

If we would like to show that default logic is at least as expressive as disjunctive logic programming, we need to clarify the notion of extensions in DLP - this can be done as usual identifying models of logic programs as extensions.

Remember that for both DL and DLP the inference problem (cautious reasoning) is Π_2^P-complete; however it is very easy to see that default logic is more expressive than disjunctive logic programming – it is easy to construct an embedding of DLP into DL, but it is not possible to do it vice versa – one way to show this is to use the model checking results of the next chapter. We will continue this discussion in Section 5.8.

4.10 Summary and Stable Class Semantics

In general, several settings of non-monotonic propositional reasoning are one level harder than classical propositional reasoning. Non-monotonic theories introduce an additional independent source of complexity due to context-based reasoning.

Variants which make problems easier either restrict the syntax of formulae to need no oracle calls for SAT or restrict the syntax of the non-monotonic logic – restricting both may allow for polynomial cases.

However, there are also settings of non-monotonic reasoning which are far more complex than the presented ones. Consider the stable class semantics proposed by Baral and Subrahmanian [1992].

This is a semantics proposed for logic programming and default logic: It does not use the concept that extensions are formed based on fixed-points $\Gamma(S) = S$, but uses sets of bases of extensions (or in case of logic programming, sets of interpretations) employing the concept of fixed-point cycles – for instance consider the default theory containing the single default $\frac{:\neg b}{b}$. Starting with the context $\{b\}$ yields $\emptyset$ and using the empty context results in $\{b\}$. Hence, this stable class is $S = \{\emptyset, \{b\}\}$.

All strict stable classes (i.e. smallest cycles) are members of S. The semantics $S(P)$ is defined as the union of all $\preceq$-minimal classes. Between two classes S_1 and S_2 holds $S_1 \preceq S_2$ iff $\forall c \in S_1\ \exists\, d \in S_2\ :\ c \subseteq d$. Hence, some elements are ruled out and $S(P)$ is defined as stable class semantics. Cautious reasoning is defined such that f can be derived of all elements of $S(P)$.

This problem is rather involved from the perspective of complexity as all cycles need to be considered, then all strict (smallest) cycles need to be detected, and afterwards all cycles which are not $\preceq$-minimal should be sorted out.

So, first, all stable classes need to be constructed, i.e. all those which are cycles not containing smaller ones:

- find all smallest cycles;
- cut all those which are not minimal w.r.t. $\preceq$;
- check if all elements of $S(P)$ are models of f.

Steps 2 and 3 are $\mathbf{coNP}$-steps (in 3: guess one set in $S(P)$ and check if f cannot be deduced).

However, task 1 is more difficult. Cycles can be of any size! Of each vertex only one arrow leaves (going to one other interpretation), however one vertex may have more than one incoming edge.

So, you need to consider each interpretation as the starting point of a possible cycle. Problem: There is an exponential number of interpretations, i.e. the graph has 2^n vertices!

The $\preceq$ relation furthermore adds some flavour, maybe making the problem a little bit unintuitive.

Example 4.10.1 *For instance, if we given four cycles*

$$\{\{p_1\}, \{p_2, p_3\}\}, \{\{p_2\}, \{p_1, p_3\}\}, \{\{p_3\}, \{p_1, p_2\}\}, \{\{\emptyset, \{p_1, p_2, p_3\}\},$$

then the fourth one is not $\preceq$ minimal. And $S(P)$ is hence the set

$$\{\{p_1\}, \{p_2, p_3\}, \{p_2\}, \{p_1, p_3\}, \{p_3\}, \{p_1, p_2\}\}.$$

We conjecture, due to the necessity of the first step, which cannot be replaced with some guessing steps, that this problem is $\mathbf{PSPACE}$-complete.

In the next chapters we present a wide variety of our new complexity results for default logic and autoepistemic logic.

Part II

Chapter 5

Complexity of Model Checking for Propositional Default Logics: New Results and Tractable Cases

Liberatore and Schaerf [1998] gave a proof that model checking for propositional normal default theories is in Δ_2^P and $\Delta_2^P[O(\log n)]$-hard. However, the precise complexity was left as an open problem. We solve this problem by proving that model checking for normal default theories is complete for $\Delta_2^P[O(\log n)]$. This is the class of decision problems solvable in polynomial time with a logarithmic number of calls to an oracle in $\mathbf{NP}$.

Additionally, in this chapter we analyse the computational cost of model checking w.r.t. weak extensions, stable expansions and **N**-expansions, and take a look at the complexity of model checking for disjunction-free default-theories. Furthermore, we show that not only for disjunction-free default theories, but also for a larger class of default theories, which we call *default theories in extended Horn normal form*, the complexity of model checking is, in the case of normal default theories, tractable. The concluding sections of this chapter are devoted to using the complexity results to draw some interesting conclusions on translatability issues already sketched in the previous chapter. In particular, there exists no pfm-function from default logic into logic programming unless $\mathbf{coNP} = \Sigma_2^P$.

5.1 The Model Checking Approach

The complexity of default reasoning, i.e. determining whether a given formula can be inferred from at least one extension of a given default theory (*brave reasoning*) or whether it can be inferred from all extensions of a given default theory (*cautious reasoning*) is already well understood in literature (Section 4.3). However, here we choose an alternative approach in search for a model-based representation by looking at the complexity of the model checking problem instead of the inference problem.

As Halpern and Vardi [1991] argue, model checking is a beneficial alternative simplifying reasoning tasks (for instance, in classical propositional logic, model checking can be done using an easy polynomial algorithm, however reasoning is $\mathbf{coNP}$-complete) and allowing for representing the agent's knowledge as a semantic structure instead of a collection of formulae and additionally, this approach introduces a kind of closed-world assumption. Furthermore, the complexity of model checking is closely related to the notion of *representational succinctness* [Gogic *et al.*, 1995] of non-monotonic formalisms (compare to 4.9.1).

Liberatore and Schaerf [1998] discuss model checking for several default logics and show that in general this problem is Σ_2^P-complete, even for semi-normal prerequisite-free default theories. In the case of normal default theories, model checking is easier than the corresponding reasoning task – they show that the problem is in Δ_2^P and $\Delta_2^P[O(\log n)]$-hard, and $\mathbf{coNP}$-complete if the defaults are also prerequisite-free. We strengthen their results with a non-trivial membership proof to a completeness-result for the class $\Delta_2^P[O(\log n)]$. To obtain this result in Section 5.3 (with a proof consisting of four steps) we state some propositions and lemmata in Section 5.2.

In Section 5.4 we deal with the complexity of model checking for disjunction-free default theories, namely theories where prerequisites, justifications and consequents are limited to be conjunctions of positive and negative literals, and introduce a class of default theories, called default theories in extended Horn normal form (EHNF), which does not raise the complexity of model checking w.r.t. disjunction-free default theories. Normal EHNF default theories admit a tractable model checking problem - this is an ample class of default theories and hence very useful in practical applications.

Additionally, we investigate the complexity of model checking when considering weak extensions of a default theory ()Section 5.5) and use these results to obtain complexity results for propositional model checking in autoepistemic logic and logic **N** in Section 5.7.

Furthermore, in Section 5.8, the complexity results are used to draw some interesting conclusions w.r.t. translatability issues. In particular, there exists no pfm-function from different variants of default logic into logic programming unless $\mathbf{coNP} = \Sigma_2^P$. Finally, Chapter 6 settles the complexity of model checking for disjunctive default logic and stationary default logic. In Chapter 8, the main complexity results of this and the ensuing chapters will be reviewed.

Let us formally define when an interpretation is a model of a default theory. In this chapter $\mathcal{M}$ denotes an interpretation whereas M denotes a Turing machine.

Definition 5.1.1 *An interpretation (valuation) $\mathcal{M}$ is a model of Δ iff $\mathcal{M}$ is a model of at least one (consistent) extension of Δ.*

The *model checking* problem for default logic is to decide, w.r.t. a given interpretation $\mathcal{M}$ and a given default theory Δ, whether $\mathcal{M} \models \Delta$ holds. Whenever we use the term *model* we refer to *propositional Herbrand model*. In the propositional case, a Herbrand model is the set of propositional variables that are *true* in a truth assignment. For convenience, we sometimes use the term *model* instead of *interpretation*. In what follows, $\mathcal{M}$ denotes an interpretation whereas M denotes a Turing machine.

Recently, Rosati [1999] introduced a different notion of model checking, which is a kind of *extension checking*, whereas our notion is in the general case based on *extension finding*.

Recall from Section 4.3 that Gottlob (and independently, Stillman) showed, that the complexity of the question whether a formula f is in at least one extension of a given default theory (called brave reasoning) is Σ_2^P-complete, even in the case of normal and prerequisite-free default theories, and even if f is a single literal (additionally, cautious reasoning is Π_2^P-complete [Gottlob, 1992; Stillman, 1992]).

5.2 Some Useful Tools

Definition 5.2.1 *A monotonic rulesystem $\langle R, W\rangle$ consists of a finite set*

$$R = \left\{\frac{\alpha_1}{\gamma_1}, \ldots, \frac{\alpha_n}{\gamma_n}\right\}$$

of propositional inference rules and a set W of propositional sentences.

Lemma 5.2.2 *Let $\langle D, W\rangle$ be a normal default theory. If the consequents of all defaults are jointly consistent and consistent with W, then there are no mutually incompatible defaults and thus there exists only one extension.*

Proof. Assume there are n defaults; in a normal default theory the consequent of each default is the same as its justification: $c(d_i) = j(d_i)$ for all defaults $d_i (1 \le i \le n)$.

We use the quasi-inductive base-operator characterisation of extensions and restrict it to normal default theories, i.e. to the case when $c(d) = j(d)$.

$$E_0 = W$$
$$E_{k+1} = E_k \cup \left\{c(d) \mid \frac{p(d) : j(d)}{c(d)} \in D \text{ with } E_k \models p(d) \text{ and } W \cup c(H) \not\models \neg j(d)\right\}$$

$E = cons(W \cup c(H))$ is an extension of a finite default theory with $|D|$ defaults if $W \wedge c(H) \equiv E_{|D|}$; and H is a generating set.

Since $W \wedge c(D)$ is consistent, and every extension is a subset of $cons(W \wedge c(D))$ and $c(d) = j(d)$ for each default, and each E_k is a subset of E, it always holds that $W \wedge c(H) \not\models \neg j(d)$.

Because we have shown that this condition is always fulfilled we can leave it out of the quasi-inductive definition.

$$E_0 = W$$
$$E_{k+1} = E_k \cup \{c(d) \mid p(d) : j(d)/c(d) \in D \text{ with } E_k \models p(d)\}$$

This turns the quasi-inductive characterisation into an inductive characterisation. At most at $E_{|D|}$ all defaults with applicable prerequisites have been used. There is no context in this characterisation any longer, hence there is only one extension: $E = \text{cons}E_{|D|}$ (Because no default blocks another default, this extension is created by all defaults with applicable prerequisites.). $\square$

Lemma 5.2.3 *Let $\langle R, W\rangle$ be a monotonic rulesystem (or a normal or justification-free default theory) with W and the consequents b_i of $R = \{\frac{a_1}{b_1}, \ldots \frac{a_n}{b_n}\}$ jointly consistent. Then at least k rules do not fire iff it is possible to choose a set $B \subseteq R$ with $|B| = k$ and k interpretations $\mathcal{I}_1, \ldots \mathcal{I}_k$ in such a way that*

1. $\forall j(1 \leq j \leq k) : \ \mathcal{I}_j \models W$
2. $\forall j\ \forall b \in R\backslash B : \ \mathcal{I}_j \models b$
3. $\forall a \in B\ \exists j : \ \mathcal{I}_j \not\models a$.

Proof. The statement that at least k rules do not fire is equivalent to the existence of a set B with $|B| = k$ rules which are not used. This is equivalent to the statement that it is not possible to infer the prerequisites of the rules of B from W and the use of the consequents $b \in c(R\backslash B)$ of $R\backslash B$. Of course, it is maybe impossible to infer some of the prerequisites of the rules in $R\backslash B$, too, therefore there are maybe more than k rules which do not fire – however, all that is needed is that *at least* k rules do not fire. Formally:

$$\forall a \in B : \quad W \wedge c(R\backslash B) \not\models a$$

Due to our requirements the conjunction of W with $c(R\backslash B)$ is always consistent (and hence forbids that a could be inferred trivially). Further we know that $v \not\models w$ is equivalent with the existence of an interpretation $\mathcal{I}$ with $\mathcal{I} \models v$ and $\mathcal{I} \not\models w$. In this case:

$$\forall a \in B\ \exists \mathcal{I} : \ \mathcal{I} \models W \wedge c(R\backslash B), \mathcal{I} \not\models a$$

Therefore:

$$\forall j : \quad \mathcal{I}_j \models W \wedge c(R\backslash B)$$

and

$$\forall a \in B \quad \exists j : \quad \mathcal{I}_j \not\models a$$

Finally, since $\mathcal{I} \models v \wedge w$ iff $\mathcal{I} \models v$ and $\mathcal{I} \models w$, we achieve that this is equivalent to:

$$\forall j : \quad \mathcal{I}_j \models W$$

$$\forall j \quad \forall b \in R\backslash B : \quad \mathcal{I}_j \models b$$

$$\forall a \in B \quad \exists j : \quad \mathcal{I}_j \not\models a$$

All steps of the proof can be carried out in the reverse direction, too, therefore we have shown equivalence of these both statements. □

The following proposition formalises a technique (binary search) that is well known in literature [Papadimitriou, 1993; Wagner, 1990].

Proposition 5.2.4 *Let $a(I_1, \ldots, I_n)$ be a function of n instances of a problem into $\mathbb{N}$, and let $a(I_1, \ldots, I_n)$ be polynomially bounded in n, i.e. $a(I_1, \ldots, I_n) \leq p(n)$ for some polynomial $p(n)$. If the problem of deciding $a(I_1, \ldots, I_n) \leq r$ is in $\mathbf{NP}$, then the computation of $a(I_1, \ldots, I_n)$ is possible in polynomial time requiring $O(\log n)$ oracle calls[1] (the corresponding decision problem, for instance if $a(I_1, \ldots, I_n)$ is odd, is hence in $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$).*

Proof. Start with calling the oracle for $r = \lceil p(n)/2 \rceil$. If the answer for $a \leq \lceil p(n)/2 \rceil$ is *yes*, then in the next step choose $r = \lceil p(n)/4 \rceil$, if *no*, then choose $r = \lceil 3p(n)/4 \rceil$ and continue the same procedure on the following answers. With this *binary search* the oracle Turing machine with oracle in $\mathbf{NP}$ is able to determine a after x oracle-calls with $2^x = p(n)$, that is $x = \lceil \log_2 p(n) \rceil$ oracle calls, which is of $O(\log n)$; therefore this computation problem requires only a logarithmic number of calls to an $\mathbf{NP}$-oracle. The proof is analogous for $a \geq r$. □

For instance a could be the number of rules of a monotonic rulesystem (with n rules) which cannot be applied; or another example is that a is the number of SAT-instances which can be satisfied out of a number of n such instances (SAT_n^{odd}). In both cases, a is trivially polynomially bounded, since $a \leq n$. Of course, if $a(I_1, \ldots, I_n) \geq r$ is in $\mathbf{NP}$ the proposition holds, too.

Proposition 5.2.5 *$\mathcal{M}$ is a model of a default theory $\langle D, W \rangle$ iff there exists an extension of $\langle D, W \rangle$ generated by a subset $G_1 \subseteq G$, where $G = \{d \mid \mathcal{M} \models c(d)\}$.*

Proof. By definition, $\mathcal{M}$ is a model of $\langle D, W \rangle$ if and only if $\mathcal{M}$ is a model of at least one extension of $\langle D, W \rangle$. The extension(s) cannot contain any consequent of the set $D \backslash G$ (otherwise $\mathcal{M}$ would not be a model of the extension), therefore G is a superset of the set of generating defaults. It is a superset because there may be defaults which do not fire because their prerequisites are not applicable. Therefore at least one set $G_1 \subseteq G$ must be a set of generating defaults. □

Proposition 5.2.6 *In a normal default theory there is at most one extension E (generated by $G_1 \subseteq G$; G defined as above) w.r.t. to a given model $\mathcal{M}$, i.e. at most one extension E with $\mathcal{M} \models E$.*

Proof. Assume there is more than one set $G_1 \subseteq G$ which generates an extension. Since G consists only of compatible defaults, due to Lemma 5.2.2 there is exactly one extension $W \wedge c(G_1)$ and therefore one or no extension w.r.t. to a given model $\mathcal{M}$ (this proposition also follows immediately

[1] In the main proof of the next section these computations are only needed as intermediate steps in a decision problem, hence there is no need to deal with complexity classes of functions.

of Theorem 4.8 in [Marek and Truszczyński, 1993] that the union of two different extensions of a normal default theory is always inconsistent – hence $\mathcal{M}$ satisfies at most one extension). □

However, it is possible that more than one "extension" in the sense $F = cons(W \cup \{c(d) \mid F \models p(d), F \not\models \neg\beta_i \, \forall \beta_i \in j(d)\})$ exists – this is the definition of *weak extensions* (see also Sections 5.5 and 2.5.4); for instance

$$D = \{a : b/b,\ b : a/a,\ : c/c\}$$

has two weak extensions, namely $cons\{a, b, c\}$ and $cons\{c\}$; however only $cons\{c\}$ forms an extension.

Proposition 5.2.7 *Let E be an extension of the default theory $\langle G, W\rangle$ and $G \subseteq D$; then E is an extension of $\langle D, W\rangle$ iff for each default d in $D\backslash G$, $W \wedge c(GD(G, E)) \not\models p(d)$ or $W \wedge c(GD(G, E)) \models \neg\beta_i$ for at least one i with $\beta_i \in j(d)$.*

Proof. An extension E of $\langle G, W\rangle$ with $G \subseteq D$ is an extension of $\langle D, W\rangle$ iff no default of $D\backslash G$ is in the generating set of E w.r.t. $\langle D, W\rangle$, i.e. of each default in this set either the prerequisite cannot be inferred or one of the justifications is not consistent with the extension. □

5.3 Model checking for Normal Default Theories is $\Delta_2^P[O(\log n)]$-complete

Theorem 5.3.1 *Let $\mathcal{M}$ be an interpretation and $\langle D, W\rangle$ a (not necessarily prerequisite-free) normal default theory. Deciding whether $\mathcal{M} \models \langle D, W\rangle$ is in $\Delta_2^P[O(\log n)]$.*

Proof. We describe a Turing machine M which decides this problem in polynomial time using an $\mathbf{NP}$-oracle for only $O(\log n)$ times. M works in four steps. A normal default theory $\langle D, W\rangle$ with n defaults is of the form $D = \{\alpha_1 : \beta_1/\beta_1, \ldots, \alpha_n : \beta_n/\beta_n\}$.

<u>*Step 1:*</u> M checks whether $\mathcal{M} \models W$. Since each extension contains W, $\mathcal{M}$ can only be a model of $\langle D, W\rangle$ if $\mathcal{M}$ satisfies W.

Furthermore M determines the set $G \subseteq D$ be the set of all defaults d with $\mathcal{M} \models c(d)$. This step eliminates all undesired ("bad") defaults $B = D\backslash G$. None of the defaults of B shall fire in this construction of an extension, or $\mathcal{M}$ is not a model of this extension (Proposition 5.2.5).

The check whether $\mathcal{M} \models W$ and the construction of G can be achieved in polynomial time, because model checking for propositional formulas is tractable. In the following, let m be the cardinality of $|G|$.

<u>*Step 2:*</u> M now computes the cardinality of the set G_1 of generating defaults of the extension, i.e. of defaults in G which are applicable because their prerequisites can be inferred. Because of Lemma 5.2.2 the set $cons(W \cup c(G_1))$ is unique and hence also G_1 is unique (in G there are no

defaults that block each other – due to the construction of G all consequents, and therefore all justifications, are jointly consistent, see also Proposition 5.2.6.).

The machine M assumes that $E = cons(W \cup c(G_1))$ is an extension and in step 3 and 4 M will verify that none of the "bad" defaults fires (using the results of Propositions 5.2.6 and 5.2.7), because then this set of generating defaults really forms an extension.

To compute the cardinality of G_1, M determines the number $|G_2| = |G \backslash G_1|$ of defaults that do not fire. It is easy to see that we can identify $\langle G, W\rangle$ with a monotonic rulesystem $\langle R, W\rangle$ in which all consequents are jointly consistent and use the results of the lemmata of the previous chapter:

Claim The problem to decide if at least r rules of a monotonic rulesystem (in n rules) do not fire is in $\mathbf{NP}$ (i.e. $|G_2| \geq r$).

Proof of Claim: The machine guesses a data-structure $\langle G_2', \{\mathcal{I}_1, \dots, \mathcal{I}_r\}\rangle$ with $|G_2'| = r$. G_2' is a set of defaults (rules), the $\mathcal{I}_j (1 \leq j \leq r)$ are interpretations. Now M proves in polynomial time (using the monotonic rule system syntax of Lemma 5.2.3):

$$\forall j: \quad \mathcal{I}_j \models W$$

$$\forall j \quad \forall b \in R \backslash G_2': \quad \mathcal{I}_j \models b$$

$$\forall a \in G_2' \quad \exists j: \quad \mathcal{I}_j \not\models a$$

M initially called an oracle in $\mathbf{NP}$ and then polynomially tests the guessed G_2' and the r guessed interpretations. Due to Lemma 5.2.3 this is equivalent to the question if at least r rules do not fire (An intuitive explanation why it is possible to guess a data-structure is, that the question is of the form of $\exists G_2' \dots f \not\models g$?; the fact that "$\not\models$" is required to guess a structure instead of more than one guessing stage is important as inference is $\mathbf{coNP}$-complete, hence the converse problem is $\mathbf{NP}$-complete, and as shown this question is equivalent to the question $\exists G_2' \exists \mathcal{I}_1 \dots \exists \mathcal{I}_r \dots$? – preceeding a polynomial procedure only existential quantifiers occur, which can be guessed in a single step; rf. to Section 3.11 for a discussion of logical characterisations of complexity classes). $\diamond$

The exact computation of the number t of rules which do not fire can, due to Proposition 5.2.4, be done in polynomial time (see also the discussion following the referred proposition) with a logarithmic number $(\log m)$ of calls to an $\mathbf{NP}$-oracle. After concluding this step, M knows $|G_2| = t$, hence $|G_1| = m - t$.

Step 3: M now has to prove that none of the "bad" defaults is applicable, because then the assumption that G_1 is a set of generating defaults is valid.

If it can be shown that $W \cup c(G_1) \models p(d)$ and $W \wedge c(G_1) \wedge j(d)$ is consistent for at least one $d \in B = D \backslash G$, then G_1 is not a set of generating defaults and the given interpretation $\mathcal{M}$ is not a model for $\langle D, W\rangle$. So it must be shown that for all "bad" defaults at least one of these two conditions is not fulfilled. Then, $cons(W \cup c(G_1))$ is an extension E (Proposition 5.2.7) and $\mathcal{M} \models E$, so $\mathcal{M}$ is a model of $\langle D, W\rangle$.

Two types of "bad" defaults need to be distinguished. There is one set B_1 consisting of defaults in which the prerequisites are not applicable, formally

$$B_1 = \{d_i \in B \mid W \cup c(G_1) \not\models p(d_i)\}.$$

$B_2 = B \backslash B_1$ is the set of defaults of B in which the prerequisites are applicable. The goal is now to determine the exact number u of defaults in B_1. Then M is able (in the fourth step) to test for each default in B_2 if the justifications are consistent with $W \wedge c(G_1)$ after guessing the right G_1 and B_1.

$$\overbrace{\overbrace{\underbrace{G_1}_{|G_1|=m-t}\quad \underbrace{G_2}_{|G_2|=t}}^{|G|=m}\quad \overbrace{\underbrace{B_1}_{|B_1|=u}\quad \underbrace{B_2}_{|B_2|=n-m-u}}^{|B|=n-m}}^{|D|=n}$$

Claim The problem to decide, given the cardinality of G_1, if the number of defaults in B_1 is at least s is in $\mathbf{NP}$.

Proof of Claim: For a given s, M guesses a data-structure

$$\langle G_1 \subseteq G, \{\mathcal{N}_1, \dots, \mathcal{N}_t\}, B_1' \subseteq B, \{\mathcal{O}_1, \dots, \mathcal{O}_s\}\rangle$$

with $|G_1| = m - t$, $|B_1'| = s$, and where $\mathcal{N}_e$ and $\mathcal{O}_y$ are interpretations. Now, M proves in polynomial time:

- $\forall d \in G_2 = G \backslash G_1$ there exists an interpretation $\mathcal{N}_e\,(1 \leq e \leq t)$ with $\mathcal{N}_e \not\models p(d)$; and $\forall e : \mathcal{N}_e \models W \wedge c(G_1)$ (In Step 2, M was provided with the information about the cardinality of G_1, therefore M has now just to guess a G_1 of suitable cardinality and test if it is the right one.).
- $\forall d \in B_1'$ there exists an interpretation $\mathcal{O}_y(1 \leq y \leq s)$ such that $\mathcal{O}_y \not\models p(d)$; and $\forall y : \mathcal{O}_y \models W \wedge c(G_1)$. If it is possible to choose the $\mathcal{O}_y$ this way then there are at least s not applicable prerequisites of defaults, hence $|B_1| \geq s$. This is not an application of Lemma 5.2.3 because there is a fixed set $c(G_1)$ – there is no need to know the consequents of B_1' (This statement is equivalent to $\forall d \in B_1' : W \cup c(G_1) \not\models p(d)$.).

Both steps can be performed in polynomial time and are due to the initial guess in $\mathbf{NP}$. $\diamond$

Due to Proposition 5.2.4, $u = |B_1|$ can be computed with $\lceil \log(n - m) \rceil$ oracle-calls.

<u>Step 4:</u> In the previous steps M has determined the cardinality of B_1 and of G_1. Now the machine finally has to check that $W \wedge c(G_1) \wedge j(d)$ is inconsistent for all defaults $d \in B_2$ (if *yes*, then G_1 is a generating set).

To this end, let us show that the converse problem is in $\mathbf{NP}$ and introduce a machine N solving the converse problem.

Claim The problem to decide if $W \wedge c(G) \wedge j(d)$ is consistent for at least one $d \in B_2$, given the cardinality of G_1 and B_1, is in $\mathbf{NP}$.

Proof of Claim: Machine N guesses a data-structure

$$\langle G_1 \subseteq G, \{\mathcal{N}_1, \dots, \mathcal{N}_t\}, B_1 \subseteq B, \{\mathcal{P}_1, \dots, \mathcal{P}_u\}, \mathcal{Q}\rangle$$

with $|G_1| = m - t$ and $|B_1| = u$. Now N tests in polynomial time:

- $\forall d \in G_2 = G \backslash G_1$ there exists interpretations $\mathcal{N}_e$ $(1 \leq e \leq t)$ with $\mathcal{N}_e \not\models p(d)$; and $\forall e : \mathcal{N}_e \models W \wedge c(G_1)$
- $\forall d \in B_1$ there exists interpretations $\mathcal{P}$ $(1 \leq z \leq u)$ such that $\mathcal{P}_z \not\models p(d)$; and $\forall z : \mathcal{P}_z \models W \wedge c(G_1)$. N knows the cardinality of B_1 (step 3), therefore N just needs to show that it has guessed the right B_1.
- Now N just needs to prove that $\exists d \in B_2 : \mathcal{Q} \models W \wedge c(G_1)$ and $\mathcal{Q} \models c(d)$ (with $c(d) = j(d)$). This states the fact that there is a default which is consistent with $cons(W \cup c(G_1))$.

All steps can be concluded in polynomial time and are, due to the initial guess, in $\mathbf{NP}$. (Intuitively, the problem of machine M is of the form: $\exists G_1 \exists B_1 \forall Q$ $(a \wedge b \Rightarrow c)$. Thus, the converse problem is $\forall G_1 \forall B_1 \exists Q$ $(a \wedge b \wedge \neg c)$. However, as the cardinality of G_1 and B_1 is known from the previous steps it does not matter if the question includes $\forall G_1$ which fulfill a special property a or $\exists G_1$ with a special property a, as there is exactly one G_1 of this cardinality, hence a data-structure can be guessed.) $\diamond$

This means, that the output *yes* of the oracle Turing-machine N is achieved iff there exist defaults of B_2 which fire (with guessing the right G_1, the right B_1 and a suitable $\mathcal{Q}$). We define that the Turing-machine M rejects on output *yes* and accepts on output *no* (getting the answer *no* means that no default of B_2 is consistent with the extension). This converse problem therefore needs one (negative) oracle-call.

Overall, we showed that Step 1 can be accomplished in polynomial time, Step 2 in polynomial time with $\lceil \log m \rceil$ oracle-calls. Step 3 was concluded after at most $\lceil \log(n-m) \rceil$ oracle-calls and Step 4 just needed one oracle call. Therefore this problem is solvable in polynomial time with $\lceil \log(m(n-1)) \rceil + 1 = O(\log n)$ calls to three different oracles in $\mathbf{NP}$.

For the sake of a simpler presentation we have described three different $\mathbf{NP}$-oracles (the three oracles of step 2,3,4, respectively). With standard techniques, these three oracles of course can be replaced with a single oracle. Hence this problem is in the class $\Delta_2^P[O(\log n)]$. $\square$

In Figure 1 the algorithm of the membership proof of the above theorem is represented in a compact way (however, for ease of readability, not every detail is described, for instance quantifiers are used instead of "**for** ... **do**" declarations (like in Figure 2) and auxiliary procedures like "**function** $\models$(N: interpretation,F: formula): boolean;" are not described).

The completeness of model checking for normal default theories follows directly from the results of Liberatore and Schaerf [1998], however, we will give a different proof in the following corollary, using a well-known problem for reduction as introduced in Section 3.9:

Proposition 5.3.2 *[Wagner, 1990] The problem to decide, given the quantified boolean formulae $F_1, \ldots F_n$, such that $F_i \notin QBF_{k,\exists}$ implies $F_{i+1} \notin QBF_{k,\exists}$ for $1 \leq i \leq n$, whether $max\{i : 1 \leq i \leq n, F_i \in QBF_{k,\exists}\}$ is odd is $\Delta_{k+1}^P[O(\log n)]$-complete.*

As introduced in Section 3.9, *Quantified Boolean Formula* $QBF_{k,\exists}$ is the following problem:

$$\exists P \forall Q \exists R \ldots F(P, Q, R, \ldots)?$$

Input: An interpretation $\mathcal{M}$, a normal default theory $\langle D, W\rangle$.
Result: "Accept" if $\mathcal{M} \models \langle D, W\rangle$; "Reject" otherwise.

begin
if $\mathcal{M} \not\models W$ **then reject and halt**;
$G := \{d \mid \mathcal{M} \models c(d)\}$; $m := |G|$
$x := 1$; $r :=$ int$(m/2 + 1)$;
while $2^{x-1} \leq m$ **do**
    begin
    $x := x + 1$; $|G_2'| := r$;
    guess $\langle G_2', \{\mathcal{I}_1, \ldots, \mathcal{I}_r\}\rangle$;
    if $\forall j: \mathcal{I}_j \models W,\ \forall d \in D \backslash G_2'\ \forall j: \mathcal{I}_j \models c(d),\ \forall d \in G_2'\ \exists j: \mathcal{I}_j \not\models p(d)$
        then $r := r +$ int$(m/(2^x) + 1)$
        else $r := r -$ int$(m/(2^x) + 1)$;
    end;
if $\forall j: \mathcal{I}_j \models W,\ \forall d \in D \backslash G_2'\ \forall j: \mathcal{I}_j \models c(d),\ \forall d \in G_2'\ \exists j: \mathcal{I}_j \not\models p(d)$
    then $|G_1| := m - r$ **else** $|G_1| := m - (r - 1)$;
$x := 1$; $s :=$ int$((m - n)/2 + 1)$;
while $2^{x-1} \leq n - m$ **do**
    begin
    $x := x + 1$; $|B_1'| := s$;
    guess $\langle G_1 \subseteq G, \{\mathcal{I}_1, \ldots, \mathcal{I}_{|G_2|}\}, B_1' \subseteq B, \{\mathcal{O}_1, \ldots, \mathcal{O}_s\}\rangle$;
    if $\forall j: \mathcal{I}_j \models W,\ \forall d \in G_1\ \forall j: \mathcal{I}_j \models c(d),\ \forall d \in G_2\ \exists j: \mathcal{I}_j \not\models p(d)$
        and $\forall y: \mathcal{O}_y \models W \wedge c(G_1),\ \forall d \in B_1'\ \exists y: \mathcal{O}_y \not\models p(d)$
            then $s := s +$ int$((n - m)/(2^x) + 1)$
            else $s := s -$ int$((n - m)/(2^x) + 1)$;
    end;
if $\forall j\ \mathcal{I}_j \models W,\ \forall d \in G_1\ \forall j: \mathcal{I}_j \models c(d),\ \forall d \in G_2\ \exists j: \mathcal{I}_j \not\models p(d)$
    and $\forall y: \mathcal{O}_y \models W \wedge c(G_1),\ \forall d \in B_1'\ \exists y: \mathcal{O}_y \not\models p(d)$
        then $|B_1| := s$ **else** $|B_1| := s - 1$;
inverseguess $\langle G_1 \subseteq G, \{\mathcal{I}_1, \ldots, \mathcal{I}_{|G_2|}\}, B_1 \subseteq B, \{\mathcal{O}_1, \ldots, \mathcal{O}_{|B_1|}\}, \mathcal{Q}\rangle$;
if $\forall j: \mathcal{I}_j \models W,\ \forall d \in G_1\ \forall j: \mathcal{I}_j \models c(d),\ \forall d \in G_2\ \exists j: \mathcal{I}_j \not\models p(d)$
    and $\forall y: \mathcal{O}_y \models W \wedge c(G_1),\ \forall d \in B_1\ \exists y: \mathcal{O}_y \not\models p(d)$
    and $\exists d \in B_2: \mathcal{Q} \models W \wedge c(G_1) \wedge c(d)$
        then reject else accept;
end.

Figure 5.1: *Deterministic Algorithm of Theorem 5.3.1 using $O(\log n)$ Oracle-Calls*

with k quantifier alternations starting with $\exists$, using second-order quantification: Is this formula valid, i.e. does there exist an evaluation of the variables in P that for each evaluation of the variables of Q there is an evaluation of the variables in $R\ldots$ that $F(P,Q,R,\ldots)$ is true?

Corollary 5.3.3 *Let $\mathcal{M}$ be an interpretation and $\langle D,W\rangle$ a not necessarily prerequisite-free normal default theory $\langle D,W\rangle$. Deciding whether $\mathcal{M}\models\langle D,W\rangle$ is $\mathbf{P}^{\mathbf{NP}[O(\log n)]}$-complete.*

Proof. To show hardness we use a reduction of the problem mentioned in Proposition 5.3.2 for $\Delta_2^P[O(\log n)]$: This is the problem to decide, that given n formulae F_i, such that F_i is not satisfiable implies that F_{i+1} is not satisfiable, if the number of satisfiable formulae is odd.

Let z_j ($1\le j\le m$) be the propositional atoms which appear in the formulae $F_1,\ldots,F_n$ (and, although not necessary here, it can be assumed that the atoms of each two formulae are disjunct). We construct, if n is odd, the following default theory

$$W=\bigcup_i\{(\neg F_i\vee\neg x_{i-1})\Rightarrow\neg x_i\}$$

$$D=\left\{\frac{\neg x_1\wedge\ldots\wedge\neg x_n : x_0}{x_0},\frac{\neg x_3\wedge\ldots\wedge\neg x_n : x_0\wedge x_1\wedge x_2}{x_0\wedge x_1\wedge x_2},\right.$$
$$\left.\frac{\neg x_5\wedge\ldots\wedge\neg x_n : x_0\wedge x_1\wedge x_2\wedge x_3\wedge x_4}{x_0\wedge x_1\wedge x_2\wedge x_3\wedge x_4},\ldots,\frac{\neg x_n : x_0\wedge x_1\wedge\ldots\wedge x_{n-1}}{x_0\wedge x_1\wedge\ldots\wedge x_{n-1}}\right\}$$

If the number n is even, the last default has to be replaced with

$$\frac{: x_0\wedge x_1\wedge\ldots\wedge x_n}{x_0\wedge x_1\ldots\wedge x_n}$$

In this default theory the *new* propositional atoms $x_0,x_1,\ldots,x_n$ and (in the F_i only) the old atoms z_j ($1\le j\le m$) are occurring.

The transformation is clearly polynomial. Hence it remains to be shown that the empty set is a model of $\langle D,W\rangle$ iff the number of satisfiable formulae is odd.

- Assume the number is odd. If F_1 is satisfiable, then $\neg F_1$ is not a tautology, so the first implication in W does not allow to infer x_1. We will reach the point where F_i, i even, is not satisfiable (or all F_i, only if n is odd, are satisfiable); then $\neg F_i$ is a tautology, i.e. this implication yields $\neg x_i$. The same holds for each implication where $\neg x_{i+l}$, $1\le l\le n-i$, is inferred due to the construction of W. Now, none of the defaults can be used – the smallest inferred $\neg x_i$ is even, and we cannot say anything about the truth value of the smaller $\neg x_i$ – of course, some prerequisites can be inferred, however, then the justifications are not consistent with the context (which is W here, the only context which forms an extension in this case). Therefore, $cons(W)$ is the only extension. The empty set satisfies W (this is the reason why $\neg x_{i-1}$ occurs in each implication, allowing the z_j to have any value; $\neg x_0$ on the left-hand side in each implication would have been sufficient, too, because $taut(\neg F_i)$ implies $taut(\neg F_{i+1})$ anyway), hence $\mathcal{M}=\emptyset$ is a model. The same holds in the case where all F_i (with odd n) are satisfiable, since in this case no default is applicable either.

$\frac{\neg a \wedge b : c}{c}$ $\frac{:\neg a}{\neg a}$ $\frac{: a \vee b}{a \vee b}$	$\frac{d : b}{b}$	$\frac{:\neg b}{\neg b}$ $\frac{c : \neg d}{\neg d}$	$\frac{\neg b : a \wedge c}{a \wedge c}$
G_1	G_2	B_1	B_2

Figure 5.2: *Good and Bad Defaults of Example 5.3.6*

- $\mathcal{M} = \emptyset$ is a model implies that no positive x_i occurs in at least one extension. This implies that none of the default rules should be used, which implies that the smallest inferred $\neg x_i$ is even, i.e. the largest not inferred $\neg x_i$ is odd; this implies that the first p implications are not applied, with p is an odd number; hence it follows that the first p $\neg F_i$ are no tautologies, i.e. the first p F_i are satisfiable. Therefore an odd number of F_i is satisfiable.

Therefore, $\emptyset \models \langle D, W \rangle$ iff the number of satisfiable formulae is odd. □

Corollary 5.3.4 *Let $\mathcal{M}$ be an interpretation and $\langle D, W \rangle$ a prerequisite-free normal default theory $\langle D, W \rangle$. Deciding whether $\mathcal{M} \models \langle D, W \rangle$ is in* $\mathbf{coNP}$.

Proof. This fact, which is also proved in [Liberatore and Schaerf, 1998] immediately follows from the proof of Theorem 5.3.1, because steps 2 and 3 are not needed and it is sufficient to guess a $\mathcal{Q}$ in step 4 for prerequisite-free default-theories which is clearly in $\mathbf{coNP}$ (Figure 5.3). □

Corollary 5.3.5 *Let $\mathcal{M}$ be an interpretation and $\langle D, W \rangle$ a prerequisite-free normal default theory $\langle D, W \rangle$. Deciding whether $\mathcal{M} \models \langle D, W \rangle$ is* $\mathbf{coNP}$*-hard.*

Proof. This fact, which is also proved in [Liberatore and Schaerf, 1998] via a reduction from the model checking problem can be proved in a more intuitive and easier way. Just consider the default theory with a single default $D = \left\{ \frac{: w \wedge \neg F}{w \wedge \neg F} \right\}$ where w is a new propositional atom not occurring in F and F is a formula in conjunctive normal form. Now we show that F is a tautology exactly if the empty set is a model: The transformation is clearly polynomial. If F is a tautology, then this is the only chance that the default does not fire, and that the empty set is a model (since all variables in F are allowed to occur negated, since F is a tautology anyway). Otherwise, w is in the extension and $\mathcal{M}$ is hence not a model. □

Example 5.3.6 *Consider $\mathcal{M} = \{b, c, d\}$. The defaults of the default theory*

$$\left\{ \frac{\neg a \wedge b : c}{c}, \frac{: \neg a}{\neg a}, \frac{: a \vee b}{a \vee b}, \frac{: \neg b}{\neg b}, \frac{c : \neg d}{\neg d}, \frac{d : b}{b}, \frac{\neg b : a \wedge c}{a \wedge c} \right\}$$

Input: An interpretation $\mathcal{M}$, a normal prer.-free default theory $\langle D, W\rangle$.
Result: "Accept" if $\mathcal{M} \models \langle D, W\rangle$; "Reject" otherwise.

begin
inverseguess $\mathcal{Q}$;
if $\mathcal{M} \not\models W$ **then reject and halt**;
$G := \{d \mid \mathcal{M} \models c(d)\}$;
for each $d \in D \backslash G$ **do**
  begin
  if $\mathcal{Q} \models W \wedge c(G) \wedge c(d)$ **then reject and halt**;
  end;
accept;
end.

Figure 5.3: *Converse Non-Deterministic Algorithm of Corollary 5.3.4*

are split into $G = \left\{\frac{\neg a \wedge b : c}{c}, \frac{:\neg a}{\neg a}, \frac{:a \vee b}{a \vee b}, \frac{d:b}{b}\right\}$ *and* $B = \left\{\frac{:\neg b}{\neg b}, \frac{c:\neg d}{\neg d}, \frac{\neg b : a \wedge c}{a \wedge c}\right\}$.
Then the cardinality of G_1 and B_1 is determined; and finally checked if no justification of B_1 is applicable. Figure 5.2 shows the categories in which each defaults falls. As illustrated, one default of B_1 fires due to the applicable prerequisite.

5.4 Complexity of Model Checking for Disjunction-free and EHNF Default Theories

We introduce a class of default theories in "extended Horn normal form" (EHNF default theories) with a model checking problem which is one level easier than in general; to the best of our knowledge, no one so far has studied this class, probably because reasoning is still $\mathbf{NP}$-complete (at least, if the given formula is disjunction-free, too), even in the case of prerequisite-free normal default theories.

As special case disjunction-free default theories are analysed, where the computational cost of model checking resides in the same class as for EHNF default theories.

A default theory $\langle D, W\rangle$ is *disjunction-free*, if W, all $p(d)$, all elements of each $j(d)$, and all consequents $c(d)$ are conjunctions of (negated and not negated) literals (see Section 2.5.3). Concerning complexity results of reasoning in such limited settings, cf. Section 2.5.3.

We prove that the complexity of *model checking* is polynomial in the case of normal disjunction-free default theories and $\mathbf{NP}$-complete in the case of general disjunction-free default theories, even if limited to prerequisite-free ones. Moreover, as mentioned above, we introduce the ample class of *default theories in extended Horn normal form* and show that the the problem still remains in the corresponding classes.

Due to Section 2.3, we already know that a *Horn clause* is a disjunction of literals with at most one positive literal. A *Horn theory* is a conjunction of Horn clauses. A *dual Horn clause* is a conjunction of literals containing at most one negative literal. A *dual Horn theory* is a disjunction of dual Horn clauses.

Definition 5.4.1 *A default theory $\langle D, W\rangle$ is in extended Horn normal form (EHNF) iff W and all elements of each $j(d)$ are disjunction of Horn theories, each $p(d)$ is a conjunction of dual Horn theories, and each consequent c(d) is a Horn theory. If $\langle D, W\rangle$ is normal, then trivially, the only element of each $j(d)$ is only a Horn theory, too.*

The class of EHNF default theories is a superset of the class of disjunction-free default theories: Simply limit W and each β_i to one Horn theory with only one-element Horn clauses, limit each $c(d)$ to a Horn theory with unary clauses, too; and let each $p(d)$ be a conjunction of one-element dual Horn theories.

Example 5.4.2 *Let $\mathcal{L}$ be a propositional language consisting of the atoms $\{p, q, r, s, t\}$. Then, the default theory*

$$W = \{p \wedge \neg r\}$$

and

$$D = \left\{ \frac{p : \neg r, \neg s \wedge q}{q}, \frac{\neg t : \neg s}{\neg t \wedge p} \right\}$$

is a disjunction-free default theory. The default theory $\langle D, W\rangle$ with

$$W = \{((p \vee \neg r) \wedge (\neg s \vee \neg p)) \vee t\},$$

$$D = \left\{ \frac{((p \wedge r) \vee q) \wedge s \wedge t) : p \vee q \vee \neg s}{(p \vee \neg r) \wedge q} \right\}$$

is an EHNF default theory. EHNF default theories already admit very complicated defaults - like

$$\frac{((s \wedge p \wedge \neg q) \vee (q \wedge r) \vee \neg t)) \wedge ((v \wedge w) \vee (p \wedge \neg q)) : p \vee r \vee s, p \vee (q \wedge s), v \wedge w}{(p \wedge \neg r) \wedge (q \vee \neg v \vee \neg w) \wedge w}.$$

(E.g., looking at each justification reveals disjunctions of Horn theories).

5.4.1 General Case

Lemma 5.4.3 *Let $\mathcal{M}$ be an interpretation and $\langle D, W\rangle$ an EHNF default theory. Deciding whether $\mathcal{M} \models \langle D, W\rangle$ is in* $\mathbf{NP}$.

Proof. A non-deterministic polynomial time Turing-Machine M solving this problem is described as follows (see also Figure 3). After this description, we will argue why all satisfiability checks and inference checks can be done in polynomial time. M starts with guessing a set $G \subseteq D$. Let E be $cons(W \wedge c(G))$. Now M computes

$$E_{i+1} = E_i \cup \{c(d) \mid d \in D,\, E_i \models p(d), \forall \beta_i \in j(d) \;\; W \wedge c(G) \not\models \neg\beta_i\}$$

with $E_0 = W$ up to E_n where n is the number of defaults in D. Remember that all E_i are finite sets of formulae (Definition 2.5.9 and the following discussion). To compute $E_{|D|}$, only a polynomial number of inference checks and satisfiability checks is required. Additionally, M needs to check if $cons(E)$ is an extension. Hence, M shows that $(W \cup c(G)) \equiv (E_n)$, this is the case exactly if $G_n = G$ with $E_n = W \cup c(G_n)$ (and stores the set of used defaults in each step of the computation).

Finally, M just needs to show that $\mathcal{M}$ is a model of this extension, hence M needs to show that $\mathcal{M} \models W \wedge c(G)$ - since propositional model checking is tractable, this can be accomplished in polynomial time.

To show that all satisfiability and inference problems can be decided in polynomial time observe that deciding satisfiability of a Horn theory and tautology checking of a dual Horn theory (which is equivalent to the question if a Horn theory is not satisfiable) is polynomial. Furthermore, the question $sat(f_1 \vee \ldots \vee f_m)$? is equivalent to $sat(f_1)$ or $\ldots$ or $sat(f_m)$ and $taut(f_1 \wedge \ldots \wedge f_m)$? is equivalent to $taut(f_1)$ and $\ldots$ and $taut(f_m)$.

The satisfiability checks of this problem are of the form $W \wedge c(G) \not\models \neg\beta_i$, which is equivalent to $sat(W \wedge c(G) \wedge \beta_i)$. Let m be the maximum number of disjunctions of Horn theories in W and of all β_i of each $j(d)$. Then the disjunctions can be split with $O(m^2)$ steps to at most m^2 disjunctions of Horn theories of the form $W_k \wedge c(G) \wedge \beta_{il}$ where $W = W_1 \vee \ldots \vee W_k \vee \ldots \vee W_{\leq m}$ and $\beta_i = \beta_{i1} \wedge \ldots \wedge \ldots \wedge \beta_{il} \wedge \ldots \wedge \beta_{i,\leq m}$. Due to the above remarks the satisfiability checks can be split to at most m^2 checks $sat(W_k \wedge c(G) \wedge \beta_{il})$ where at least one check must be successful to achieve that the overall checking procedure yields a positive answer; for each $j(d)$ consisting of a maximum number of p justifications, $O(pm^2)$ checks are needed. Each check is polynomial since each formula is a Horn theory and satisfiability checking for Horn theories is polynomial.

One could even think about allowing some of the $c(d)$ to be *a disjunction* of Horn theories: However, if all $c(d)$ are allowed to be disjunctions of Horn theories, the problem would require $O(m^n)$ splits and turn into an intractable problem. Even if only a logarithmic number $log(n)$ of consequents would be allowed to be a disjunction of Horn theories, the problem would (probably) have no polynomial algorithm, as $O(m^{\log n})$ is still of exponential order. Hence, only a fixed number of consequents could be allowed to be disjunctions of Horn theories.

The inference checks $E_i \models p(d)$ (each E_i is of the form $W \wedge c(G_i)$ with $G_i \subseteq D$) are due to the deduction theorem equivalent to $\models \neg E_i \vee p(d)$. $\neg(W_k \wedge c(G_i))$ is a dual Horn theory since $W_k \wedge c(G_i)$ is a set of Horn formulae. Similar as above, at most $O(m^2)$ conjunctions are needed to split the formula $\neg E_i \vee p(d)$ up to conjunctions of $\neg W_k \vee \neg c(G_i) \vee p_l(d)$. Since tautology checking of conjunctions can be split as remarked above, and the formula is a tautology if all of its conjunctions are tautologies, we face $O(m^2)$ tautology tests where each check can be done in polynomial time since only dual Horn theories occur in each test. □

Input: An interpretation $\mathcal{M}$, a disjunction-free default theory $\langle D, W\rangle$.
Result: "Accept" if $\mathcal{M} \models \langle D, W\rangle$; "Reject" otherwise.

begin
guess $G \subseteq D$;
$i := 0$; $E(0) := W$; $G(0) := \emptyset$; $disw :=$ no. of disj. of Horntheories in W;
while $i \leq n$ **do**
  begin
  $E(i+1) := E(i)$; $G(i+1) := G(i)$;   % $E(i)$ is not needed
  for each $d \in D$ **do**
    begin
    $disp :=$ no. of conj. of dual Horntheories in $p(d)$; $x := 0$;
    for $k := 1$ **to** $disw$
      begin
      for $l := 1$ **to** $disp$
        begin
        if $taut(\neg W_k \vee \neg c(G_i) \vee p_l(d))$ **then** $x := x + 1$;
        end;
      end;
    if $x = disw \cdot disp$ **then**   % i.e. all subformulae are tautologies
      begin
      $x := 0$; $y := 0$; $nob :=$ no. of $\beta_i \in j(d)$
      for each $\beta_i \in j(d)$ **do**
        begin
        $disb :=$ no. of disj. of Horntheories in β_i;
        for $k := 1$ **to** $disw$
          begin
          for $l := 1$ **to** $disb$
            begin
            if $sat(W_k \wedge c(G) \wedge \beta_{il}(d))$ **then** $x := 1$;
            end;
          end;
        end;
      $y := y + x$; $x := 0$;
      end;
      if $y = nob$ **and** $d \notin G(i+1)$ **then**   % $y = nob$ iff each $W \wedge c(G) \wedge \beta_i$ is satisf.
        $E(i+1) := E(i+1) \cup \{c(d)\}$ **and** $G(i+1) := G(i+1) \cup \{d\}$;
    end;
  $i := i + 1$;
  end;
if $G(n) \neq G$ **then reject and halt**;
if $\mathcal{M} \not\models W \wedge c(G)$ **then reject and halt**;
accept;
end.

Figure 5.4: *Non-Deterministic Algorithm of Lemma 5.4.3*

Corollary 5.4.4 *Let $\mathcal{M}$ be an interpretation and $\langle D, W\rangle$ a disjunction-free default theory. Deciding whether $\mathcal{M} \models \langle D, W\rangle$ is in* $\mathbf{NP}$.

Lemma 5.4.5 *Let $\mathcal{M}$ be an interpretation and $\langle D, W\rangle$ an EHNF default theory. Deciding whether $\mathcal{M} \models \langle D, W\rangle$ is $\mathbf{NP}$-hard, even if $\langle D, W\rangle$ is prerequisite-free, semi-normal, disjunction-free and all consequents are single literals.*

Proof. We use a reduction from 3SAT. Let $F(t_1, \ldots, t_m)$ be a formula in conjunctive normal form with exactly three literals per clause:

$$F = F_1 \wedge \ldots \wedge F_i \wedge \ldots \wedge F_n$$

with $F_i = x_{i1} \vee x_{i2} \vee x_{i3}$ (the x_{ij} are literals, and the t_k are propositional atoms). Deciding if F is satisfiable is $\mathbf{NP}$-complete.

We prove that F is satisfiable iff $\mathcal{M} = \emptyset$ is a model of the following default theory: $W = \emptyset$,

$$D = \bigcup_k \left\{ \frac{: u_k \wedge \neg t_k}{\neg t_k}, \frac{: t_k \wedge \neg u_k}{\neg u_k} \right\} \cup$$
$$\bigcup_i \left\{ \frac{: f(x_{i1}) \wedge \neg y_i}{\neg y_i}, \frac{: f(x_{i2}) \wedge \neg y_i}{\neg y_i}, \frac{: f(x_{i3}) \wedge \neg y_i}{\neg y_i}, \frac{: y_i}{y_i} \right\} \cup$$
$$\left\{ \frac{: \neg y_1 \wedge \ldots \wedge \neg y_n \wedge \neg z}{\neg z}, \frac{: z}{z} \right\}$$

with $f(x_{ij}) = \neg t_k \wedge u_k$ if $x_{ij} = \neg t_k$ and $f(x_{ij}) = \neg u_k \wedge t_k$ if $x_{ij} = t_k$.

This is a semi-normal prerequisite-free default theory[2] in the propositional atoms

$$\{t_1, \ldots t_m, u_1, \ldots, u_m, y_1, \ldots, y_n, z\}.$$

Intuitively the new variables u_k are introduced because it is required that the empty set (i.e. the model in which all atoms are evaluated as *false*) is a model regardless which evaluation of $(t_1, \ldots, t_m)$ satisfies F. This default theory is semi-normal (If we just used $: \neg u_k / \neg u_k$ and $: \neg t_k / \neg t_k$ instead, an extension could contain for any value of k both of $\neg u_k$ and $\neg t_k$; in the above theory one default blocks the other and takes care that for any k exactly one of $\neg t_k$ or $\neg u_k$ will be in the extension, hence representing an evaluation of F.). "Disjunctions" which are necessary to represent F are used in this theory via three defaults with the same consequence $\neg y_i$.

The transformation is clearly polynomial, so it only remains to show that $\mathcal{M} = \emptyset$ is a model iff F is satisfiable.

[2]The last two defaults could be dropped without modifying the proof very much, however they are kept for ease of argumentation.

- $\mathcal{M} = \emptyset$ is a model implies that $\mathcal{M}$ must be a model of an extension containing $\neg z$ because every possible extension has to be a set where a maximum number of possible defaults is used and hence must either contain z (from $: z/z$), or contain $\neg z$. It only contains $\neg z$ if $\neg y_1 \wedge \ldots \wedge \neg y_n$ is consistent with the extension. This can only then be accomplished if at least one of the three defaults with consequence $\neg y_i$ fires, because if not, then, due to $: y_i/y_i$ y_i has to be used (this default theory is total w.r.t. z and the y_i) and the conjunction of the $\neg y_i$ would not be consistent with the extension. The conjunction of all $\neg y_i$ will be consistent with the context, if there is a combination of $\neg t_k$ and $\neg u_k$ in the context (exactly one of them needs to be in the context to have a chance that this context forms an extension) that allows to infer all $\neg y_i$, i.e. that for each i at least one of the three defaults yielding $\neg y_i$ is applicable. If $\neg u_k$ is part of the combination, t_k is *true*, if not (then $\neg t_k$ is in the combination) t_k is *false*. This is a propositional model which satisfies F (each clause evaluates to *true* under this evaluation), therefore F is satisfiable.

- F is satisfiable implies that there is at least one truth evaluation $\mathcal{N}$ which satisfies F. We construct an extension containing for each k $\neg t_k$ if t_k is not in $\mathcal{N}$ and $\neg u_k$ if t_k is in $\mathcal{N}$. For a fixed model $\mathcal{N}$, $\mathcal{N} \models F$ is equivalent to $\mathcal{N} \models F_1$ and $\ldots$ and $\mathcal{N} \models F_n$. Therefore, in this extension each $\neg y_i$ can be derived because at least one of the justifications is consistent with the context and therefore due to the same reasons, also $\neg z$. The set which contains for each k either $\neg t_k$ or $\neg u_k$, and which contains all $\neg y_i$ and $\neg z$, is a maximal set, because no justification of another default is consistent with it, and is, since all mentioned literals are inferred with this context, too, an extension. This extension contains only negated literals, hence $\mathcal{M} = \emptyset$ is a model of the default-theory.

Therefore: $\mathcal{M} = \emptyset$ is a model iff F is satisfiable. $\square$

In the above proof instead of using a prerequisite-free default theory an "almost" normal default theory with prerequisites could be used:

$$D = \Big\{ \frac{: u_1 \wedge \neg t_1}{\neg t_1}, \ldots, \frac{: u_m \wedge \neg t_m}{\neg t_m}, \frac{: t_1 \wedge \neg u_1}{\neg u_1}, \ldots, \frac{: t_m \wedge \neg u_m}{\neg u_m},$$

$$\frac{f(x_{11}) : \neg y_1}{\neg y_1}, \frac{f(x_{12}) : \neg y_1}{\neg y_1}, \frac{f(x_{13}) : \neg y_1}{\neg y_1}, \ldots,$$

$$\frac{f(x_{n1}) : \neg y_n}{\neg y_n}, \frac{f(x_{n2}) : \neg y_n}{\neg y_n}, \frac{f(x_{n3}) : \neg y_n}{\neg y_n}, \frac{\neg y_1 \wedge \ldots \wedge \neg y_n : \neg z}{\neg z}, \frac{: z}{z} \Big\}$$

In this case $f(x_{ij}) = \neg t_k$ if $x_{ij} = \neg t_k$ and $f(x_{ij}) = \neg u_k$ if $x_{ij} = t_k$. Since the $f(x_{ij})$ are used as prerequisites here, they have to be inferred and need not only be consistent with the context – in the context we have to use for instance $\neg t_k \wedge u_k$ to avoid that an unwanted default is used if $\neg u_k$ has been inferred, in the case of prerequisites we do not want to infer positive literals.

If $W = \{\neg u_1 \oplus \neg t_1, \ldots, \neg u_k \oplus \neg t_k, \ldots, \neg u_m \oplus \neg t_m\}$ with $\oplus$ being the *exclusive or* operator, then one might think that the above default theory could be represented as a normal one replacing

$$\bigcup_k \left\{ \frac{: u_k \wedge \neg t_k}{\neg t_k}, \frac{: t_k \wedge \neg u_k}{\neg u_k} \right\}$$

with

$$\bigcup_k \left\{ \frac{: \neg t_k}{\neg t_k}, \frac{: \neg u_k}{\neg u_k} \right\}.$$

However, on the one hand, this W is not a Horn theory as required, since $\neg u_k \oplus \neg t_k = (\neg u_k \vee \neg t_k) \wedge (u_k \vee t_k)$ (the second clause is not a Horn clause). On the other hand, the more important problem is that this W definitely settles the value of t_k if we knew u_k (and vice versa) – this is completely unwanted; the special property is that there shall only be negated literals in an extension to get $\mathcal{M} = \emptyset$ as model – here we would force positive literals to be in the wanted extension, too (if $\neg u_k$ is part of the extension, then t_k is element of it, too). However, if we are allowed to use the *effective disjunction* operator of disjunctive default theories we could define W (or special defaults without prerequisites and justifications) as $\{\neg u_1 | \neg t_1, \ldots, \neg u_k | \neg t_k, \ldots, \neg u_m | \neg t_m\}$ and drop the first defaults (see [Gelfond *et al.*, 1991] for details on disjunctive default logic) to achieve a normal default theory.

With the previous results, we immediately obtain the following theorem:

Theorem 5.4.6 *Let $\mathcal{M}$ be an interpretation and $\langle D, W\rangle$ an EHNF default theory. Deciding if $\mathcal{M} \models \langle D, W\rangle$ is $\mathbf{NP}$-complete, even if $\langle D, W\rangle$ is prerequisite-free, semi-normal, disjunction-free and all consequents are single literals.*

5.4.2 Normal Default Theories

We show that in the case of normal EHNF default theories model checking is tractable and moreover, belongs to the hardest problems in $\mathbf{P}$ w.r.t. Logspace-reductions, i.e. is $\mathbf{P}$-complete. Compare this to the result of Section 5.3: Model checking with normal default theories is $\Delta_2^P[O(\log n)]$-complete, which is equal to the class $\mathbf{L}^{\mathbf{NP}}$ [Papadimitriou, 1993] (usually we abbreviate the class of logarithmic space computations with $\mathbf{L}$ whereas for reductions we use the term Logspace-reductions.). For normal EHNF default theories no oracle-calls are necessary. However, we show that these oracle calls turn into $\mathbf{P}$-hard problems, hence this problem is $\mathbf{P}$-hard (more precisely, if trying the same procedure as in the corresponding proof in Chapter 5.3, then the oracle calls would remain due to an exponential number of interpretations; and hence an alternative checking procedure has to be used).

Lemma 5.4.7 *Let $\mathcal{M}$ be an interpretation and $\langle D, W\rangle$ an EHNF normal default theory. Deciding whether $\mathcal{M} \models \langle D, W\rangle$ is in $\mathbf{P}$.*

Proof. Let the input length m of this problem be the number of defaults multiplied with the largest number of disjunctions of Horn theories in W or β_i, resp. conjunctions of dual Horn theories in $p(d)$ multiplied with the largest number of conjunctions of all Horn theories (resp. dual Horn theories) and multiplied with the largest number of disjunctions in the Horn clauses (resp. dual Horn clauses).

First, M constructs $G = \{d \,|\, \mathcal{M} \models c(d)\}$. Due to Proposition 5.2.6, we know that for normal default theories there is only one possible set of generating defaults in G. This set $G_1 \subseteq G$ can be constructed in polynomial time: M uses an arbitrary ordering of the defaults in G, call this tuple H.

M then scans the first default: if $p(d_1)$ is applicable, it adds the default to G_1 and deletes it from H. If not, nothing happens. Then M scans the next default with the same procedure. If H is not empty after the first run of the algorithm, the algorithm is called again starting with the new first default in H. The procedure stops if H is empty or if the algorithm was already called n times. After n runs ($n \leq m$; where n is the number of defaults in G), G_1 has been created (all remaining defaults in H have not applicable prerequisites). This algorithm is polynomial – it is called at most n times, and each time tests at most n defaults if $W \wedge c(G_1) \models p(d)$ is polynomial for EHNF default-theories; the check is split to $O(m^2)$ tautology checks for dual Horn theories, namely if $taut(\neg W_k \vee \neg c(G_1) \vee p(d))$. Checking if a dual Horn theory is a tautology is $O(m^2)$. There are at most $n^2 (n \leq m)$ steps of the algorithm, hence this step can be decided after $O(m^6)$ easy steps (i.e. steps, if a conjunction of literals infers (is a superset of) a conjunction of literals).

G_1 is the generating set of $\langle G, W \rangle$. To show that the extension $W \wedge c(G_1)$ is an extension of $\langle D, W \rangle$, M finally needs to test if no default of $D \backslash G$ is applicable, i.e. for all defaults $d \notin G_1$ M shows that $W \wedge c(G_1) \not\models p(d)$ or $W \wedge c(G) \wedge c(d)$ is inconsistent which is implemented in the loop for each $d \in B$ of Figure 4 (in case of normal default theories, each $j(d)$ has only one element which is a Horntheory) and is in $\mathbf{P}$, too.

If *yes*, then $\mathcal{M}$ is a model of $\langle D, W \rangle$, if *no*, then $\mathcal{M}$ is a not a model of $\langle D, W \rangle$. □

In Figure 4, a deterministic algorithm, which checks in polynomial time if $\mathcal{M} \models \langle D, W \rangle$ of a normal default theory, is represented (the not explicitly mentioned procedures *taut(dualHorntheory)?* and *sat(Horntheory)?* are feasible in polynomial time, too.).

Lemma 5.4.8 *Let $\mathcal{M}$ be an interpretation and $\langle D, W \rangle$ an EHNF normal default theory. Deciding whether $\mathcal{M} \models \langle D, W \rangle$ is $\mathbf{P}$-hard, even if $\langle D, W \rangle$ is either disjunction-free or prerequisite-free.*

Proof. We first show hardness in the case of disjunction-free default theories: We use a reduction of the $\mathbf{P}$-complete problem of reasoning in propositional not-free logic programming [Dantsin *et al.*, 1997], i.e. given an atom g and a logic program P, if g is in the stable model of P.

Let $\langle D, W \rangle$ be the default theory where each clause $b \leftarrow a_1 \wedge \ldots \wedge a_n$ (with each a_i and b being positive literals) of P is translated into a normal default $\frac{a_1 \wedge \ldots \wedge a_n : b}{b}$ and the default $\frac{:\neg g}{\neg g}$ is added.

Then the model $\mathcal{M}$ containing all atoms of the language is a model of the default theory iff there exists an extension without $\neg g$ (in all other defaults only positive literals occur) iff g can be inferred of the other defaults iff g is in the stable model of P.

Input: An interpretation $\mathcal{M}$, a disj.-free normal default theory $\langle D, W\rangle$.
Result: "Accept" if $\mathcal{M} \models \langle D, W\rangle$; "Reject" otherwise.

begin
$G := \{d \,|\, \mathcal{M} \models c(d)\}$; $G_1 := \emptyset$; $n := |G|$; $B = D \backslash G$;
$disw :=$ no. of disj. of Horntheories in W; $i := 0$;
while $i \leq n$ **do**
  begin
  $i := i + 1$;
  for each $d \in G$ **do**   % in arbitrary order
    begin
      $disp :=$ no. of conj. of dual Horntheories in $p(d)$; $x := 0$;
      for $k := 1$ **to** $disw$
      begin
      for $l := 1$ **to** $disp$
        begin
        if $taut(\neg W_k \vee \neg c(G_1) \vee p_l(d))$ **then** $x := x + 1$;
        end;
      end;
    if $x = disw \cdot disp$ **then** $G_1 := G_1 \cup \{d\}$ **and** $G := G \backslash \{d\}$;
    end;
  end;
for each $d \in B$ **do**
  begin
  $disp :=$ no. of conj. of dual Horntheories in $p(d)$; $x := 0$; $y := 0$;
  for $k := 1$ **to** $disw$
    begin
    for $l := 1$ **to** $disp$
      begin
      if $taut(\neg W_k \vee \neg c(G_1) \vee p_l(d))$ **then** $x := x + 1$;
      end;
    if $sat(W_k \wedge c(G_1) \wedge j(d))$ **then** $y := 1$;
    end;
  if $x = disw \cdot disp$ and $y = 1$ **then reject and halt**;
  end;
accept;
end.

Figure 5.5: *Deterministic Algorithm of Lemma 5.4.7*

The translation can be done in logarithmic space (the defaults are the program clauses and just a fixed number (namely, one) of defaults is added), hence this problem is $\mathbf{P}$-hard.

In the case of prerequisite-free EHNF default theories we use a reduction of the $\mathbf{P}$-complete problem if a Horn theory is satisfiable [Papadimitriou, 1993] (which is essentially the same if a logic program is satisfiable, because the program clauses are Horn clauses) and use the simple default theory $\langle \emptyset, W \rangle$ with W being a Horn theory. $\square$

Theorem 5.4.9 *Let $\mathcal{M}$ be an interpretation and $\langle D, W \rangle$ an EHNF normal default theory. Deciding whether $\mathcal{M} \models \langle D, W \rangle$ is $\mathbf{P}$-complete, even if $\langle D, W \rangle$ is either disjunction-free or prerequisite-free.*

If a normal default-theory is disjunction-free *and* prerequisite-free the problem no longer remains $\mathbf{P}$-hard and can easily be seen to be in nondeterministic logarithmic space:

Theorem 5.4.10 *Let $\mathcal{M}$ be an interpretation and $\langle D, W \rangle$ a disjunction-free normal prerequisite-free default theory. Deciding whether $\mathcal{M} \models \langle D, W \rangle$ is in $\mathbf{L}$.*

Proof. Given an interpretation $\mathcal{M}$ and a default theory, first check which defaults are good and form the set G - these are simple subset-checks. Form the conjunction of all good consequents, which is a conjunction of literals.

Then check for each bad default if it is not consistent with this literal conjunction. Hence, $\mathcal{M}$ is not a model of the default theory exactly if $(g \wedge b_1) \vee (g \wedge b_2) \vee \cdots \vee (g \wedge b_m)$ is satisfiable – where g is the conjunction of all literals of the good defaults, and b_j are conjunctions of literals of each consequent of bad defaults.

Hence, model checking for such default logics is the converse problem of SAT for DNF (with additional subset-checks at the beginning possible to be carried out in parallel time). Therefore this problem is in $\mathbf{L}$, too. $\square$

Let us intuitively explain the results of this chapter, the previous chapter and the results in [Liberatore and Schaerf, 1998]: For general default theories, model checking has two sources of hardness. On the one hand, we cannot construct a set of generating defaults w.r.t. a given model in polynomial time: We can construct a set $G \subseteq D$ of defaults with $\mathcal{M} \models c(d)$ for each $d \in G$ – however, in general, there are $2^{|G|}$ subsets, which are possible sets of generating defaults (one default may block another one) and we need to take a look at all of these. On the other hand, the other source of intractability w.r.t. a guessed set of generating defaults is the hardness of the classical inference problem (which is used for checks of consistency and if prerequisites follow from use of other defaults).

In the case of normal default theories there is, due to Proposition 5.2.6, w.r.t. one fixed model only one set $G_1 \subseteq G$ which is maybe a set of generating defaults. Therefore the initial guessing stage can be eliminated and G_1 can be constructed in polynomial time and the problem is hence in Δ_2^P

(and as shown in Theorem 5.3.1, $\Delta_2^P[O(\log n)]$-complete). In the case of prerequisite-free normal default-theories, just one oracle call remains and the problem is in $\mathbf{coNP}$.

If we restrict the defaults in a way that satisfiability and inference is polynomial, the other source of intractability is affected: If all inference problems are as in $\mathbf{P}$ the problem is, due to the initial guessing stage, in $\mathbf{NP}$. If such a default theory is restricted to normal defaults model checking complexity is polynomial, because the initial guessing stage can be eliminated, too.

Reasoning in EHNF is one level easier than with arbitrary default theories, too, and as model checking is one level easier than reasoning for normal default theories, a polynomial algorithm for model checking for normal EHNF default theories was derived. The class of EHNF default theories is a large class of default theories where model checking remains polynomial (in the case of normal default theories) and hence this tractability result seems very useful for practical applications.

5.5 Complexity of Model Checking for Default Theories with Weak Extensions

The concept of weak extensions was introduced in Section 2.5.4.

We call $\mathcal{M}$ a *weak model* of $\langle D, W\rangle$ iff $\mathcal{M}$ is a model of at least one (consistent) weak extension. Due to the well-known fact that every extension is a weak extension, too, it easily follows that every model of $\langle D, W\rangle$ is a weak model of $\langle D, W\rangle$. However, the converse does not hold – it would hold, if every extension were a subset of a weak extension. However, this is in general not the case; consider the following example:

$$W = \emptyset \quad D = \left\{ \frac{a : b}{b}, \frac{b : a}{a}, \frac{: \neg a}{\neg a} \right\}.$$

This default theory admits the weak extension $cons\{a, b\}$ which is, however, no extension and has no subset which forms one. Therefore the interpretation $\{a, b\}$ is only a weak model, but not a model of this default theory. Hence, the hardness and membership results for model checking for normal default theories cannot be directly applied for weak model checking, but have to be modified and yield the interesting result that weak model checking does not become easier with normal default theories.

5.5.1 General Case

Theorem 5.5.1 *Let $\mathcal{M}$ be an interpretation and $\langle D, W\rangle$ a default theory. Then, the problem whether $\mathcal{M}$ is a weak model of $\langle D, W\rangle$ is Σ_2^P-complete, even if $\langle D, W\rangle$ is semi-normal and prerequisite-free.*

Proof. The hardness result follows immediately of the proof of Theorem 2 in [Liberatore and Schaerf, 1998], where Σ_2^P-hardness of model checking with extensions is shown using a prerequisite-free default theory, and the fact that in prerequisite-free default theories extensions coincide with weak extensions.

Membership can be shown by initially guessing a set $G \subseteq D$: Then it is tested whether

$$(W \wedge c(G)) \equiv W \cup \{c(d) \mid d \in D, W \wedge c(G) \models p(d),$$
$$W \wedge c(G) \not\models \neg\beta_i \quad \forall \beta_i \in j(d)\}$$

which can be done in polynomial time with access to an oracle in $\mathbf{NP}$ for the satisfiability and inference checks. If the equation is true, then $E = cons(W \wedge c(G))$ is a weak extension. Finally, it is tested in polynomial time if $\mathcal{M} \models W \wedge c(G)$. Therefore the problem is in $\mathbf{NP}^{\mathbf{NP}}$. □

5.5.2 Normal Default Theories

One might ask, if the membership proof of Theorem 5.3.1 for model checking for normal default theories can in a similar way be applied to weak model checking: However, this is not the case, because Proposition 5.2.6 does not hold for weak extensions. But there is at least a unique largest generating set and a unique smallest generating set (corresponding to the extension of $\langle G, W\rangle$) of the jointly compatible defaults of G.

Consider the following default theory $\langle D, W\rangle$ with

$$W = \emptyset, D = \left\{ \frac{a : b}{b}, \frac{b : a}{a}, \frac{c : d}{d}, \frac{d : c}{c}, \frac{c : \neg e}{\neg e}, \frac{: \neg b}{\neg b} \right\}$$

and the model $\{a, b, d, c, e\}$. The first four defaults form the set G and the smallest weak generating set w.r.t. G is the empty set, the largest is the set $\{a, b, c, d\}$. However, in both cases one of the bad defaults fires and $\neg b$ and $\neg e$ can be inferred, respectively. Nevertheless, $\mathcal{M}$ is a model of this default theory, namely of the extension $cons\{a, b\}$.

Therefore, it is neither sufficient to look at the smallest generating set of G, nor at the largest, and the problem of Step 2 turns into a procedure where an exponential number of different sets deserve to be looked at.

Even if it would be sufficient to consider the largest weak generating set (then step 3 and step 4 could be left unchanged) there is no polynomial algorithm with calling only a logarithmic number of an $\mathbf{NP}$-oracle to compute the cardinality of this largest set G_1: The problem if $|G_1| \geq r$ is equivalent to $\exists G_1' \,\forall d \in G_1' : \; W \wedge c(G_1') \models p(d)$. Recall the discussion in the first claim of Theorem 5.3.1, where we showed the importance of "$\not\models$" together with the existential quantification. If we try to guess interpretations in a data-structure together with a set G_1', the problem looks like $\exists G_1' \forall I \ldots \Rightarrow \ldots$ and, due to the quantifier alternation, guessing a data-structure is impossible.

Moreover, we will show that the problem is even Σ_2^P-hard as all possible weak generating sets have to be considered.

The non-constructive nature of weak extensions allows an easier characterisation of weak extensions than with the iterative characterisation of extensions; however, the constructive nature of extensions allows an easier approach to algorithms as the different complexity results in the case of normal default theories with prerequisites shows.

Theorem 5.5.2 *Let $\mathcal{M}$ be an interpretation and $\langle D, W\rangle$ a normal default theory. Then the problem to decide whether $\mathcal{M}$ is a weak model of $\langle D, W\rangle$ is Σ_2^P-complete.*

Proof. Only hardness remains to be shown as membership has been shown in the previous subsection for arbitrary default theories. We use a reduction from the Σ_2^P-complete problem $\mathrm{QBF}_{2,\exists}$, this is the problem if there exists an evaluation of the $p_i \in P$ such that for each evaluation of the $q_j \in Q,\ F(P, Q)$ is true (abbreviated as: $\exists P \forall Q\, F(P, Q)$ with second order quantification).
Let $\langle D, W\rangle$ be $W = \emptyset$,

$$D = \bigcup_i \left\{ \frac{w \Rightarrow p_i : w \Rightarrow p_i}{w \Rightarrow p_i}, \frac{w \Rightarrow \neg p_i : w \Rightarrow \neg p_i}{w \Rightarrow \neg p_i} \right\} \cup \left\{ \frac{\neg w : v}{v}, \frac{: w \wedge \neg F}{w \wedge \neg F}, \right\}$$

with w and v being new propositional atoms.
This transformation is feasible in polynomial time. We show that $\mathcal{M} = \emptyset$ is a model of $\langle D, W\rangle$ iff $\exists P \forall Q\, F(P, Q)$.

- $\exists P \forall Q\, F(P, Q)$ implies that there is a valuation of the p_i which makes F true for all evaluations of the q_j. Due to the non-constructive nature of weak extensions we are allowed to choose which of the first two defaults for each i fire – we can choose that none, one or both of them fire, as their consequents allow to derive their own prerequisites. In the required extension exactly one consequent for each i shall be in the context, namely the one corresponding to the truth evaluation. Then the last default cannot be applied, as assuming $w \wedge \neg F$ is inconsistent ($\neg(w \wedge \neg F) \equiv w \Rightarrow F$ can be inferred of a context containing for each i either $w \Rightarrow p_i$ or $w \Rightarrow \neg p_i$ exactly if $w \Rightarrow F$ can be inferred and hence, exactly if F is valid under a valuation of the p_i), and the default yielding v cannot be used either, hence $\emptyset$ is a model.
- Assume, $\emptyset \models \langle D, W\rangle$. Then the last two defaults shall not be used. W.r.t. extensions the final default would always be used as it is the only prerequisite-free default. However, w.r.t. weak extensions a suitable combination of the other defaults may forbid to use it. Of the first defaults for each i one, two or no default can fire. The case that for some i no default fires is not interesting, as the final default then can be used (unless for all undefined i an evaluation of the the p_i exists that F is valid for all evaluations of the not fixed p_i, however, then also the weaker result holds that an evaluation exists). If more than one default for some i is used, then $\neg w$ can be inferred, but the default yielding v needs to be used, and hence $\emptyset$ would not be a model. Therefore, use exactly one default for each i. So, we need a context like this:

$$\bigcup_i \left\{ (w \Rightarrow p_i) \mid p_i \in P\} \cup \bigcup_i \{(w \Rightarrow \neg p_i) \mid \neg p_i \in P \right\}$$

with P is an evaluation making F valid (i.e. true for each evaluation of the q_j). This context allows to use the defaults yielding $w \Rightarrow \neg p_i$ or $w \Rightarrow p_i$, respectively. Only this context disallows using the default yielding $\neg F \wedge w$, since $\neg F \wedge w$ is inconsistent (if w is assumed, then the p_i and $\neg p_i$ are inferred due to the context and F is valid, hence $\neg F \wedge w$ is inconsistent). Therefore context and inferred formulae coincide, the context forms a weak extension and $\mathcal{M}$ is a model of it (since $\neg w$ is $true$ in the model, the p_i can have any value *in the model*).

Hence, $\mathcal{M} = \emptyset$ is a model of $\langle D, W \rangle$ iff $\exists P \forall Q\, F(P, Q)$. □

Theorem 5.5.3 *Let $\mathcal{M}$ be an interpretation and $\langle D, W \rangle$ a normal prerequisite-free default theory. Then, the problem to decide whether $\mathcal{M}$ is a weak model of $\langle D, W \rangle$ is* $\mathbf{coNP}$*-complete.*

Proof. Since the theory is prerequisite-free, extensions coincide with weak extensions and therefore the proof of Theorem 7 in [Liberatore and Schaerf, 1998] for model checking w.r.t. extensions can be used. □

5.5.3 EHNF default theories

Theorem 5.5.4 *Let $\mathcal{M}$ be an interpretation and $\langle D, W \rangle$ an EHNF default theory. Then, the problem to decide whether $\mathcal{M}$ is a weak model of $\langle D, W \rangle$ is* $\mathbf{NP}$*-complete even if the theory is disjunction-free and either semi-normal and prerequisite-free or normal.*

Proof. The membership proof for arbitrary disjunction-free default theories is essentially the same as the membership proof in Theorem 5.5.1 without access to an oracle in $\mathbf{NP}$ for satisfiability and inference checks, and hence this problem is in $\mathbf{NP}$ (the satisfiability and inference checks are tractable in an analogous way to the corresponding proofs of Section 5.4, only G is used instead of G_i for inference checks).

For semi-normal, prerequisite-free default theories the hardness proof is exactly the same as in Lemma 5.4.5; to show hardness for normal default theories (with prerequisites) we have to modify the corresponding proof to achieve normalization using the power of weak extensions.

We use a reduction from 3SAT: Let $F(t_1, \ldots, t_m)$ be a formula in conjunctive normal form with exactly three literals per clause:

$$F = F_1 \wedge \ldots \wedge F_i \wedge \ldots \wedge F_n$$

with $F_i = x_{i1} \vee x_{i2} \vee x_{i3}$ (the x_{ij} are literals, and the t_k are propositional atoms). Deciding if F is satisfiable is $\mathbf{NP}$-complete.

Let $\langle D, W \rangle$ be $W = \emptyset$ and

$$D = \bigcup_k \Big\{ \frac{\neg u_k : \neg u_k}{\neg u_k}, \frac{\neg t_k : \neg t_k}{\neg t_k}, \frac{\neg u_k \wedge \neg t_k : z}{z} \} \cup$$

$$\bigcup_i \{ \frac{f(x_{i1}) : \neg y_i}{\neg y_i}, \frac{f(x_{i2}) : \neg y_i}{\neg y_i}, \frac{f(x_{i3}) : \neg y_i}{\neg y_i}, \frac{: y_i}{y_i} \}$$

with $f(x_{ij}) = \neg t_k$ if $x_{ij} = \neg t_k$ and $f(x_{ij}) = \neg u_k$ if $x_{ij} = t_k$.

We prove that F is satisfiable iff $\mathcal{M} = \emptyset$ is a model of $\langle D, W \rangle$.

Due to the "self-referential" power of weak extensions we achieve that a context may include for each k $\neg u_k$ and $\neg t_k$ or just one of these, or even none. However, if both are, for some k part of the context, then it is forced that v is inferred, and hence $\emptyset$ is not a model.

If a context is chosen which for some k neither includes $\neg t_k$ nor $\neg u_k$, then some of the required prerequisites $f(x_{ij})$ maybe cannot be used, and hence, all we can achieve with such a context to prohibit that a default yielding a positive y_i needs to be used can be done with a context, where for each k exactly one of $\neg u_k$ or $\neg t_k$ is in the context, too.

Hence, choose a context with for each k either $\neg t_k$ or $\neg u_k$ (and including all $\neg y_i$): F is satisfiable iff this evaluation of F is chosen in the context, which makes all $\neg y_i$ inferrable, and therefore only negated literals are in this extension; and if F is not satisfiable it is necessary to include some y_i as in the corresponding proof for extensions. So it can be concluded that $\emptyset \models \langle D, W \rangle$ iff F is satisfiable. □

Theorem 5.5.5 *Let $\mathcal{M}$ be an interpretation and $\langle D, W \rangle$ a prerequisite-free normal EHNF default theory. Then the problem to decide if $\mathcal{M}$ is a weak model of $\langle D, W \rangle$ is $\mathbf{P}$-complete.*

Proof. This proof follows immediately of Lemma 5.4.7 and 5.4.8 and the fact that in prerequisite-free default theories weak extensions and extensions coincide. □

5.6 Tautology Checking

Definition 5.6.1 *A default theory is a tautology if every possible interpretation is a model of the default theory.*

Hence, the goal is to find out, if, for each interpretation, one can find an extension which satisfies the interpretation. The converse problem is to find out if an interpretation exists which is not a model of at least one extension.

Thus, the straightforward algorithm is guessing an interpretation and then showing that it is not a model. Showing that an interpretation is not a model of a default theory is in Π_2^P, because it is the converse problem of model checking. Hence, the converse problem of tautology checking admits Σ_3^P as upper bound, therefore tautology checking is in Π_3^P.

Compare this to satisfiability checking: this is in general in Σ_2^P, since it coincides with the question if a default theory admits a (consistent) extension. If it does, it certainly has a model. Furthermore, every normal default theory is satisfiable.

5.7 Complexity of Model Checking with AEL and N

In Sections 2.5.5 and 2.5.6 we described autoepistemic logic (AEL) and non-monotonic logic **N** in detail – for more information the reader is referred to [Marek and Truszczyński, 1989] and to [Gottlob, 1992] for definitions and complexity results w.r.t. reasoning in AEL and **N**.

$\mathcal{M}$ is a propositional AEL-model of a set of premises in $\mathcal{L}_m$ (the propositional language extended with the modal operator L) iff $\mathcal{M}$ satisfies an objective part of at least one stable expansion.

$\mathcal{M}$ is a propositional **N**-model of a set of premises in $\mathcal{L}_m$ iff $\mathcal{M}$ satisfies an objective part of at least one **N**-expansion.

It is well known that the weak extensions of a default theory correspond to objective (i.e. propositional without the modal operator) parts of the stable extensions of a polynomial translation of this default theory into an autoepistemic theory. Additionally, it is known that extensions of a default theory correspond to objective parts of **N**-expansions in the same way [Marek and Truszczyński, 1993]. Hence, from the results of the previous chapters we immediately get the following Corollaries:

Corollary 5.7.1 *Let Σ be a set of premises in $\mathcal{L}_m$ and $\mathcal{M}$ an interpretation. Deciding if $\mathcal{M}$ is a propositional AEL-model is Σ_2^P-hard.*

Corollary 5.7.2 *Let Σ be a set of premises in $\mathcal{L}_m$ and $\mathcal{M}$ an interpretation. Deciding if $\mathcal{M}$ is a propositional* **N***-model is Σ_2^P-hard.*

To prove membership we use the results of Gottlob [1992] w.r.t. reasoning.

Lemma 5.7.3 *Let Σ be a set of premises in $\mathcal{L}_m$ and $\mathcal{M}$ an interpretation. Deciding if $\mathcal{M}$ is a propositional AEL/***N***-model is in Σ_2^P.*

Proof. In [Gottlob, 1992] it is shown that the problem if a formula is not occurring in all expansions (the converse problem of cautious reasoning) is in Σ_2^P (using the finitary characterisation of Niemelä [1991] with modal subformulae). Consider the propositional language $\{a_1, \ldots, a_n\}$ and any model $\mathcal{M}$ containing some of these atoms. Then $\mathcal{M} \models E$ iff $E \not\models L_1 \vee \ldots \vee L_n$ with $L_i = \neg a_i$ if $a_i \in \mathcal{M}$ and $L_i = a_i$ if $a_i \notin \mathcal{M}$. Therefore the model checking problem is a special instance of this problem. The proof is analogous for the case of **N**-expansions. $\square$

Hence, both, propositional model checking with autoepistemic logic and with non-monotonic logic **N** is Σ_2^P-complete.

5.8 About Translations of Default Logic into Disjunctive Logic Programming

We would like to sketch some known results concerning the comparison of the expressive power of default logic and disjunctive logic programming and sketch some new negative results arising from complexity results of [Liberatore and Schaerf, 1998] and our work. In the spirit of [Gottlob, 1995b; Janhunen, 1998] let us reformulate the definition of pfm-functions of Section 4.9.2:

Definition 5.8.1 (pfm-function) *A pfm-function $f : A \mapsto B$ is a function translating a non-monotonic formalism A into a non-monotonic formalism B fulfilling the additional criteria of polynomiality, faithfulness and modularity as described in Section 4.9.2.*

In [Janhunen, 1998] a weaker concept of faithfulness is proposed instead ignoring inconsistent extensions and allowing the introduction of new variables. In [Gottlob, 1995b] for instance, a non-modular translation function from default logic into autoepistemic logic is given.

Definition 5.8.2 *The non-monotonic formalism B is at least as expressive as A if a pfm-function f exists which maps all theories of A into theories of formalism B.*

To learn more about logic programming, the reader is referred to section 2.5.7. Here, we consider the stable models semantics of disjunctive logic programming (DLP) and the answer set semantics of extended disjunctive logic programs (EDLP) and normally view a logic program as a collection of its ground clauses.

As discussed, in DLP with the stable models semantics cautious reasoning is Π_2^P-complete and brave reasoning Σ_2^P-complete [Eiter and Gottlob, 1995] as in default logic, which gives hope for a translation which fulfills the above criteria. However, the high complexity of inference in DLP has different reasons: reasoning is defined via models and model checking is an easier (and more natural) task than in default logic. Model checking in DLP is $\mathbf{coNP}$-complete for several semantics and in $\mathbf{P}$ for non-disjunctive programs (Horn-clauses).

Extended programs have two kinds of negation: the negation-by-default and true negation. Stable models are a subset of the minimal models if the negation-by-default is interpreted as classical negation. In extended disjunctive programs answer sets are considered, extending the concept of stable models. Answer sets are like partial models: some atoms are negated, some are not negated and some are left undefined. Answer set checking is $\mathbf{coNP}$-complete and for non-disjunctive extended logic programs in $\mathbf{P}$ – there are no Horn-clauses any longer, however, unlike the stable models semantics, one does not need to look at the $\leq$-relation, but only onto the subset relation: Does there exist $S' \subset S$ with S closed under $Red(S, P)$? This is in $\mathbf{coNP}$ due to the choices in the effective disjunction: without disjunctions, no choices are left and it is very easy to determine if a subset of S is closed under $Red(S, P)$.

In the literature, polynomial, modular and faithful translations from logic programming with stable semantics into default logic have already been explored [Inoue and Sakama, 1993] in the sense that stable models are identified with default logic extensions. In [Gelfond *et al.*, 1991] and [Inoue and Sakama, 1993], an embedding from EDLP into disjunctive default theories is presented, where answer sets can be identified with extensions[3]:

Definition 5.8.3 *[Definition 4.1 in [Inoue and Sakama, 1993]] Let P be a disjunctive logic program. Then its associated default theory D_P is constructed as follows:*

- *Each clause $a_1 \vee \ldots \vee a_l \leftarrow b_1 \wedge \ldots b_m \wedge not\, b_{m+1} \wedge \ldots \wedge not\, b_{m+n}$ in P is transformed into the default*
$$\frac{: \neg b_{m+1}, \ldots, \neg b_{m+n}}{b_1 \wedge \ldots \wedge b_m \Rightarrow a_1 \vee \ldots \vee a_l}.$$
- *For each atom in P, the following closed world assumption default is in D_P: $(: \neg a / \neg a)$.*
- *Nothing else is in D_P.*

Theorem 5.8.4 *(Theorem 4.5 in [Inoue and Sakama, 1993]) Let P be a logic program and D_P be its associated default theory. $\mathcal{M}$ is a stable model of P iff $cons(At_{\mathcal{M}})$ is an extension (with $At_{\mathcal{M}}$ being the atoms in $\mathcal{M}$).*

In [Inoue and Sakama, 1993], a pfm-translation from EDLPs into default logic is presented. In [Gelfond *et al.*, 1991], a translation from EDLPs into disjunctive default logic is introduced:

Theorem 5.8.5 *(Theorem 7.2 in [Gelfond* et al.*, 1991]) Let P be an extended disjunctive logic program. A set of literals M is an answer set of P if and only if M is the set of all literals from an extension for $emb(P)$ into the disjunctive default theory with $emb(r)$ for each rule $r = L_1 | \ldots | L_l \leftarrow L_{l+1} \wedge \ldots L_m \wedge not\, L_{m+1} \wedge \ldots \wedge not\, L_n$ defined as:*

$$emb(r) = \frac{L_{l+1} \wedge \ldots \wedge L_m : \overline{L}_{m+1}, \ldots \overline{L}_n}{L_1 | \ldots | L_l}$$

with $\overline{L}$ is the opposite literal of L.

These translation functions show that default logic and disjunctive default logic are at least as expressive as logic programming. However, one may ask if there is any chance to establish a pfm-function *into the other direction* (maybe with slightly changing the criteria of the function) at least for some subclass of default logic.

In the above translations stable models are treated like "total" extensions and compared with the extensions of default logic. However, the concept of faithfulness could also be defined in the terms of models (see below, p$\overline{\text{f}}$m-functions).

[3]The following quotations slightly changes the wording used in the mentioned articles.

With the usual concept of faithfulness, it is easy to see that default logic is more expressive than disjunctive logic programming, since the default theory $\left\{\frac{:x}{x}, \frac{:\neg x}{\neg x}\right\}$ has the two extensions $cons\{x\}$ and $cons\{\neg x\}$; however, no disjunctive program with the stable semantics can have the models $\{x\}$ and $\emptyset$, since every stable model is a minimal model, too; at least this problem could be solved with introducing a new variable x' to represent $\neg x$ in a disjunctive program, however other problems remain (introducing new variables is similar to using EDLPs instead of DLPs) – in default logic extensions do not have to be total, but models of DLP are total (this could be solved with answer sets); however, in default logic an extension can be of the form $\{a \vee b\}$ and we cannot find a corresponding answer set of extended logic programming (and would need to compare more then one answer set with one extension).

Since we require a polynomial translation function, known complexity results can help to clarify matters: Model checking with stable models is $\mathbf{coNP}$-complete – however, in the translation of the DLP into default logic the task is not to compare this complexity with the complexity of model checking in default logic. The corresponding task which needs to reside in the same class for a polynomial function is *extension checking* (that is, for a set G of defaults check if $cons(W \wedge c(G))$ is an extension). Refer to [Rosati, 1999] for complexity results about extension checking; in general, extension checking can be displayed as an $\mathbf{NP}$-tree for standard default theories and hence the problem is $\Delta_2^P[O(\log n)]$-complete, which is easier than model checking because model checking is in this setting based on *extension finding*.

In disjunction-free default theories the task is tractable, since all oracle-calls turn into polynomial problems (see also [Gottlob, 1992] and [Selman, 1990; Kautz and Selman, 1991] for related results).

Instead of directly comparing the complexity of *model checking* in DLP with the corresponding task of *extension checking* (as the corresponding task in a faithful translation is not model checking, since models of DLP are treated as extensions), the model checking results of the present report and of [Liberatore and Schaerf, 1998] immediately clarify the matter of a pfm-translation from disjunctive logic programming into the default logic formalism as illustrated in the following theorem:

Theorem 5.8.6 *There exists no pfm-function mapping default logic, normal default logic or disjunction-free default logic into disjunctive logic programming unless the polynomial hierarchy collapses (resp. unless* $\mathbf{NP} = \mathbf{coNP}$ *for disjunction-free default logic).*

Proof. Assume the existence of a pfm-function embedding default logic into disjunctive logic programming. Due to known results of pfm-functions translating disjunctive logic programming into default logic, both formalism would be equally expressive and hence admit the same models (extensions in logic programming are defined via models).

However, as model checking with logic programming formalism is only $\mathbf{coNP}$-complete, this would imply that $\Sigma_2^P = \mathbf{coNP}$, i.e. that the polynomial hierarchy collapses. Even in the case of normal default theories, this is not possible, as model checking for normal default theories is $\Delta_2^P[O(\log n)]$-complete, and would imply that $\Delta_2^P[O(\log n)] = \mathbf{coNP}$. Restricting default

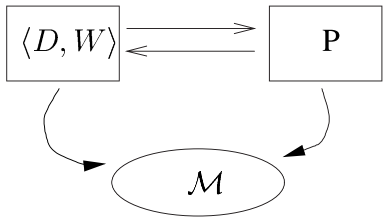

Figure 5.6: *Comparing Models of DLP and Default Logic*

theories to conjunctions of literals and assuming that a pfm-function exists from disjunction-free default logic into disjunctive logic programming would imply that $\mathbf{NP} = \mathbf{coNP}$. $\square$

The complexity results of model checking for default logic only leave it possible that prerequisite-free, normal default theories can be translated with a pfm-function into the disjunctive logic programming formalism. However, it remains uncertain if such a translation would be strongly modular, i.e. allowing to translate each default rule independently into one or more logic program clauses and faithful in the sense of [Gottlob, 1995b].

Looking at EDLPs and the answer set semantics, the results are very similar - the above example $\left\{\frac{:x}{x}, \frac{:\neg x}{\neg x}\right\}$ can be translated into an EDLP with the two answer sets $\{x\}$ and $\{\neg x\}$ without the need of introducing new variables. However, as mentioned above, answer sets are only allowed to contain literals, while extensions in default logic can contain disjunctions: the default theory $\{: a \vee b/a \vee b\}$ with the unique extension $cons\{a \vee b\}$ cannot have a corresponding answer set – one would need to compare one extension with more than one answer set.

As model checking for EDLP is $\mathbf{coNP}$-complete, too [Eiter and Gottlob, 1995], the above Theorem equally applies to EDLPs:

Theorem 5.8.7 *There exists no pfm-function mapping default logic into extended disjunctive logic programming unless the polynomial hierarchy collapses.*

However, as Theorem 5.8.5 shows, there exists a pfm-function from EDLP into a very limited subclass of *disjunctive* default logic (where prerequisites are conjunction of literals, justifications and consequents are single literals). This can happen because extension checking in disjunctive default theories is more complex than for standard default theories; even in disjunction-free disjunctive default theories the task is in $\mathbf{coNP}$. Due to this complexity, the above mentioned bijective pfm-translation from EDLP to disjunction-free disjunctive default theories with only literals as consequents and literals as justifications is possible.

Proceeding in our aim to find a function mapping default theories into EDLPs or DLPs, we could alter the concept of faithfulness into a concept where the models of default logic are identified with the models of DLPs or answer sets of EDLPs (of course there are default logics with the same models but different extensions, these are treated as one, as in the usual concept syntactically different default theories with the same extensions are treated; and in this case even "semantical" different default theories form an equivalence class). This concept is somehow related to the concept of succinctness as presented in section 4.9.1, however here we require a polynomial transformation and an independent translation of a propositional part (or even strong modularity – an independent translation of each default for instance).

Definition 5.8.8 *A translation function is a p$\bar{f}$m-function if it maps theories in A into equivalent theories in B, with "equivalent" meaning that the theory in A and the corresponding theory in B admit the same (partial) models and the translation is polynomial and modular.*

In the above example, $\{a\}$, $\{b\}$ and $\{a, b\}$ are models; these three cannot together be stable models of a DLP due to the minimality criterion. Instead of dealing with auxiliary variables to solve this problem, we take a look at the wider concept of answer sets of extended logic programs. These three models of a default theory can in EDLP be identified with the answer sets $\{a, \neg b\}$, $\{b, \neg a\}$ and $\{a, b\}$ – so one just has to find a corresponding EDLP – maybe in more than two variables (leaving the others, auxiliary variables, undefined in these three answer sets.).

Since answer set checking is $\mathbf{coNP}$-complete, there is only a chance for a polynomial function if we take a look at limited default theories where model checking is $\mathbf{coNP}$-complete, too (and we have to look if model finding is equally hard, too, i.e. that answer set finding has the same complexity as model finding for default theories). The results of [Liberatore and Schaerf, 1998] and this report settled the questions of model checking for several restrictions and indicate that prerequisite-free normal default theories might be one limitation we seek; the question is if we can find a modular translation.

In general, not all corresponding answer sets can be total w.r.t. an extended program of a translation of a normal prerequisite-free default theory, because at the end of section 6.1 we stated that if we know that all answer sets are total sets, then answer set checking is polynomial and thus there would not exist a p$\bar{f}$m-function.

To get some ideas as to how an embedding of prerequisite-free normal default theories into EDLPs might be possible, we consider some examples: The default theory $\{: a \vee b / a \vee b\}$ has one extension and three models, namely $\{a\}$, $\{b\}$ and $\{a, b\}$. One extended logic program which has the answer sets $\{a, \neg b\}$, $\{b, \neg a\}$ and $\{a, b\}$ is

$$\neg b \leftarrow \text{not}\, b, \text{not}\, \neg a \quad ; \quad \neg a \leftarrow \text{not}\, a, \text{not}\, \neg b \quad ;$$

$$a \leftarrow \text{not}\, \neg a \quad ; \quad b \leftarrow \text{not}\, \neg b.$$

Another one, including disjunctions and integrity clauses (i.e. empty head) would be:

$$a | \neg a \leftarrow \quad ; \quad b | \neg b \leftarrow \quad ; \quad \leftarrow \neg a \wedge \neg b.$$

However, this translation would not be modular if we add the facts $\neg a$ and $\neg b$.

The default theory $\{: a \vee b / a \vee b, : \neg a \wedge \neg b / \neg a \wedge \neg b\}$ has four models (it is a tautology as this theory allows all possible models with two atoms) and this is a corresponding extended program:

$$\neg b \leftarrow \text{not}\, b \quad ; \quad \neg a \leftarrow \text{not}\, a \quad ;$$

$$a \leftarrow \text{not}\,\neg a \quad ; \quad b \leftarrow \text{not}\,\neg b$$

with four answer sets (hence all with no undefined literals). Another program with the same answer sets would be $a|\neg a \leftarrow; b|\neg b \leftarrow$, which is the same as in the above example just leaving out the integrity clause.

However, with more defaults involved a general modular translation function will be, if possible, rather difficult to specify. The effective disjunction needs to "play the role of deciding which model or extension is chosen"- the default theory $\{: a/a, : \neg a/\neg a\}$ is clearly mapped into $a|\neg a \leftarrow$ in such an embedding; and different to the examples, in the general case auxiliary atoms have to be introduced (then answer set checking is at the same complexity level as model checking for normal prerequisite-free default theories). It remains to be seen, how more complex defaults can be translated and highly uncertain if the translation is strongly modular, i.e. if each default can be translated independently.

The translation is equivalent to the question of a modular translation of prerequisite-free normal default theories into disjunction-free disjunctive default theories (due to the above mentioned embedding) with faithfulness as comparing models of prerequisite-free normal default theories with extensions of disjunction-free disjunctive default theories.

To find a translation several problems have to be overcome. First, how will defaults involving disjunctions be presented? The easy default theory $D = \{: p \vee q \vee r/p \vee q \vee r\}$ can be translated into the following logic program:

$$p|q|r \leftarrow \quad ; \quad p|\neg p \leftarrow \text{not}\,\neg q \quad ; \quad p|\neg p \leftarrow \text{not}\,\neg r \quad ;$$
$$q|\neg q \leftarrow \text{not}\,\neg p, \quad ; \quad q|\neg q \leftarrow \text{not}\,\neg r, \quad ;$$
$$r|\neg r \leftarrow \text{not}\,\neg p, \quad ; \quad r|\neg r \leftarrow \text{not}\,\neg q.$$

The models of this default theory are exactly the answer sets of this logic program. Arbitrary formulae would have to be split inductively and auxiliary variables introduced to "call subroutines".

Even more difficult is the task to consider how a set of defaults can be translated into a set of program clauses based on the intersection and the union of logic programs if we try to find a modular translation. The problem is that we sometimes want the intersection, and sometimes the union depending if given two defaults are compatible or not, so either we have to compare all defaults with all defaults, which is exponential, or we need to know all maximal compatible default-sets, which is exponential, too.

However, one can construct a translation of each normal prerequisite-free default theory $D = \{D_1, \ldots, D_n\}$ with $D_i = \frac{:d_i}{d_i}$. Now each two defaults can be compared with each other to find if d_i and d_j are inconsistent with each other and if both are consistent. Because then, they are not both part of one extension. With $O(n^2)$ steps all defaults can be compared. If one default is consistent with all other defaults, then it shall be in all extensions. Both rules can be formalised in disjunctive default logic as:

$$D = \bigcup_{i,j} \left\{ \frac{\neg d_i \vee \neg d_j :}{d_i | d_j}, \frac{: d_i \wedge d_1, \ldots, d_i \wedge d_n}{d_i} \right\}$$

The prerequisite of the first default states that $\neg d_i \vee \neg d_j$ shall be a tautology, equivalently if $\neg(d_i \wedge d_j)$ is a tautology, i.e. the default is only used if $(d_i \wedge d_j)$ is inconsistent. The second rule says that if a default is compatible with all others, then it is in all extensions. However, this method only works if each d_i is consistent, and one would have to add other rules for that, too. Anyway, this translation from normal prerequisite-free default theories cannot be used as a step in a translation into EDLP, because in this translation, extensions are compared with extensions, and not models with extensions, and the formulae d_i are still not cut into literals needed in logic programs, probably due to the wrong comparison criterion to compare extensions with extensions. It does not help either to use any representation theorem, since a translation in an equal default theory which only consists of conjunctive normal forms may be exponential, too.

A different, alternative approach would be to compare subclasses of default logic with EDLPs via representational succinctness [Gogic *et al.*, 1995] as introduced in section 4.9.1.

Rosati [1999] proved that extension checking is $\mathbf{coNP}$-complete, too, in the case of normal default theories; hence one could think of a translation comparing extensions of default logic with answer sets of EDLPs instead – and with having our thoughts of the above in mind in trying to find a suitable mapping.

Chapter 6

Complexity of Model Checking for Disjunctive and Stationary Default Logics

In this chapter, after defining some concepts like normal, limited and other restrictions of disjunctive default theories, we settle the question of the complexity of model checking for disjunctive default theories (in the next section), and show, that although reasoning is Σ_2^P-complete even for disjunction-free disjunctive default theories, model checking is sometimes an easier task, however due to the effective disjunction, the complexity remains on the same level for theories with prerequisites. Moreover, we generalize the concept of weak extensions to disjunctive default theories in a way which is favourable from the perspective of complexity. In Section 6.3 we proceed with investigating the computational complexity of model checking for default theories under the stationary semantics.

6.1 Complexity of Model Checking for Disjunctive Theories

A *disjunctive default* is an expression of the form

$$\frac{\alpha : \beta_1, \ldots \beta_m}{\gamma_1 | \ldots | \gamma_n}.$$

The γ_i are called the consequents. Here, let $c(G)$ be the set of *all* consequents of all defaults $d \in G$. The operator " | " is called *effective disjunction*.

W consists of no effective disjunctive information; all disjunctive information is represented in D. However, below we will sometimes use effective disjunctions in W for special kinds of theories. They could of course be moved into D as prerequisite-free and justification-free defaults.

A *propositional disjunctive default theory* $\langle D, W\rangle$ (from now on, ddt) is a finite set of propositional sentences W and a finite set of disjunctive defaults D.

As characterisation of *extensions* we use a similar one as in [Gelfond *et al.*, 1991], which are both equivalent to the standard definition given in [Gelfond *et al.*, 1991]. However, we use a base operator to derive the complexity results as we did for standard default theories in Chapter 5.1.

Definition 6.1.1 (closed) *A* disjunctive rule *is a rule of the form* $\frac{\alpha}{\gamma_1|\ldots|\gamma_n}$. *A finite theory E is* closed under a disjunctive rule *if, whenever $E \models \alpha$, then there exist i, $1 \leq i \leq n$ with $E \models \gamma_i$.*

It is sufficient to use the condition $\gamma_i \in E$ instead of $E \models \gamma_i$, since we only use base theories E (which are a set of consequents plus W).

Definition 6.1.2 (Reduct) *Let D be a disjunctive default theory and let $E = W \cup H$ with $H \subseteq c(D)$. The* reduct *of D with respect to E, denoted by D^E is the set of inference rules defined as follows: If the default*

$$\frac{\alpha : \beta_1, \ldots, \beta_m}{\gamma_1| \ldots |\gamma_n}$$

is in D and $E \not\models \neg\beta_i$ for all $\beta_1, \ldots, \beta_m$, then $\frac{\alpha}{\gamma_1|\ldots|\gamma_n}$ is in D^E.

Definition 6.1.3 (Base of extension) *A finite set of formulae E is the* base of an extension *for a ddt D if E is a minimal set (w.r.t. the subset-relation) closed under propositional consequence and under the rules of the reduct D^E. Every extension is of the form $cons(E)$.*

Here we defined for ease of argumentation E as finite set, and the extension as $cons(E)$ (different than in the previous chapter). Moreover, $c(D)$ is, in the setting of disjunctive default theories, the set of all consequents of all defaults.

Brave Reasoning with ddts has been proved to be Σ_2^P-complete and cautious reasoning to be Π_2^P-complete, both even for disjunction-free disjunctive default theories [Eiter and Gottlob, 1995] (i.e. W, the prerequisite, each justification, and each consequent γ_i of all defaults consist only of conjunctions of literals). In what follows, we will show that the complexity of model checking resides in Σ_2^P, too, and is $\mathbf{NP}$-complete for restricting to disjunction-free ddts at least in some cases.

One may define the concept of a normal default theory for disjunctive default theories in several ways; to this end, it is useful (only for representational issues, one can also view this as default) to allow effective disjunctions in W (for instance $W = \{a|b|c, d|c\}$), however, only in the sense that prerequisite-free justification-free defaults are moved into W.

Definition 6.1.4 *$\langle D, W\rangle$ is a (semi)normal limited disjunctive default theory iff only W contains effective disjunctions and each default in D is a standard (semi)normal default rule.*

Instead, one could allow special normal disjunctive rules:

Definition 6.1.5 $\langle D, W\rangle$ *is a normal default theory iff each default is of the form*

$$\frac{\alpha : \gamma_1 \vee \ldots \vee \gamma_n}{\gamma_1 | \ldots | \gamma_n}$$

and W *contains no effective disjunction.*

Definition 6.1.6 $\langle D, W\rangle$ *is a seminormal disjunctive default theory iff* W *contains no effective disjunctions and each default is of the form*

$$\frac{\alpha : \beta \wedge (\gamma_1 \vee \ldots \vee \gamma_n)}{\gamma_1 | \ldots | \gamma_n},$$

where β *is an arbitrary formula and* W *contains no effective disjunction.*

We allow disjunctions in the case of disjunction-free (sometimes shortened to df) ddts, too, but only in the justifications at the places where effective disjunctions occur in the consequent. However, for arbitrary df ddts we do not allow that disjunctions occur in the justifications!

As we will see, for limited df ddts, complexity results get one level easier. Limited theories hence are an interesting restriction to consider when, at the same time, dealing only with disjunction-free theories. Not limited theories are not more complex because we allowed disjunction for normal and seminormal theories, this was just a method to get normal theories defined for ddts. They are more complex, because the complexity sources of prerequisites (or justifications) and effective disjunctions is combined. If not both is allowed *in one default*, then two "orthogonal" complexities are parallelized.

6.1.1 General Disjunctive Default Theories

Lemma 6.1.7 *Let* $\langle D, W\rangle$ *be a propositional ddt and* $\mathcal{M}$ *a propositional model in the language of* $\langle D, W\rangle$*. Then, the problem to decide whether* $\mathcal{M} \models \langle D, W\rangle$ *is in* Σ_2^P*.*

Proof. Initially, M guesses a subset H of $c(d)$ with $E = W \cup H$. M forms the reduct D^E in polynomial time with access to an $\mathbf{NP}$-oracle for the checks whether $E \not\models \neg\beta_i$. The next step is testing whether D^E is closed which involves again polynomially many inference checks and hence uses an $\mathbf{NP}$-oracle polynomially often. Finally, M has to look whether D^E is a minimal reduct, i.e. if there is no $E' \subset E$ which is closed under D^E. This involves analysing all subsets of E and hence can be done with one inverse oracle-call. If E is a minimal closed set then $cons(E)$ forms an extension. Finally we just need to check in polynomial time if $\mathcal{M} \models E$.

The proof consists of an initial guessing stage, and on the oracle M polynomially often used $\mathbf{NP}$-oracles ($\mathbf{NP}^{\mathbf{P}^{\mathbf{NP}}+\mathbf{P}^{\mathbf{NP}}+\mathbf{coNP}}$, i.e. $\mathbf{NP}^{\mathbf{NP}}$), which is the class Σ_2^P. □

If we limit ourselves to disjunction-free ddts the initial oracle call remains, and all higher calls except the inverse call for checking if a smaller set exists, remain, therefore this problem is still in Σ_2^P. Therefore reasoning with disjunction-free disjunctive default theories is Σ_2^P-complete, too.

However, w.r.t. model checking, one can eliminate this higher call sometimes – as we will see below; consider prerequisite-free theories: reducts are in that case a number of rules, where each rule consists of a number of literal-conjunctions separated with the effective disjunction operator. Hence, minimality of a base theory closed under a reduct can be checked by only analyzing the subsets of the theory with one element less. This does not work in theories with prerequisites, however, as described below.

Moreover, we will consider a generalization of weak model checking to disjunctive theories and will recognize that in this generalization no minimality check is required, but only some check of "inverse closure" and hence, we now have the different result, that sometimes for the extended concept of disjunctive default theories weak model checking gets easier, namely in the case where prerequisites occur in a disjunction-free default theory.

We will present an overview of results in Chapter 8, and compare the results for disjunctive default logic, stationary default logic and standard default logic.

Another interesting detail is that model checking resides on the second level of the PH even for (limited) normal default theories, and even for prerequisite-free theories (see the results of the previous chapter which had favourable complexity results as soon as the theory was normal).

Lemma 6.1.8 *Let $\langle D, W\rangle$ be a finite propositional ddt and $\mathcal{M}$ a propositional interpretation in the language of $\langle D, W\rangle$. Then, the problem to decide whether $\mathcal{M} \models \langle D, W\rangle$ is Σ_2^P-hard, even if $\langle D, W\rangle$ is (limited) normal and prerequisite-free.*

Proof. We use a reduction from the Σ_2^P-complete problem $\mathrm{QBF}_{2,\exists}$, this is the problem if an evaluation of the $p_i \in P$ exists such that for each evaluation of the $q_j \in Q$ $F(P, Q)$ is true (abbreviated as: $\exists P \forall Q\, F(P, Q)$ with second order quantification).

We modify the default theory of the proof of Theorem 2 in [Liberatore and Schaerf, 1998] for standard semi-normal prerequisite-free default theories to use it in our proof. With the power of the effective disjunction the defaults $: w \wedge (w \Rightarrow p_i)/(w \Rightarrow p_i)$ and $: w \wedge (w \Rightarrow \neg p_i)/(w \Rightarrow \neg p_i)$ can now be displayed as $(w \Rightarrow p_i)|(w \Rightarrow \neg p_i)$. The w in these defaults was required in standard default theory because it blocks the default yielding $w \Rightarrow \neg p_i$ if $w \Rightarrow p_i$ has be inferred and vice versa (without w, both can be inferred, and $\neg w$ is inferred) which is not needed here due to minimality constraints.

The first default in the proof of that Theorem in [Liberatore and Schaerf, 1998] can be normalized anyway, from $: w \wedge \neg F/w$ to $: (w \wedge \neg F)/(w \wedge \neg F)$

Overall, the default theory we use is in the case of limited normal prerequisite-free ddts:

$$W = \bigcup_i \{(w \Rightarrow p_i)|(w \Rightarrow \neg p_i)\}\,,\ D = \left\{\frac{: w \wedge \neg F}{w \wedge \neg F}\right\}$$

W.r.t. to normal prerequisite-free disjunctive default theories:

$$W = \emptyset,\ D = \bigcup_i \left\{\frac{: (w \Rightarrow p_i) \vee (w \Rightarrow \neg p_i)}{w \Rightarrow p_i | w \Rightarrow \neg p_i}\right\} \cup \left\{\frac{: w \wedge \neg F}{w \wedge \neg F}\right\}$$

The transformation from $\text{QBF}_{2,\exists}$ is polynomial and the proof for both cases (limited normal and normal) the same, because in each theory exactly one consequent of the effective rules is inferred since these rules do not interrelate with each other.

$\mathcal{M} = \emptyset$ is a model of $\langle D, W\rangle$ iff $\exists P \forall Q\, F(P, Q)$.

$\mathcal{M}$ is a model iff an extension without w exists; this is the case iff $\neg F$ is inconsistent, i.e. F is valid under an evaluation of the p_i.

Say, we are given the context

$$\bigcup_i \{(w \Rightarrow p_i) \mid p_i \in P\} \cup \bigcup_i \{(w \Rightarrow \neg p_i) \mid \neg p_i \in P\}$$

with P is an evaluation making F valid (i.e. true for each evaluation of the q_j). This context allows to use the defaults yielding $w \Rightarrow \neg p_i$ or $w \Rightarrow p_i$, respectively. Moreover, it disallows using the default yielding $\neg F \wedge w$, since $\neg F \wedge w$ is inconsistent (if w is assumed, then the p_i and $\neg p_i$ are inferred due to the above rules and F is valid, hence $\neg F \wedge w$ is inconsistent). Therefore, context and inferred formulae coincide, and the context forms an extension and $\mathcal{M}$ is a model of it (since $\neg w$ is $true$ in the model, the p_i can have any value in the model).

If we choose any other context, we need to include w to get an extension, since the default yielding $\neg F \wedge w$ has to be used and $\mathcal{M}$ is not a model because w is $true$ in this extension. □

With the previous two lemmata we get:

Theorem 6.1.9 *Let $\langle D, W\rangle$ be a propositional ddt and $\mathcal{M}$ an interpretation in the language of $\langle D, W\rangle$. Then, the problem to decide whether $\mathcal{M} \models \langle D, W\rangle$ is Σ_2^P-complete, even if $\langle D, W\rangle$ is a (limited) normal and prerequisite-free default theory.*

6.1.2 Disjunction-free and EHNF theories

Before we consider disjunction-free disjunctive default theories (abbreviated as df ddt sometimes) we analyse for reasons of clarity where the additional complexity comes from.

Recall that in disjunctive logic programming (Section 2.5.7) model checking is under several semantics **coNP**-complete due to some kind of minimality requirement. Here, the case is similar, and this additional complexity is a minimality requirement which is "orthogonal" to the usual complexity from guessing a suitable subset. Recall the concept of answer sets in extended disjunctive logic programming as given in Subsection 4.8.5. Answer set checking is **coNP**-complete, too, and (cautious) reasoning Π_2^P-complete as for disjunctive logic programs.

Since EDLPs are closely related to disjunctive default logic, the complexity of disjunction-free default *reasoning* can be directly derived from the results in [Eiter and Gottlob, 1995]. For details on this topic, cf. to Section 5.8.

Due to this reason, we sometimes write a disjunctive default of the form

$$\frac{a_1 \wedge \ldots \wedge a_k : b_1, \ldots, b_m}{c_1 | \ldots | c_n}.$$

where all letters are single literals as

$$c_1 | \ldots | c_n \leftarrow a_1 \wedge \ldots \wedge a_k \wedge \text{not } \neg b_1 \wedge \ldots \wedge \text{not } \neg b_m$$

However, *model checking* for df ddts is a different concept than *answer set checking* in EDLPs. Answer sets are not total sets, they are sets with positive and negative literals, and leaving some literals undefined. Keep in mind that an answer set coincides with an extension in the notion of df ddts. Hence, the complexity of answer set checking coincides with the complexity of *extension checking* in df ddts (or to be more precise, in df ddts where each consequent is only a single literal, each prerequisite is a conjunction of literals, and each justification is a literal).

For results concerning extension checking we refer to Rosati [1999] (which is called model checking there). We are interested in model checking in the usual sense, with total models. This concept can also be used for EDLPs in addition to answer set checking, however it is a more difficult task (one level higher).

In the proof of model checking for DLP, the following transformation can be used (like the one presented in Section 4.8.6):

Let F be a formula in 3-CNF of the form $f \wedge (\neg x_1 \vee \cdots \vee \neg x_n)$ with $f_i = y_{i1} \wedge y_{i2} \wedge y_{i3} = \neg A_i \wedge B_i$ (with $1 \leq i \leq m$) where A is the conjunction of the atoms occurring as negated literals and B the disjunction of the positive ones, and y_{il} is a literal x_k or $\neg x_k$. Then

$$P = \quad x_j \vee B_i \leftarrow A_i.$$

These are $n \cdot m$ program clauses. It is easy to see that the full interpretation is a model exactly if F is unsatisfiable - the following statements are equivalent:

1. The full interpretation is a model
2. The full interpretation is the only model, because it is the largest one (minimality)
3. $\bigwedge_{i,j}(f_i \vee x_j) = (f_1 \wedge \cdots \wedge f_m) \vee (x_1 \wedge \cdots \wedge x_n)$ has the full interpretation as only model
4. $f = (f_1 \wedge \cdots \wedge f_m)$ is not satisfiable or has the full interpretation as only model
5. F is not satisfiable.

Therefore, this problem is $\mathbf{coNP}$-hard (and complete).

For extended disjunctive programs the proof can be used in a similar manner. It is however, important, not to treat the disjunctions $y_{i1} \vee y_{i2} \vee y_{i3}$ as $y_{i1}|y_{i2}|y_{i3}$, because then we loose the minimality check somehow – hence we cannot get the prerequisites out of the proof.

We want special answer sets containing only positive x_j. So, for EDLP, the program looks like:

$$x_j|B_i \leftarrow A_i$$

(the disjunctions in B_i are now effective disjunctions).

However, with this construction the full interpretation is always a model, since every answer set is a subset of $\{x_1, \ldots, x_n\}$ (N.B.: Answer sets correspond to the extensions, and to be a model of an answer set, the usual definition that $\mathcal{M}$ satisfies the answer set is used). Hence, two further clauses are needed:

$$w \leftarrow x_1 \wedge \cdots \wedge x_n$$

$$\neg w \leftarrow \text{ not } w$$

Therefore, only if the full set of all x_j is an answer set, then w is in the answer set, else $\neg w$. Therefore, the full interpretation (now including w) is a model iff F is not satisfiable, because otherwise every extension contains $\neg w$.

Hence, model checking is $\mathbf{coNP}$-hard for answer set semantics, and due to the transformation, clearly for df ddts, too.

However, it is even harder – we now have used the minimality criterion, additionally we implement the "default choice" – in such a way that there are exponentially possible generating sets. The details are given in the proof below.

Theorem 6.1.10 *Let $\langle D, W\rangle$ be a propositional disjunction-free ddt and $\mathcal{M}$ a propositional interpretation in the language of $\langle D, W\rangle$. Then, the problem to decide whether $\mathcal{M} \models \langle D, W\rangle$ is Σ_2^P-complete, even if $\langle D, W\rangle$ is normal.*

Proof. Consider the following df ddt (written in the syntax of EDLP to show that the same is possible for EDLPs):

We transform the formula $\exists p_k \neg \exists x_j F$ (with the same F as above, but now not only using the literals x_j, but additionally p_k) into a df ddt with a polynomial transformation:

A sample clause of F, $p_k \vee \neg p_i \vee x_j \vee \neg x_l$ is translated into n logic program clauses (where q_k shall represent "a negated p_k"; $1 \leq h \leq n$):

$$x_h|x_j \leftarrow p_i \wedge q_k \wedge x_l$$

In df ddts more literals are allowed to occur within each effective disjunction, hence these h rules can in an equivalent way be displayed as:

$$\frac{p_i \wedge q_k \wedge x_l : \mathbf{t}}{x_1 \wedge \ldots \wedge x_n | x_j}.$$

Our method to display a clause is just a sample for easier understanding, of course not every clause consists of one positive p, one negative p, one positive x and one negative one, but can be only three literals per clause which can be three positive p's, etc. - however, this sample clause has all those features needed and since in the body of EDLPs the number of conjunctions is not limited it makes no difference.

Above clause reflects the minimality check as explained above the proof; two more clauses are needed (as above), and a selection if p_i or $\neg p_i$ for each i where q_i shall represent the negation of p_i.

$$w \leftarrow x_1 \wedge \cdots \wedge x_n$$

$$\neg w \leftarrow \text{ not } w$$

$$p_i | q_i$$

These rules can be displayed as normal default rules, too:

$$\left\{ \frac{p_i \wedge q_k \wedge x_l : (x_1 \wedge \ldots \wedge x_n) \vee x_j}{x_1 \wedge \ldots \wedge x_n | x_j}, \frac{x_1 \wedge \cdots \wedge x_n : w}{w}, \frac{: \neg w}{\neg w}, \frac{: p_i \vee q_i}{p_i | q_i} \right\}$$

Meanwhile, the reader may consider if these rules can be displayed as limited normal or at least as limited ddts – we will show below that this is not possible by giving a membership proof in **NP** instead.

Back to the proof, it is easily verified that this transformation is feasible in polynomial time. We show that the full interpretation is a model exactly if $\exists p_k \forall x_j \neg F$ holds. Since F is in CNF, $\neg F$ is in DNF (and the problem QBF$_{2,\exists}$ is already hard for Σ_2^P if F is in 3-DNF).

$\exists p_k \forall x_j \neg F$ is equivalent to $\exists p_k \neg \exists x_j F$, which is equivalent to that there is an evaluation of the p_k and q_k (remember, only one for each k can be in the extension, since there is only one rule deriving each p_k or q_k, hence the minimality requirement allows for just one) where $x_1 \wedge \ldots \wedge x_m$ is part of the answer set (because $\neg \exists x_j \; F$) which is equivalent to that w is in one of the answer sets (otherwise $\neg w$ is forced to be in, here $\neg w$ can be in, but this is not the answer set we want for our interpretation). Since in every other case $\neg w$ is in the answer set, $\mathcal{M} = \mathcal{M}_{full}$ is a model exactly if $\exists p_k \forall x_j \neg F$ holds.

Lemma 6.1.11 *If a ddt is prerequisite-free, then the minimality check is polynomial.*

Proof. It suffices to check all subsets of E which have one element less, i.e. with one consequent less. If E is closed and no subset with one element less is closed, then E is minimal:

This is because if $E' = E \backslash c(d)$ is not closed, then no subset of it is closed. A subset of E' has a reduct which is a superset of $R^{E'}$ (because of more valid justifications or at least the same number).

Hence, in a larger reduct a smaller base is certainly not closed either, since no prerequisites can be used to block the use of some consequents! □

Hence, as soon as a theory is without prerequisites the minimality check is polynomial. However, this advantage is only useful for disjunction-free theories. In the general case, model checking remains at the second level, because it is at the second level for standard default theories, too. So why is model checking with normal ddts without prerequisites not easier although minimality can be checked in an easier way?

Observe that the closure-checks are not checks for which we need an oracle if prerequisite-free (only then!)! It is sufficient to check that at least one consequent for each default appears in the guessed base – which can be done in linear time! Above we stated that we need oracles for closure-checks but since we let E be a set of consequents, they are only needed when prerequisites come into play: The check $E \vdash p(d)$ needs oracles: and of course creating the reduct needs oracle-calls in the general case since the justifications are checked.

To answer the question of the above paragraph: the complexity of model checking for a normal and prerequisite-free theory is not easier, because also in normal default theories an oracle needs to be called if justifications which are arbitrary formulae in the not disjunction-free case, can be inferred of a set of consequents. It could be the case that they can be inferred although it cannot be directly seen via an element of the base – i.e. some defaults where we do not have at least one consequent in the base, may fire, too (this means that the interrelation of the default rules if one can be derived from some others etc. force the complexity) Due to these checks (and of course, an exponential selection of possible minimal sets) this problem is Σ_2^P-complete in that case, too.

Theorem 6.1.12 *Let $\langle D, W\rangle$ be a propositional disjunction-free prerequisite-free ddt and $\mathcal{M}$ a propositional interpretation in the language of $\langle D, W\rangle$. Then, the problem to decide whether $\mathcal{M} \models \langle D, W\rangle$ is $\mathbf{NP}$-complete, even if $\langle D, W\rangle$ is semi-normal.*

Proof. Let us first show membership: Initially we guess a subset $H \subseteq c(D)$ and let $E = W \cup H$ (W contains no effective disjunctions). We form the reduct D^E in polynomial time (all the inference checks if $E \not\models \neg\beta_i$ are polynomial now).

The next step is testing if D^E is closed, which involves again polynomially many inference checks and hence is polynomial, too. Now, instead of calling an oracle for a minimality check, there is an easier option:

Checking if E is minimal can be done by analysing all those subsets of E having one consequent less than E, i.e. just looking at n subsets instead of 2^n.

This works because for prerequisite-free ddt's it holds that: If E is closed and no subset $E' = E \backslash c(d)$ is closed, then E is minimal. This fact can be easily proved with using the fact proved in Lemma 6.1.11 that if $E \backslash c(d)$ is not closed, then of course in a prerequisite-free theory, no subset of this set can be closed.

Finally, check if $\mathcal{M} \vdash E$.

Hardness: Follows immediately from Lemma 5.4.5 for the case of prerequisite-free standard disjunction-free default theories. □

Let us again think about whether a normal df ddt like the one we used in the proof of Theorem 6.1.10 can somehow be transformed into a limited (normal) df ddt. Well, the default

$$\frac{p_i \wedge q_k \wedge x_l : (x_1 \wedge \ldots \wedge x_n) \vee x_j}{x_1 \wedge \ldots \wedge x_n | x_j}$$

could be translated into $\frac{p_i \wedge q_k \wedge x_l : s_h}{s_h}$, $\frac{: \neg s_h}{\neg s_h}$, $\frac{: s_h}{x_k | x_1 \wedge \ldots \wedge x_n}$. However, this is not limited, too! The prerequisite of a default with effective disjunction was just moved into a justification – and we do not allow such defaults in limited ddts either. We will now show, that indeed, model checking with limited default theories, in case they are disjunction-free, is one level easier for theories with prerequisites.

Theorem 6.1.13 *Let $\langle D, W\rangle$ be a propositional disjunction-free limited ddt and $\mathcal{M}$ a propositional interpretation in the language of $\langle D, W\rangle$. Then, the problem to decide whether $\mathcal{M} \models \langle D, W\rangle$ is* $\mathbf{NP}$*-complete, even if $\langle D, W\rangle$ is semi-normal and prerequisite-free or normal.*

Proof. Membership: The idea for limited theories is that we have two different kinds of independent parts in dtt: namely firstly the part with the effective disjunctions but no prerequisites/justifications, and secondly the usual default theory part.

- In the case of normal default theories we distinguish two kinds of defaults – the one with effective disjunctions (no prerequisites, no justification) D_1, and the ones which are normal defaults D_2. Each consequent is a conjunction of literals. Now guess a generating set R of D_1 and check if it is closed and minimal – checking minimality is now polynomial, since no prerequisites occur! That is the reason why model checking gets easier – that no prerequisites occur in defaults with effective disjunction (and no justifications!). Then just do model checking as usual in normal standard df default theories with adding this generating set to W, i.e. do model checking for the theory $\langle D_2, W \cup R\rangle$.
- The proof is analogous for arbitrary df ddts: Now a reduct E is guessed: Then we check if E is minimal with respect to the literals occurring in D_1 (possible in polynomial time since D_1 is disjunction-free and no prerequisites in it) and if it is closed. If yes, then these literals are added to W, and with the guessed generating set for D_2 (the subset of E restricted on the literals occurring in D_2) check if $\mathcal{M}$ is a model, as usual for standard theories.

The hardness follows from Lemma 5.4.5: For the case of semi-normal prerequisite-free default theories, use the default theory of that proof. For the case of normal default theories, use the default theory which we presented immediately after the proof – this theory is normal except for the defaults

$$\bigcup_k \left\{ \frac{: u_k \wedge \neg t_k}{\neg t_k}, \frac{: t_k \wedge \neg u_k}{\neg u_k} \right\}$$

– recall the discussion at that point: Now these defaults can be normalized by replacing them for each k with $\neg u_k|\neg t_k$ (in W). □

Example 6.1.14 *$\mathcal{M} = \{a, b\}$, $E = \{a, b\}$, $D = \left\{a|b, \frac{:\neg a}{\neg a}, \frac{:\neg b}{\neg b}\right\}$. Then E is a closed set, but E is not minimal, since subsets of E are closed. It does not matter if E is a superset of an extension or, as in this case, is not: Here, no subset of E forms an extension, but $\{a, \neg b\}$ and $\{\neg a, b\}$ create the only extensions; that is the reason why we need the minimality check also for model checking – because not every closed set has a minimal closed subset!*

One might ask if a limited normal *and* prerequisite-free disjunction-free theory can be used in the above proof, too, as we did in the proof of Theorem 6.1.9. The first set of defaults is replaced as above with $\neg u_k|\neg t_k$ (in W) for each k. The defaults $(:\ f(x_{ij}) \wedge \neg y_i)/\neg y_i$ remain the only semi-normal defaults. Can they be eliminated and replaced by normal prerequisite-free defaults? Yes, they can be of the form $: \neg u_k \wedge t_k \wedge \neg y_i / \neg u_k \wedge t_k \wedge \neg y_i$, but each of these default has one unnegated literal in their consequents and intuition says that if we cannot construct such a theory without positive literals in these or similar consequents then it will be difficult to show that model checking is $\mathbf{NP}$-hard (these or similar defaults would only work to show that reasoning is $\mathbf{NP}$-hard). However, as the next proof shows, model checking with normal prerequisite-free disjunction-free ddts is $\mathbf{NP}$-complete, too, for both limited and not limited theories (remember, in the second case, defaults look like $\frac{:a \vee b}{a|b}$).

One might think about transforming a 3-CNF formula $F = \bigwedge_i (y_{i1} \vee y_{i2} \vee y_{i3})$ into a prerequisite-free df theory with defaults $f(y_{i1})|f(y_{i2})|f(y_{i3})$ with $f(y_{ij})$ is $\neg u_k$ if $y_{ij} = \neg u_k$ and $\neg t_k$ if $y_{ij} = u_k$ and a default like $\neg u_k|\neg t_k$. However, be careful! In the above theory this default does no harm, since $\neg u_k$ and $\neg t_k$ occur nowhere else as consequents, hence this theory was minimal where exactly one was chosen, however with these default rules, for one k both of $\neg u_k$ and $\neg t_k$ can be in an extension, for instance if F is inconsistent. Therefore, with such defaults, we again require semi-normal defaults like $: \neg u_k \wedge t_k / \neg u_k$ and $: \neg t_k \wedge u_k / \neg t_k$.

Such a theory has an extension exactly if F is satisfiable, hence the empty set is a model exactly in this case. Hence, for such theories the complexity of the effective choice and of the generating set are somehow no longer two different sources of complexity (compare this discussion with the previous proof which was a different hardness-proof for such theories) but parallelised.

However, it is wrong to assume that model checking is really easier for normal default theories. Minimality is such a strong requirement that even a fixed model and a normal prerequisite-free ddt, regardless if limited or not, requires a guessing step. The intuition behind this is that even if a reduct is constructed corresponding to a model, i.e. that first R' is formed containing all positive atoms of $\mathcal{M}$ and all negated atoms not in $\mathcal{M}$, and then R as the set of all consequents which can be deduced of R' is created. If R is closed and minimal, everything is fine - $\mathcal{M}$ is a model. If R is not closed, everything is fine, too. $\mathcal{M}$ is not a model. However, we are on the wrong side if R is closed but not minimal! Then each subset of R needs to be analysed to find the minimal closed ones if any – which is again an exponential task. Observe: The minimality check itself is polynomial, however if it fails we need another R and finding such a minimal R is exponential.

Extension checking would be polynomial, since with conjunction of literals in prerequisite-free theories, minimality is easy to check. Model Checking on the contrary requires taking a look at all extensions which $\mathcal{M}$ is a model of.

We chose to give seperate proofs for not normal theories first, and now provide the proof that minimality is actually such a strong requirement that even model checking for normal prerequisite-free theories is $\mathbf{NP}$-hard. Hence:

Theorem 6.1.15 *Let $\langle D, W\rangle$ be a normal prerequisite-free (limited or not) disjunction-free ddt and $\mathcal{M}$ a propositional interpretation. Then, the problem to decide whether $\mathcal{M} \models \langle D, W\rangle$ is $\mathbf{NP}$-complete, even if the theory is limited normal and each consequent is a conjunction of at most two literals.*

Proof. Let $F = f_1 \wedge \cdots \wedge f_n$ with $f_j = y_{j1} \vee y_{j2} \vee y_{j3}$ be a formula in 3-CNF made up of the literals x_i and $\neg x_i (1 \leq i \leq m)$.

Let P be the following limited disjunctive default theory:

$$W = \{\neg x_1 \wedge \neg z_1 | \neg x_1 \wedge \neg v_1 | \neg z_1 \wedge \neg v_1, \ldots,$$

$$\neg x_m \wedge \neg z_m | \neg x_m \wedge \neg v_m | \neg z_m \wedge \neg v_m,$$

$$f(y_{11}) | f(y_{12}) | f(y_{13}), \ldots, f(y_{n1}) | f(y_{n2}) | f(y_{n3})\},$$

$$D = \left\{ \frac{: v_1}{v_1}, \frac{: v_2}{v_2}, \ldots, \frac{: v_m}{v_m} \right\}$$

and let P' be the following disjunctive default theory:

$$D = \left\{ \frac{: (\neg x_1 \wedge \neg z_1) \vee (\neg x_1 \wedge \neg v_1) \vee (\neg z_1 \wedge \neg v_1)}{\neg x_1 \wedge \neg z_1 | \neg x_1 \wedge \neg v_1 | \neg z_1 \wedge \neg v_1}, \ldots, \right.$$

$$\frac{: (\neg x_m \wedge \neg z_m) \vee (\neg x_m \wedge \neg v_m) \vee (\neg z_m \wedge \neg v_m)}{\neg x_m \wedge \neg z_m | \neg x_m \wedge \neg v_m | \neg z_m \wedge \neg v_m},$$

$$\left. \frac{: f(y_{11}) \vee f(y_{12}) \vee f(y_{13})}{f(y_{11}) | f(y_{12}) | f(y_{13})}, \ldots, \frac{: f(y_{n1}) \vee f(y_{n2}) \vee f(y_{n3})}{f(y_{n1}) | f(y_{n2}) | f(y_{n3})} \right\},$$

$$D = \left\{ \frac{: v_1}{v_1}, \frac{: v_2}{v_2}, \ldots, \frac{: v_m}{v_m} \right\}$$

with $f(y_{jk}) = \neg x_i$ iff $y_{jk} = \neg x_i$ and $f(y_{jk}) = \neg z_i$ iff $y_{jk} = x_i$.

This transformation is feasible in polynomial time, hence for both cases it remains to be shown that the empty set is a model exactly if F is satisfiable.

- f not satisfiable implies that the defaults depicting the translated formula force for at least one variable x_i that both $\neg z_i$ and $\neg x_i$ are in each minimal set (only a satisfiable F allows for an evaluation where in a minimal reduct only occurs at most one of $\neg z_i$ and $\neg x_i$). Consider the first defaults: We have three choices, but choosing one of the second or third is for at least one x_i not minimal, since a $\neg v_i$ is introduced, but not needed because both of $\neg x_i$ and $\neg y_i$ for at least one i, are in each minimal set. Therefore, at least one of the defaults yielding v_i needs to be used in each extension (maybe different v_i in different extensions) and clearly the empty set cannot be a model of such extensions with positive literals.

- The empty set is not a model implies that no minimal reduct may contain all $\neg v_i$. Hence, for at least one i it must be forced that adding $\neg v_i$ would be non-minimal. This is only possible if for one i $\neg x_i \wedge \neg z_i$ are forced to be in a minimal reduct, which is only possible if F is not satisfiable, otherwise we would find a suitable extension for the empty model.

Hence, $\mathbf{NP}$-completeness is proved. □

It is clear that the proof holds for both cases, for P and P'. It holds in the second case, too, because it makes no difference to the first case – if we used y_{jk} instead of $f(y_{jk})$ then it would have made a difference, since then not all defaults at the same time could have fired if F is not satisfiable. However, we have only negated literals in these formulae, so all are consistent with each other. And that is exactly what we need, because then it is straightforward to implement the minimality criterion.

Limited default theories exhibited easier complexity results w.r.t. model checking than general ones if disjunction-free. However, they also suffer a disadvantage that not every limited normal default theory admits an extension, whereas a not limited normal default theory always has an extension.

Proposition 6.1.16 *If W is consistent, a normal ddt always has at least one consistent extension. However, a limited normal theory suffers the disadvantage of not always having a consistent extension.*

Proof. Use this constructive way to get an arbitrary extension: Choose a random order of defaults. Choose the first default with applicable prerequisites due to W. If the justification is consistent with W, then add this consequent to the constructed base of an extension, else choose the next one.

Then proceed and consider the next default with applicable prerequisites due to W and already added consequents. If the justification is consistent with $W \wedge c(F)$ where F is the current set of used consequents, then add the first consequent (arbitrary order of effective disjunctions) which can be inferred of $W \wedge c(F)$. If none can be inferred, then add the first one that cannot be inferred. This check assures minimality.

The algorithm will check all defaults for n times, with n is the number of defaults. After this a minimal (because no subset is closed due to the above criteria that first already inferred information is added) and closed (all applicable defaults used) subset of consequents is formed, which is the base of an extension. This routine works only for normal default theories, since justifications coincide with consequents.

However, now consider limited normal default theories. Consider the theory

$$y_{11}|y_{12}|y_{13}$$
$$\dots$$
$$y_{i1}|y_{i2}|y_{i3}$$
$$\dots$$
$$y_{n1}|y_{n2}|y_{n3}$$

and a formula $F = (y_{11} \vee y_{12} \vee y_{13}) \wedge \dots (y_{i1} \vee y_{i2} \vee y_{i3}) \wedge \dots (y_{n1} \vee y_{n2} \vee y_{n3})$. This theory has a consistent extension exactly if F is satisfiable. $\square$

Hence, instead of dealing with limited default theories, another kind of restricted disjunction-free default theories could be introduced, which have no prerequisites if an effective disjunction appears. In the case of normal default theories this kind of limited default theories exhibit the same complexity results as limited ones ($\mathbf{NP}$-complete; justifications equal the justifications in normal ddts). However, if allowing arbitrary justifications in defaults with effective disjunction, then model checking is Σ_2^P-complete – even if these defaults are prerequisite-free, as arbitrary justifications also contribute to the complexity.

Hence, if another notion of limited default theories is used it should be the notion that each default containing effective disjunctions is required to be prerequisite-free and normal (whereas the standard defaults of the theory are not required to be normal). However, this is a rather artificial notion, not like the one without justifications and prerequisites which can be added to W which we called "limited".

We only considered disjunction-free default theories in this section, and now generalise the results to EHNF theories. Hardness clearly is valid, and membership is straightforward to show: Recall Lemma 6.1.11 – the minimality check is polynomial in the case of prerequisite-free theories – hence, since for EHNF no SAT-oracle is needed for inference and satisfiability checks, the problem is in $\mathbf{NP}$, too.

The same holds for limited EHNF default theories: Since limited theories have no prerequisites in defaults with effective disjunctions, and no justifications, too, just guess a set of consequents E, check if $c(E_1)$ with $E_1 \subseteq D_1$ (defined as in the proof of Theorem 6.1.13) is minimal (in $\mathbf{P}$) and use this as W for $c(E_2)$ ($E_2 \subseteq D_2$) and check whether this set is closed and minimal, which is polynomial as well.

6.1.3 Weak Model Checking

What happens with weak model checking? How can we generalise this setting to disjunctive default theories? And in particular, how to define a weak extension of a disjunctive default theory?

We have used the definition of [Gelfond *et al.*, 1991] for the concept of extensions which is equivalent to the usual one for default theories if restricted to no occurrence of effective disjunctions.

The minimality requirement of default theories is somehow mixed in this definition with the minimality requirement of the effective disjunction. Weak extensions however, for standard default theories, do not directly force such a minimality requirement as weak extensions may contain each other, anyway, they force some form of minimality that no unfounded defaults shall be used.

We have seen that minimality forces for a higher complexity class in case of restricted ddts (both for reasoning as shown in [Eiter and Gottlob, 1995] and for model checking as shown here). So maybe for weak extensions this problem does not occur – depending on a suitable generalization of weak extensions to ddts.

Well, in weak extensions prerequisites need not be applied in a constructive way, allowing for weak extensions to contain other weak extensions. One way of generalizing to ddts is hence to give up the minimality requirement – but this means that, for instance, the theory $a|b$ has extensions $\{a\}$, $\{b\}$ and $\{a, b\}$. Therefore, this generalization gives up the fact that weak extensions equal extensions in the prerequisite-free case after effective disjunctions are introduced.

However, since one property of weak extensions is that they can contain each other and are hence somehow not based on minimality, this is no real disadvantage and one possible useful generalization.

We need to give a definition of weak extensions for disjunctive default theories and prove that it equals the usual weak extension definition in the case of standard default theories:

Definition 6.1.17 *A weak extension is a finite set of formulae $E \subseteq W \cup c(G)$ which is*

- *closed*
- *inverse closed: i.e. for each rule $\frac{\alpha}{\gamma_1|\ldots|\gamma_n}$, if exists a $\gamma_i \in E$, then $\alpha \in E$.*
- *each element of E (i.e. each $c(d)$) must appear in the head of a rule where $E \models \alpha$.*

For a complexity theoretic view it is important to choose only E which are of the form $W \cup c(G)$ (for condition 3) – this is no restriction, however, since each extension has such an E as base!

Lemma 6.1.18 *The above definition matches the usual characterisation of weak extensions if restricted to standard default theories.*

Proof. $\Leftarrow$: The usual definition is $E = \{c(d) \mid E \models p(d)\}$; i.e. all consequents are added where $p(d)$ can be inferred of E, this implies that the prerequisite can be inferred of E (E is a set of consequents here) exactly if the corresponding consequent is in E: $E \models p(d) \Leftrightarrow c(d) \in E$. This is property (1) and (2). Additionally each element which occurs in a given E must occur as a consequent (as one consequent in the head of the rule, however here we only have one consequent per default), otherwise E would be larger than the set on the right-hand side, hence condition (3) is also fulfilled.

$\Rightarrow$: Property (1) and (2) imply that $E \models p(d) \Leftrightarrow c(d) \in E$ which implies that $\{c(d) \mid E \models p(d)\} \subseteq E$. Property (3) implies that each element of E appears as consequent of some rule where the prerequisite is satisfied by F, hence $\{c(d) \mid c(d) \in E\} \supseteq \{c(d) \mid E \models p(d)\}$, so overall: $\{c(d) \mid E \models p(d)\} = E$ which is the usual definition of weak extensions. $\square$

Property (3) is needed because else E can still contain unfounded information (cf. to the next example) and be a superset of $\{c(d) \mid E \models p(d)\}$.

Example 6.1.19 $D = \{\frac{:\neg b}{b}, \frac{:a}{a}\}$. *Then* $E = \{a, b\}$ *has the reduct* $R = \; a \leftarrow$ *and is closed and inverse closed. However, the theory has no weak extension, hence this base theory* $\{a, b\}$ *cannot be one. Therefore the third requirement is necessary: Each element of* E *must appear in the head of some rule where* $\alpha \in E$*. Testing all candidates for weak extensions for this dtt always violates one of the three conditions, proving that this ddt has no weak extension.*

Given a suitable definition, we are now able to analyse the complexity of weak model checking for the same cases as for model checking. Let us first state that checking if a base E is an extension can be done in $\mathbf{P}$ in the case of disjunction-free theories, since no oracle calls are needed.

Theorem 6.1.20 *Let* $\langle D, W\rangle$ *be a ddt and* $\mathcal{M}$ *a propositional interpretation in the language of the default theory. Then the problem to decide whether* $\mathcal{M} \models \langle D, W\rangle$ *is* Σ_2^P*-complete, even if the theory is normal or limited normal, or semi-normal/limited semi-normal and prerequisite-free.*

Proof. Membership for arbitrary theories can be shown in a similar way as in Lemma 6.1.7 for model checking (and requires one $\mathbf{coNP}$-check less on the oracle). Hardness cannot be shown as in Lemma 6.1.8 for extensions (now both heads of $w \Rightarrow p_i | w \Rightarrow \neg p_i$ can fire in a weak extension, and yield $\neg w$ and hence cannot be used) and indeed is easier in the normal prerequisite-free case! However, hardness holds for the case of normal theories with prerequisites (use the proof of Theorem 5.5.2) and the case of semi-normal theories without prerequisites (due to the results of [Liberatore and Schaerf, 1998] and the fact that for standard default theories extensions and weak extensions coincide in the prerequisite-free case). $\square$

Theorem 6.1.21 *Let* $\langle D, W\rangle$ *be a ddt and* $\mathcal{M}$ *a propositional interpretation in the language of the default theory. Then, the problem to decide whether* $\mathcal{M} \models \langle D, W\rangle$ *is* $\mathbf{coNP}$*-complete if the theory is normal/limited normal and prerequisite-free.*

Proof. Hardness follows immediately of Theorem 5.5.3 for weak model checking with normal standard default theories.

Membership is more involved to show: Split the sets of defaults into a good and bad set. The good set contains all defaults where at least one consequent is satisfied by the given model (these consequents are called good consequents). Based on the set of good defaults, it is possible to construct a base E and a reduct for the base E, which is, first of all, the largest reduct for the set G of good defaults containing all other reducts w.r.t. G, and furthermore (as no prerequisites are involved) automatically is inverse closed and due to the construction fulfills property 3. It remains to show that the reduct is closed – if this reduct is not closed, then no subset of the base E forms a base either (no prerequisites). Note that we need no minimality here!

To show that this reduct is closed, show that none of the consequents appearing in the effective disjunctions of the bad defaults fires. Hence, this check is equal to the check whether $E \wedge j(d)$ is consistent for at least one $d \in B$. This can be checked with a $\mathbf{coNP}$-call as in the proof of Corollary 5.3.4 (inverse guess of model).

The proof is similar for limited normal theories: However, if there is at least one bad default which looks like $a|b|c$, then it fires, regardless what happens and $\mathcal{M}$ is not a model (also observe that such a theory has not always an extension, anyway this does not change complexity). And the standard defaults : a/a need to be checked as before applying a $\mathbf{coNP}$-call. □

Theorem 6.1.22 *Let $\langle D, W\rangle$ be a disjunction-free ddt and $\mathcal{M}$ a propositional interpretation in the language of the default theory. Then, the problem to decide whether $\mathcal{M} \models \langle D, W\rangle$ is $\mathbf{NP}$-complete if the theory is semi-normal/limited semi-normal and prerequisite-free, or normal/limited normal.*

Proof. Hardness is straightforward as the problem is already hard for standard weak default theories in the respective cases (and for instance, semi-normal default theories are subsets of semi-normal but also of semi-normal limited disjunctive default theories).

Membership: Guess a set $E \subseteq W \cup c(G)$. Show that the set is closed (polynomial time, no oracle calls necessary since just conjunctions of literals are involved), is inverse closed (polynomial due to the same reasons) and satisfies property 3 (also polynomial time). Finally, check if $\mathcal{M}$ satisfies E. □

Theorem 6.1.23 *Let $\langle D, W\rangle$ be an EHNF (or disjunction-free) normal/ limited normal prerequisite-free ddt and $\mathcal{M}$ a propositional interpretation in the language of the default theory. Then, the problem to decide whether $\mathcal{M} \models \langle D, W\rangle$ is $\mathbf{P}$-complete.*

Proof. Membership: As in proof of Theorem 6.1.21, split into bad and good defaults w.r.t. a given model. Form E as set of all good consequents (the consequents which are satisfied by the given model) of good defaults (not only one of each, but all, since there is no minimality requirement; if it is valid for a subset, it is valid for the set, too). Properties 2 and 3 are automatically fulfilled (2 because prerequisite-free, and 3 due to the reduct construction); checking Property 1 can now be accomplished in polynomial time since only conjunctions of literals are involved. This holds equally for limited theories: if a bad disjunctive default like $a|b|c$ exists, then the model does not satisfy the extension. Finally check if $\mathcal{M} \models E$.

Hardness: The problem is not $\mathbf{P}$-hard, but in a similar way to the problem in Theorem 5.4.10 it can be proved to be in $\mathbf{L}$. However, similar to above the problem is $\mathbf{P}$-hard in case of EHNF default theories. □

Consider the three checks: closed, inverse closed and each element of the reduct is an element of a used rule. These checks are also polynomial for EHNF theories. Hence, EHNF is of the same complexity as disjunction-free w.r.t. weak model checking.

Example 6.1.24 *Let* $D = \left\{\frac{:a}{a}, \frac{:b \vee d}{b|d}, \frac{:\neg c}{\neg c}\right\}$ *and* $\mathcal{M} = \{a, b, c, \neg d\}$*. Then,* $E = \{a, b\}$*, and* $R^E = \{a \leftarrow, b|d \leftarrow, \neg c \leftarrow\}$*, as* $\neg c$ *is also consistent with* E*.* E *is automatically inverse closed and every member of* E *automatically appears as a used consequent (since furthermore prerequisite-free), so it just remains to check if* E *is closed: Here,* E *is not closed since* $\rightarrow \neg c$ *is not closed. Hence,* $\mathcal{M}$ *is not a model of this default theory.*

6.2 Some words on Reasoning for Disjunctive Default Logic

Σ_2^P-hardness (completeness) of brave reasoning and Π_2^P-hardness (completeness) of cautious reasoning was shown for disjunction-free ddts in [Eiter and Gottlob, 1995]. Σ_2^P-hardness even holds for theories which only have single literals as consequences, no occurrence of $\neg$ in the theory and only one single literal justification for each default, and even if the formula to be inferred is a single literal. These facts are not all explicitly stated in the cited paper, however, they follow in a straightforward way.

In brief, we consider the remaining cases: prerequisite-free disjunction-free ddt's, normal disjunction-free ddts and limited disjunction-free ddts.

Prerequisite-free df ddts. $\mathbf{NP}$-complete: Guess a generating set, check if closed and minimal. The minimality check can be done in polynomial time in prerequisite-free and df ddts. Hence, after constructing a generating set it is checked if the given formula f can be inferred. This can be done without oracle calls in the case f is also limited to be disjunction-free (which we implicitly assume if we limit the default theory to be df). $\mathbf{NP}$-hardness can easily be shown using a somewhat simpler proof than in model checking.

Normal df ddts. Σ_2^P-complete: Slightly modify the proof of Theorem 6.1.10 and let $f = w$; the question to determine whether f is in at least one extension is Σ_2^P-complete.

Limited df ddts. $\mathbf{NP}$-complete: Limited theories have no prerequisites and no justifications for the defaults with effective disjunctions, hence minimality of this part can be checked in polynomial time after guessing a set of consequences. For the other non-effective defaults proceed as usual. Hence, the problem is in $\mathbf{NP}$. Moreover, it is straightforward to prove $\mathbf{NP}$-hardness.

These results are related to the fact that in the case of disjunctive default theories, the advantage that in some special cases model checking is easier than reasoning is lost.

6.3 Model Checking with Stationary Default Theories

In this chapter we state some results for model checking for stationary default theories. For details of stationary default logic, we refer the reader to [Przymusinska and Przymusinski, 1995] and to Section 2.5.4. The complexity of brave reasoning was proved to be Σ_2^P-complete, however, the complexity of cautious reasoning is on the first level of the polynomial hierarchy, and Δ_2^P-complete due to the existence of a smallest extension as intersection of all extensions [Gottlob, 1995]. In the case of reasoning for normal default theories, Gottlob [1995] showed that only a logarithmic number of oracle calls is needed, hence the problem of cautious reasoning with normal default theories is only $\Delta_2^P[O(\log n)]$-complete.

The fact that each normal default theory has at least one extension was not directly important for the proof in Section 5.3, since we started with a given interpretation and wanted to find an extension with $\mathcal{M} \models E$. It does not help to know that there always exists an extension of a normal default theory since this does not state anything about if $\mathcal{M}$ is a model of it.

However, in some variants of default logic, like default logic with stationary extensions, there exists a minimal extension as subset of all other extensions, therefore, if we are given an interpretation and want to check if it is a model for this theory, it suffices to look if it is a model of the minimal extension.

Given Gottlob's finite characterisation of default logic under the stationary semantics [Gottlob, 1995] the following proofs are straightforward. We call $\mathcal{M}$ a stationary model of $\langle D, W\rangle$ iff $\mathcal{M}$ satisfies at least one extension of $\langle D, W\rangle$ under the stationary fixed-point semantics.

Theorem 6.3.1 *Let $\langle D, W\rangle$ be a default theory and $\mathcal{M}$ an interpretation in the language of $\langle D, W\rangle$. Then, the problem to decide whether $\mathcal{M}$ is a stationary model of $\langle D, W\rangle$ is Δ_2^P-complete, even if $\langle D, W\rangle$ is prerequisite-free and semi-normal.*

Proof. Membership: Modify the membership proof of Theorem 3.1 in [Gottlob, 1995]: Instead of the final step to check if $\gamma_\Delta^{2d} \models \psi$ check if $\mathcal{M} \models \gamma_\Delta^{2d}$. Still, a polynomial number of oracle-calls remain, hence the problem is in Δ_2^P.

Hardness: Modify the hardness proof of Theorem 3.1 in [Gottlob, 1995]: Add the default $\frac{:\neg q_n}{\neg q_n}$ to that default theory. $\mathcal{M} = \{q_1, \ldots, q_n\}$ is a model of $\langle D, W\rangle$ exactly if there exists an extension without $\neg q_n$. The smallest extension either contains $\neg q_n$ (if $v(p_n) = 0$) or neither q_n nor $\neg q_n$ (if $v(p_n) = 1$, in this case there are extensions containing q_n and $\neg q_n$ and the smallest one is the empty set). $\mathcal{M}$ is only a model in the second case. Therefore, determining if $\mathcal{M} \models \langle D, W\rangle$ is Δ_2^P-hard. $\Box$

Theorem 6.3.2 *Let $\langle D, W\rangle$ be a normal default theory and $\mathcal{M}$ an interpretation in the language of $\langle D, W\rangle$. Then the problem to decide whether $\mathcal{M}$ is a stationary model of $\langle D, W\rangle$ is $\Delta_2^P[O(\log n)]$-complete.*

Proof. (Sketch) Membership is shown with similar methods as in the proof of Theorem 4.2 in [Gottlob, 1995] and Theorem 5.3.1 of this report.

Hardness is shown with the default theory of Theorem 5.3.3: There exists a unique stationary extension, namely either W or W plus one default. Hence, this is the smallest extension and $\mathcal{M}$ is a model iff an odd number of formulae is satisfiable. $\square$

Theorem 6.3.3 *Let $\langle D, W\rangle$ be a normal prerequisite-free default theory and $\mathcal{M}$ an interpretation in the language of $\langle D, W\rangle$. Then, the problem to decide whether $\mathcal{M}$ is a stationary model of $\langle D, W\rangle$ is $\mathbf{coNP}$-complete.*

Proof. Membership: First machine M checks if $W \wedge c(D)$ is inconsistent ($\mathbf{coNP}$-call). If the answer is *yes*, then $\mathcal{M}$ is a model regardless which model is given, since then the empty set is the smallest extension.

If it is consistent, two cases are distinguished depending if $\mathcal{M} \models W \wedge c(G)$, which can be checked in polynomial time. If yes, then $\mathcal{M}$ is a model, if no, then $\mathcal{M}$ is not a model (since then a bad default fires which contradicts the model but not the other defaults).

Hardness: Show that $\mathcal{M} = \emptyset$ is a model iff F is not satisfiable. Let $F = W \wedge c(D')$ and $W = \emptyset$, $D = D' \cup \left\{\frac{:z}{z}\right\}$ with D' an arbitrary set of normal prerequisite-free defaults, and z a new variable. This transformation is clearly polynomial, so it remains to be shown that the empty set is a model exactly if F is not satisfiable.

- $\emptyset$ is a model implies that either $W \wedge c(D)$ is inconsistent, which implies that $W \wedge c(D') \wedge z$ and $W \wedge c(D')$ is inconsistent; or that $W \wedge c(D)$ is consistent with $\emptyset \models W \wedge c(D)$. But since $c(D) = c(D') \wedge z, \emptyset$ would be no model in this case (this is why we constructed the theory that way!). Hence, it follows that $W \wedge c(D')$ is inconsistent, implying that F is inconsistent.
- If F is inconsistent then $W \wedge c(D')$ is inconsistent, and therefore $W \wedge c(D)$, too; therefore, $\emptyset$ is a model since the smallest extension is $\emptyset$ in this case.

Theorem 6.3.4 *Let $\langle D, W\rangle$ be a disjunction-free or EHNF default theory and $\mathcal{M}$ a model in the language of $\langle D, W\rangle$. Then, the problem to decide whether $\mathcal{M}$ is a stationary model of $\langle D, W\rangle$ is $\mathbf{P}$-complete.*

Proof. (Sketch) Modify the membership proof of Theorem 3.1 in [Gottlob, 1995]: Instead of the final step $\gamma_\Delta^{2d} \models \psi$ check if $\mathcal{M} \models \gamma_\Delta^{2d}$. Computing γ_Δ^{2d} is polynomial, since all oracle calls turn into tractable problems. Hardness can be proved similar to Lemma 5.4.8 (and in case of EHNF theories, just W being a Horn theory suffices to prove hardness). $\square$

Chapter 7

Complexity of Fixed-point 3-valued Semantics for Autoepistemic Logic

Denecker, Marek and Truszczyński [Denecker *et al.*, 1998] introduced a three-valued semantics for autoepistemic logic (refer to Section 2.5.5) and gave an efficient algorithm computing a belief pair $\mathcal{B}$ which is in the class Δ_2^P. We show [Baumgartner and Gottlob, 1999c] that computing a belief pair is Δ_2^P-hard even if no nested modalities occur. Additionally we show that complexity slightly decreases in the case of only one single top level modality to $\Delta_2^P[O(\log n)]$-completeness.

Denecker *et al.* used a derivation operator and defined the semantics for AEL as its least fixed-point. This semantics is three-valued because for some formulae the least fixed-point does not specify whether they are believed or not.

Complete fixed-points of the derivation operator correspond to Moore's stable expansions. In the general case, determining whether an autoepistemic theory admits at least one expansion is a Σ_2^P-complete problem (section 4.5) which is a more complex problem than this three-valued proximity approach.

The three-valued semantics is based on the notion of a belief pair (P, S), where P and S are sets of two-valued interpretations such that $S \subseteq P$, and is a generalization of Moore's characterisation of expansions as sets of two-valued interpretations [Moore, 1990].

A propositional formula Y extended by the constants $\mathbf{t}$, $\mathbf{f}$ and $\mathbf{u}$ (interpreted as the logical values *true*, *false* and *undefined*, respectively) constitutes the syntactical equivalent of a belief pair [Denecker *et al.*, 1998]. The models of the formula Y replacing each undefined value with $\mathbf{t}$ (called Y^{wk}) yield the set P and replacing each undefined value with $\mathbf{f}$ (Y^{str}) creates the set S. Keep in mind that these replacements are not usual substitutions – they depend on the polarity of $\mathbf{u}$, i.e. the number of negations which have influence on each $\mathbf{u}$: In the case of P, if $\mathbf{u}$ is the polarity of $\mathbf{u}$ is odd, then the letter $\mathbf{u}$ is replaced with $\mathbf{f}$, however, otherwise it is replaced with $\mathbf{t}$.

The next sections present the main definitions of the paper of Denecker, Marek and Truszczyński, afterwards we exploit some useful lemmata. Section 7.3 provides better understanding with some examples; in section 7.4 the completeness proof for several top-level modalities is given, and section 7.5 deals with the case of a single top level modality.

7.1 Definitions: Semantics and Syntax

We give a short review of the definitions of this three-valued semantics and syntax introduced in [Denecker *et al.*, 1998]. First, the semantical characterisation is given, and then the syntactical one is introduced. Both are based on fixed-point-iterations and are equivalent in each step of the iteration process. The idea is to start with an initial belief pair which intuitively corresponds to no knowledge about the literals, i.e. each literal is undefined and the fixed-point iteration in each step fixes some values until a maximal possible number of literals has a definite value.

Definition 7.1.1 *A belief pair $B = (P, S)$ consists of two interpretations P and S with $S \subseteq P$. The initial belief pair $(\mathcal{A}, \emptyset)$ is also denoted $\perp$.*

Definition 7.1.2 *Let (P, S) be a belief pair and I an interpretation. Then the truth function $\mathcal{H}_{B,I}$ is defined inductively (with $\mathbf{f}^{-1} = \mathbf{t}$, $\mathbf{t}^{-1} = \mathbf{f}$, $\mathbf{u}^{-1} = \mathbf{u}$):*

$$\begin{array}{ll}
\mathcal{H}_{B,I}(A) = & I(A) \textit{ if } A \textit{ is an atom} \\
\mathcal{H}_{B,I}(\neg F) = & \mathcal{H}_{B,I}(F)^{-1} \\
\mathcal{H}_{B,I}(F_1 \vee F_2) = & max\{\mathcal{H}_{B,I}(F_1), \mathcal{H}_{B,I}(F_2))\} \\
\mathcal{H}_{B,I}(F_1 \wedge F_2) = & min\{\mathcal{H}_{B,I}(F_1), \mathcal{H}_{B,I}(F_2)\} \\
\mathcal{H}_{B,I}(F_1 \Rightarrow F_2) = & max\{\mathcal{H}_{B,I}(F_1), \mathcal{H}_{B,I}(F_2)^{-1}\} \\
\mathcal{H}_{B,I}(KF) = & \left\{ \begin{array}{ll} \mathbf{t} & \textit{iff } \forall_{J \in P} \mathcal{H}_{B,I}(F) = \mathbf{t} \\ \mathbf{f} & \textit{iff } \exists_{J \in S} \mathcal{H}_{B,I}(F) = \mathbf{f} \\ \mathbf{u} & \textit{else} \end{array} \right.
\end{array}$$

The ordered set $\mathcal{B}$ of all belief pairs is not a lattice, however it is chain-complete.

Definition 7.1.3 *Given a modal theory T, the value of the derivation operator D_T is defined as follows: $D_T(B) = (\{I | \langle B, I \rangle \models_w T\}, \{I | \langle B, I \rangle \models T\})$ with $(B, I) \models_w F$ iff $\mathcal{H}_{B,I}(F) \neq \mathbf{f}$ and $(B, I) \models F$ iff $\mathcal{H}_{B,I}(F) = \mathbf{t}$.*

I is an interpretation, $B = (P, S)$ a belief pair. Therefore, $D_T(B) = (D_T(P), D_T(S))$ is a belief pair and is monotonic on $\mathcal{B}$ ($S \subseteq D_T(S), D_T(P) \subseteq P$) and if B is complete (i.e. $P = S$), then of course $D_T(B)$ is complete, too, and $D(B) = B$. For sake of simplicity we sometimes drop T if clear from the context.

D_T has a least fixed-point. This fixed-point reflects a constructive reasoning process and is hence proposed as semantics [Denecker *et al.*, 1998].

However, for fast implementations one has to introduce another way to calculate this fixed-point. In [Denecker *et al.*, 1998] an equivalent characterisation is presented:

Let F be a three-valued formula. Then F^{wk} is the formula where each occurrence of $\mathbf{u}$ is replaced with $\mathbf{t}$ if this undefined value is of positive polarity (i.e. if it is a positive occurrence) else with $\mathbf{f}$. We denote by F^{str} the formula resulting by replacing each positive occurrence of $\mathbf{u}$ with $\mathbf{f}$ and each negative one with $\mathbf{t}$. $F^{str} \Rightarrow F^{wk}$ is a tautology.

Example 7.1.4 *Let $F = \neg((\mathbf{u} \wedge a) \vee (\mathbf{u} \Rightarrow (b \wedge \neg a)))$. Two undefined values occur, the first one is a negative occurrence, since it is of odd polarity (one negation). However, the second one is somewhat more subtle: it is in the range of one negation, but there is a second negation implicitly given by the implication, which can be written explicitly as follows: $F = \neg((\mathbf{u} \wedge a) \vee (\neg \mathbf{u} \vee (b \wedge \neg a))$. Only if just conjunctions and disjunctions appear in a formula F, we can decide the polarity of an occurrence of $\mathbf{u}$ in a straightforward way. Hence, $F^{str} = \neg((\mathbf{t} \wedge a) \vee (\neg \mathbf{f} \vee (b \wedge \neg a)) = \mathbf{f}$ and $F^{wk} = \neg((\mathbf{f} \wedge a) \vee (\neg \mathbf{t} \vee (b \wedge \neg a)) = \neg b \vee a$. $F^{str} \Rightarrow F^{wk}$ is $\mathbf{f} \Rightarrow (\neg b \vee a)$ is a tautology (due to ex falso quodlibet).*

The models of F^{str} are always a subset of the models of F^{wk}. Indeed, F^{str} and F^{wk}, and hence, F correspond to a belief pair which is made up of these two sets of models.

Definition 7.1.5 *If F is objective then define:*

$$\mathcal{H}_Y(KF) = \begin{cases} \mathbf{t} & \textit{iff } Y^{wk} \vdash F^{str} \\ \mathbf{f} & \textit{iff } Y^{str} \not\vdash F^{wk} \\ \mathbf{u} & \textit{else} \end{cases}$$

otherwise replace all modal atoms KG where G is a propositional formula by $\mathcal{H}_Y(KG)$ to get an objective formula. Let T_Y be the three-valued theory by applying this procedure to each modal atom of the theory T.

The fixed-point iteration starts with $Y = \mathbf{u}$, i.e. with $Y^{str} = \mathbf{f}$ and $Y^{wk} = \mathbf{t}$. The operator $SD_T(Y)$ is defined as T_Y.

Each top level modality is assigned a three-valued truth value. If each truth value remains the same as in the previous iteration, then $SD_T(Y) = Y$.

7.2 Some useful Lemmata

This section contains some results which are implicitly present in [Denecker *et al.*, 1998], however are worked out and presented to provide a better understanding of the following theorems.

We write $F_{b|a}$ representing the formula F where each occurrence of b is replaced with a.

Lemma 7.2.1 *Let X be a set of premises. If $X \vdash F^{str}$, then $X \vdash F_{\mathbf{u}|\mathbf{t}}$ and $X \vdash F_{\mathbf{u}|\mathbf{f}}$. If $X \vdash F_{\mathbf{u}|\mathbf{t}}$ or $X \vdash F_{\mathbf{u}|\mathbf{f}}$, then $X \vdash F^{wk}$.*

Proof. In F^{str} each positive occurrence of $\mathbf{u}$ (even polarity) is replaced with $\mathbf{f}$, for instance if a formula contains $\neg(\ldots \wedge \neg(\mathbf{u} \vee \ldots))$, then $\mathbf{u}$ is replaced with $\mathbf{f}$. Each negative occurrence of $\mathbf{u}$ is replaced with $\mathbf{t}$, for instance an occurrence of $\neg\mathbf{u}$ with no further negations influencing this formula is replaced with $\mathbf{f}$ because $\mathbf{u}$ is replaced with $\mathbf{f}$ (as illustrated in the example above, too).

In $F_{\mathbf{u}|\mathbf{t}}$ each $\mathbf{u}$ is substituted with $\mathbf{t}$ instead. Consider an occurrence of $\mathbf{u}$ prefixed with possibly some negations. In F^{str} it is replaced with $\mathbf{f}$ in such a way that the formula gets "stronger", i.e. satisfies less models while in $F_{\mathbf{u}|\mathbf{t}}$ depending on the negations concerning $\mathbf{u}$, either $\mathbf{t}$ or $\neg\mathbf{t}$ (depending of how many negations influence $\mathbf{t}$) creates a stronger or weaker formula. In the worst case, all $\mathbf{u}$ are bounded with an odd number of negations, in this case F^{str} coincides with $F_{\mathbf{u}|\mathbf{t}}$. Therefore, in the general case $F_{\mathbf{u}|\mathbf{t}} \vdash F^{str}$. The proof follows a similar line w.r.t. the other three cases. □

Lemma 7.2.2 *A truth value* $\mathbf{t}$ *or* $\mathbf{f}$ *of a top-level modality never changes back to* $\mathbf{u}$ *in a subsequent iteration step.*

Proof. Y^{wk} of the current iteration, abbreviated as $T(Y^{wk})$ is a stronger premise than the previous Y^{wk}, i.e. $Y^{wk} \subseteq T(Y^{wk})$.

Consider each F^{str} in two subsequent iteration steps, called $F^{str}(i)$ and $F^{str}(i+1)$. We want to show that it always holds that $F^{str}(i) \Rightarrow F^{str}(i+1)$. For the innermost modalities of T this is trivially valid, since no undefined values are involved resulting from inner modalities, $F^{str}(i) = F^{str}(i+1)$. Now apply induction and say we have proved this fact already for $n-1$ nested modalities. In the case that F has a nesting level n we can assume that in the $i+1$-st iteration some of the undefined values are changed to $\mathbf{t}$ or $\mathbf{f}$ (and due to the induction hypotheses, the ones that are already settled to $\mathbf{t}$ or $\mathbf{f}$ keep this truth value).

Due to Lemma 7.2.1 $F^{str} \Rightarrow F_{\mathbf{u}|\mathbf{t}}$ and $F^{str} \Rightarrow F_{\mathbf{u}|\mathbf{f}}$, and moreover these facts can easily be seen to remain valid if only *some* of the $\mathbf{u}$ are substituted with $\mathbf{f}$ or $\mathbf{t}$, and hence are also valid if some $\mathbf{u}$ are substituted with $\mathbf{t}$ and some with $\mathbf{f}$ (and the others are settled via the evaluation for F^{str}), abbreviated as $F^{str}_{u\rightarrow}$.

$F^{str}(i) \Rightarrow F^{str}_{u\rightarrow}(i)$ therefore implies that $F^{str}(i) \Rightarrow F^{str}(i+1)$, since F^{str} of the subsequent iteration step is created like $F^{str}_{u\rightarrow}$. Overall: $Y^{wk} \vdash F^{str}(i)$ implies that $T(Y^{wk})(\supseteq Y^{wk}) \vdash F^{str}(i+1)$.

The proof for Y^{str} proceeds analogously. □

At each iteration step at least one top level modality needs to change from $\mathbf{u}$ to $\mathbf{t}$ or $\mathbf{f}$ (if not, the fixed-point has already been reached), and at most after $n+1$ iterations, with n is the number of top level modalities, a fixed-point is reached.

Since each step in the iteration corresponds to a step in the "syntactical" iteration stated here, we are allowed to treat the algorithm as efficient for the general case, i.e. its computation is efficient with respect to the semantical characterisation.

If a modal theory T contains no nested modalities, then it is easy so see that $F^{str} = F^{wk} = F$ for each F in T, since no undefined value as a result of evaluating inner modalities occurs in any iteration step.

If a modal theory T contains just one top-level modality, then it is easy to see that the fixed-point is reached after two iterations: Starting with $Y = \mathbf{u}$ (corresponding to the belief pair $\perp$) in the first run, and using $Y = T_Y$ in the second run (only one top level modality can change from $\mathbf{u}$ to a defined value or remain undefined, hence after the second step a fixed-point is reached).

As mentioned in the introduction, this approximation semantics therefore introduces meta-ignorance. The formula $Kp \vee \neg Kp$ is not a tautology.

This semantics is closely related to other non-monotonic formalisms which involve a smallest extension or a smallest model like stationary default logics or well-founded logic programming semantics. Indeed, the main proof follows a similar concept to the proof that reasoning in default logic with stationary extensions is Δ_2^P-hard, however due to the three-valued semantics, it is in many respects more complex and requires a rather tricky construction.

Furthermore, we show that if an autoepistemic theory involves only one top-level modality then the problem is only $\Delta_2^P[O(\log n)]$-hard. This hints that the Δ_2^P-hardness only occurs due to the fixed-point iteration and not if a single step (or a fixed number of steps) is needed in the iteration: At each step of the algorithm a linear number of oracle calls is needed, however it can be displayed similar to an $\mathbf{NP}$-tree [Gottlob, 1995c] and hence is a $\Delta_2^P[O(\log n)]$-complete problem.

One can observe similarities to Carnap's logic regarding substitution instances: $r \vee \neg K\neg p$ is true, but substituting $q \wedge \neg q$ for p yields $\mathbf{f}$, hence it is not closed under substitution. Indeed, the proof of $\Delta_2^P[O(\log n)]$-completeness in case of a single top-level modality exhibits very close relations to Carnap's modal logic.

7.3 Some Examples

Example 7.3.1 *Let* $T = Kp \Rightarrow q$*.*

$\mathcal{H}_{B,I}(Kp \Rightarrow q) = \max\{\mathcal{H}_{B,I}(q), \mathcal{H}^{-1}_{B,I}(Kp)\} = \max\{I(q), \mathcal{H}^{-1}_{B,I}(Kp)\}$.

$\mathcal{H}_{\perp,I}(Kp \Rightarrow q) = \max\{I(q), u\}$.

$D_{Kp\Rightarrow q}(\perp) = (\mathcal{A}, \{\{p,q\},\{\neg p,q\}\})$ *because* $\mathcal{H}_{B,I}(Kp \Rightarrow q) \neq \mathbf{f}$ *for each interpretation in* $\mathcal{A}$ *and* $\mathcal{H}_{B,I}(Kp \Rightarrow q) = \mathbf{t}$ *iff* $q \in I$*.*

Second step of the iteration: $\mathcal{H}^{-1}_{(\mathcal{A};\{p,q\},\{\neg p,q\}),I}(Kp) = t$*. Hence,*

$$\mathcal{H}_{(\mathcal{A};\{p,q\},\{\neg p,q\}),I}(Kp \Rightarrow q) = max\{\mathcal{H}^{-1}_{(\mathcal{A};\{p,q\},\{\neg p,q\}),I}(Kp), I(q)\} = t$$

which is valid for all interpretations and therefore $D^2_{Kp\Rightarrow q}(\perp) = (\mathcal{A}, \mathcal{A})$*. Since* $S = P$ *this belief pair already constitutes the fixed-point.*

Observe that $T = Kp \Rightarrow q$ *has exactly one stable expansion (containing* $\neg Kp$ *but neither* q *nor* $\neg q$*), hence it admits a unique autoepistemic model* W*, therefore* $S = P$ *in this case. The* *objective part of the stable expansion is $cons(\emptyset)$, which is also realized in the fully connected Kripke-structure $\mathcal{A}$. From each world a world is reachable where $\neg p$ is true, therefore $\neg Kp$, hence the structure satisfies this stable expansion.*

Example 7.3.2 *Consider the theory $T = Kp \Rightarrow p$. This theory admits two stable expansions, one containing p and Kp and the other one containing $\neg Kp$ but neither p nor $\neg p$ [Konolige, 1988].*

$\mathcal{H}_{B,I}(Kp \Rightarrow p) = \max\{I(p), \mathcal{H}^{-1}_{B,I}(Kp)\}$. Starting with $B = \bot$, $\mathcal{H}_{B,I}$ is $max\{I(p), \mathbf{u}\}$, yielding $D_{Kp\Rightarrow p}(\bot) = (\mathcal{A}, p)$. $\mathcal{H}_{\mathcal{A},p} = \max\{I(p), u\}$, hence $D_{Kp\Rightarrow p}(\mathcal{A}, p) = (\mathcal{A}, p) = (\{\{p\}, \{\neg p\}\}, \{p\})$.

The fixed-point is computed with the algorithm in the same way: The initial belief pair $\bot$ corresponds to $Y = \mathbf{u}$. Initially, Kp is evaluated to $\mathbf{u}$, hence $SD_{Kp\Rightarrow p}(\mathbf{u}) = \mathbf{u} \Rightarrow p = \neg\mathbf{u} \vee p$. $SD^{wk}_T(\mathbf{u}) = \mathbf{t} \vee p = \mathbf{t}$, $SD^{str}_T(\mathbf{u}) = \mathbf{f} \vee p = p$. Therefore $\mathcal{H}_{SD(Y)}(Kp) = \mathbf{t}$ iff $\mathbf{t} \vdash p$, $\mathbf{f}$ iff $p \not\vdash p$; since both are not fulfilled the evaluation remains $\mathbf{u}$ as in the previous step. Therefore $SD^2_T(\mathbf{u}) = \mathbf{u} \Rightarrow p$. All models of $\mathbf{t}$ yield P and all models of p yield S – as expected, this is the belief pair $(\mathcal{A}, p)$.

Example 7.3.3 *The theory $T = \neg Kp \Rightarrow p$ has no stable expansion (this theory corresponds to the default theory $\left\{\frac{:\neg p}{p}\right\}$).*

$\mathcal{H}_{B,I}(Kp) = \mathbf{u}$ for $B = \bot$, hence $D_{\neg Kp\Rightarrow p}(\bot) = (\mathcal{A}, p)$. Since $\mathcal{H}_{B,I}(Kp)$ remains undefined for the belief pair $(\mathcal{A}, p)$ as in the previous example and $\mathbf{u}^{-1} = \mathbf{u}$, $(\mathcal{A}, p)$ is the fixed-point in this example, too.

7.4 General Case: Δ^P_2-hardness

At each step of the algorithm at least one $\mathbf{u}$ changes to true or false, as a consequence, the iteration takes at most $n+1$ steps, where n is the number of top-level modalities (Lemma 7.2.2). Hence, the computation of Y and the computation of the fixed-point belief pair is in Δ^P_2, since a maximum number of $2n$ oracle calls are needed (as already stated in [Denecker *et al.*, 1998]).

Observe that in this chapter we deal with search problems and not with decision problems. Hence, to be formally correct we would have to use the corresponding complexity classes for search problems instead – anyway, in this setting it is clear how to understand it. However, for sake of correctness, we below give the results for the problems formulated as decision problems, too.

The following problem, MSA-odd, is complete for Δ^P_2 [Wagner, 1990] as discussed in Section 3.9. Given a satisfiable propositional formula F in the propositional atoms $p_1, \ldots, p_n$, decide whether the lexicographically maximum truth assignment w.r.t. the sequence $\langle p_1, \ldots, p_n\rangle$ satisfying F is odd, i.e. assigns 1 to p_n. This truth assignment v is recursively defined as

$$v(p_1) = \chi_{SAT}(F(\mathbf{t}, p_2, \ldots, p_n)),$$

$$v(p_{i+1}) = \chi_{SAT}(F(v(p_1), \ldots, v(p_{i-1}), \mathbf{t}, p_{i+1}, \ldots, p_n))$$

with χ_{SAT} being the characteristic function of satisifiability.

Theorem 7.4.1 *Deciding if an atom occurs in each interpretation of a complete belief pair is Δ_2^P-complete even if no nested modalities are involved.*

Proof. It remains to prove hardness. For this case, consider the modal theory

$$T = (\neg K \neg F_1 \vee r_1) \wedge (K \neg F_1 \vee s_1) \wedge (\neg K \neg F_2 \vee r_2) \wedge (K \neg F_2 \vee s_2) \wedge$$

$$\cdots \wedge (\neg K \neg F_n \vee r_n) \wedge (K \neg F_n \vee s_n)$$

with

$$F_i = (\bigwedge_{j<i} ((r_j \Rightarrow \neg s_j) \wedge (s_j \Rightarrow \neg r_j))) \wedge F(r_1, r_2, \ldots, r_{i-1}, \mathbf{t}, p_{i+1}, \ldots p_n)$$

and r_i, s_i and p_i being propositional variables and $1 \leq i \leq n$. This transformation is clearly feasible in polynomial time in F.

First, we observe the following:

- Intuitively, the s_i serve as negation of the r_i.
- Each $Y(k)$ created during the iteration steps is a conjunction of literals (the k indicates the form of Y at the k-th iteration step).
- Each $Y^{wk}(k)$ and $Y^{str}(k)$ is a conjunction of literals, too, and before a fixed-point is reached, Y^{wk} always exhibits more models than Y^{str} (if both are equal then the fixed-point has been reached, however the inverse conclusion is of course not true).
- $2n$ modalities are involved, however, at most $n+1$ iteration steps are needed since if the modality next to r_i gets a definite value, also the modality w.r.t. s_i is given a truth-value from the set $\{\mathbf{t}, \mathbf{f}\}$.
- Observe that no nested modalities occur, therefore $F = F^{str} = F^{wk}$ in the fixed-point iteration procedure.

We show that the conjunctive formula Y contains s_n exactly if p_n is true in the maximal truth-assignment of the formula $F(p_1, p_2, \ldots, p_n)$. Hence, the computation of Y is at least as hard.

The difficult task in this proof is to get rid of the $\mathbf{u}$ symbols - intuitively, our goal is, that after the final iteration step, Y contains r_i exactly iff p_i is true in the maximal truth-assignment and s_i else. Furthermore, we prove that after the $(k+1)$-st step, all modalities next to r_i and s_i with $i \leq k$ are given definite truth values in $Y(k+1)$.

If we were allowed to use the same proof method as in stationary default logic [Gottlob, 1995] we could use a theory like

$$\bigwedge_i \neg K \neg F(q_1, \ldots, q_{i-1}, \mathbf{t}, p_{i+1}, \ldots, p_n) \Rightarrow q_i,$$

however it is not difficult to see that this does not work in our case, because the three valued semantics does not settle the required truth values in the required steps.

Therefore, we introduced this special, rather involved construction of the formulae F_i. The F_i are of this form, because $Y^{str}(k)$ (a conjunctive formula) always needs to be a subset of $Y^{wk}(k)$, and $Y^{wk}(k+1)$ shall have less conjunctions than $Y^{wk}(k)$ while $Y^{str}(k+1)$ shall consist of more conjunctions than $Y^{str}(k)$ – however we also want that each $Y^{wk}(k)$ and $Y^{str}(k)$ are *somehow equal*, at least w.r.t. a formula F (i.e. equal for the variables occurring in F) to avoid undefined values in the crucial iteration steps.

Due to this construction, the final Y will contain no undefined values, hence the corresponding belief pair is total, and the hardness result therefore even holds for total belief pairs.

We show that after the $(k+1)$-st step, it is decided whether r_k or s_k belongs to Y and that s_i belongs to Y exactly if p_i is true in the maximum truth evaluation, and r_i otherwise.

Before the first step, $Y^{wk}(0) = \mathbf{t}$ and $Y^{str}(0) = \mathbf{f}$. Now we can apply this to each involved modality, however, currently many will be left undefined. Some could have changed to $\mathbf{t}$ (namely if F_i is unsatisfiable; in the first step, none can change to $\mathbf{f}$ due to ex falso quodlibet), however, w.l.o.g. we are allowed to leave some of the modalities in $Y(k)$ undefined although they already would be settled. We can change them to their definite truth value in any later step we like, because each $Y^{wk}(k+1) \supset Y^{wk}(k)$ (cf. proof of Lemma 7.2.2), and $Y^{str}(k+1) \subset Y^{str}(k)$, hence: $Y^{wk}(k) \vdash F$ implies $Y^{wk}(k+1) \vdash F$ and $Y^{str}(k) \not\vdash F$ implies $Y^{str}(k+1) \not\vdash F$, respectively. Therefore, we always can allow to construct a $Y(k)$ with evaluating some modalities as undefined which already should have a definite truth-value, as it does no harm and can be inferred in any later step where it is needed, anyway (we only change the value of one undefined modality in each step in this setting; we consider an arbitrary general F as for such F it can be assumed that at each step only one modality changes its value).

Hence we are allowed to assume that after the first step all modalities are evaluated to false and Y is of the form $\bigwedge_i(\mathbf{u} \vee r_i) \wedge (\mathbf{u} \vee s_i)$ (the modalities that should have evaluated to a definite truth value will be evaluated in later steps).

Hence, $Y^{wk}(1) = \mathbf{t}$, and $Y^{str}(1) = \bigwedge_i(r_i \wedge s_i)$.

If in a later step the modality next to r_i changes to $\mathbf{t}$, then $(\mathbf{f} \vee r_i) \wedge (\mathbf{t} \vee s_i) = r_i$, which will be shown to be equivalent to p_i *is false in the* maximum truth evaluation. If the modality of F_i changes to $\mathbf{f}$, then s_i instead is part of the conjunctions of the current Y. In the $(k+1)$-st step all modalities corresponding to F_i with $i < k$ changed to $\mathbf{t}$ or $\mathbf{f}$ as shown below, and therefore for all $i < k$ exactly one of r_i or s_i belongs to Y.

Hence, if we prove that in the $k+1$-step at least one, namely exactly the one of the i-th modality, changes to $\mathbf{t}$ or $\mathbf{f}$, then Y^{wk} of the current step is a conjunction of r_i and s_i, of each $i < k$. And Y^{str} is Y^{wk} conjuncted with $r_j \wedge s_j$ with $j \geq k$.

Assume, we already proved this fact for the i-th modality; we want to prove it for the $(i+1)$-st. For easier readability, we illustrate the proof with an example showing the fifth iteration step. The first three modalities here are already settled to s_i or r_i, respectively and now the truth value of the fourth modality is decided:

$$r_1 \wedge s_2 \wedge r_3 \vdash \neg F_4$$
$$r_1 \wedge s_2 \wedge r_3 \wedge (r_4 \wedge s_4) \wedge (r_5 \wedge s_5) \not\vdash \neg F_4$$

with $F_4 = \bigwedge_{i=1,2,3}((r_i \Rightarrow \neg s_i) \wedge (s_i \Rightarrow \neg r_i)) \wedge f(r_1, r_2, r_3, \mathbf{t}, p_5, \ldots p_n)$[1]. Due to the deduction theorem the above assertions can be rewritten as

$$r_1 \wedge s_2 \wedge r_3 \wedge (\bigwedge_{i=1,2,3} (r_i \Rightarrow \neg s_i) \wedge (s_i \Rightarrow \neg r_i)) \vdash \neg f(r_1, r_2, r_3, \mathbf{t}, p_5, \ldots, p_n) \tag{1}$$

$$r_1 \wedge s_2 \wedge r_3 \wedge (\bigwedge_{i=1,2,3} (r_i \Rightarrow \neg s_i) \wedge (s_i \Rightarrow \neg r_i)) \wedge (r_4 \wedge s_4) \wedge (r_5 \wedge s_5)$$
$$\not\vdash \neg f(r_1, r_2, r_3, \mathbf{t}, p_5, \ldots, p_n) \tag{2}$$

Hence, due to modus ponens and since s_i do not occur in the formula F, statement (1) is true exactly if

$$r_1 \wedge \neg r_2 \wedge r_3 \vdash \neg f(r_1, r_2, r_3, \mathbf{t}, p_5, \ldots, p_n). \tag{1'}$$

and in the statement (2), furthermore the higher r_i and s_i are irrelevant for the derivation, hence:

$$r_1 \wedge \neg r_2 \wedge r_3 \not\vdash \neg f(r_1, r_2, r_3, \mathbf{t}, p_5, \ldots, p_n), \tag{2'}$$

thus creating the converse statement of (1). Therefore, it suffices to continue with only statement (1), which is equivalent to:

$$\vdash \neg f(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}, p_5, \ldots, p_n)$$

which in turn is equivalent to

$$\neg sat(F(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}, p_5, \ldots, p_n)).$$

Due to the construction of the modal theory T, s_i occurs in $Y(k)$ exactly if F_i is satisfiable, and hence exactly if $p_i = 1$. More exactly, in this example, the following statements (A)-(F) are equivalent:

$$p_4 \in MAXSAT \tag{A}$$
$$sat\, F(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}, p_5, \ldots, p_n) \tag{B}$$
$$\not\vdash \neg F(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}, p_5, \ldots, p_n) \tag{C}$$
$$r_1 \wedge s_2 \wedge r_3 \wedge \bigwedge_{i=1,2,3} (s_i \Rightarrow \neg r_i) \not\vdash \neg F(r_1, r_2, r_3, \mathbf{t}, p_5, \ldots, p_n) \tag{D}$$
$$KF_4 \text{ evaluates to false} \tag{E}$$
$$(\neg \mathbf{f} \vee r_4) \wedge (\mathbf{f} \vee s_4) \tag{F}$$

[1] Actually, $(r_i \Rightarrow \neg s_i)$ is not needed, only $(s_i \Rightarrow \neg r_i)$, because only if s_i it somehow has to be made explicit that this is the negation of r_i. However, for symmetry reasons we kept it, nevertheless.

occurs in the conjunctions of Y, which is that s_4 occurs in Y. And if p_4 is not in MAXSAT, then r_4 occurs instead in Y.

The general proof follows the same pattern, and hence it is proved that Y contains s_n exactly if p_n is in the maximum truth evaluation, and r_n exactly if p_n is not.

We managed to introduce a negation without disturbing the monotonicity of Y^{str} and Y^{wk} (if negations were used in a direct way, Y^{str} would always yield **f** and our main goal is that Y^{str} and Y^{wk}, although they are different are w.r.t. a formula which should be inferred, equal – because the other atoms occurring additionally in Y^{wk} do not occur in the formula). To summarize, for $Y^{wk}(k)$ we got the same formula as for $Y^{str}(k)$, except that for $i \geq k$ $r_i \wedge s_i$ are included. However, these r_i and s_i do not occur in F_4, so can be dropped as premises (and they do no harm, since the implications are only for r_i with $i = 1, 2, 3$) making Y^{str} and Y^{wk} equivalent w.r.t. F_4.

Therefore, since the statements are dual to each other – one with $\vdash$, and the other exactly the same with $\not\vdash$, this leaves no room for the value $\mathbf{u}$ for this particular modality but to turn to true or false. And since a modality which changed to true or false remains at this value, and at latest at the k-th step, the $(k-1)$-st modality changes to a defined truth-value, and at each step at least one modality changes to a defined value, after $n+1$ steps we get Y as conjunction of r_i's and s_i's with exactly r_n in Y iff p_n = 1 is in the maximum truth evaluation.

Determining Y is as at least as hard as determining one of its conjunctive atoms, hence this is a Δ_2^P-hard problem. □

Corollary 7.4.2 *Determining Y is a Δ_2^P-complete problem, and therefore determining a belief pair is a Δ_2^P-complete problem, too.*

As mentioned above, to be formally correct we need to state this problem as decision problem. Therefore, we define the following. Let $B = (P, S)$:

$B \models F$ iff $\forall I \in P : I \models F$ (iff $Y^{wk} \models F$)

$B \not\models F$ iff $\exists I \in S : I \not\models F$ (iff $Y^{str} \not\models F$)

I is an interpretation, F is an arbitrary two-valued propositional formula. Observe that the model relation for B is defined in a not symmetric way as it is possible that B neither satisfies F nor not satisfies F representing the undefined value.

Corollary 7.4.3 *Let F be a two-valued propositional formula and T a modal theory. Both deciding if $B_T \models F$ and if $B_T \not\models F$ is Δ_2^P-complete.*

Proof. We only give a proof for the case of $\models$, the other proof is analogue. Hardness follows from the proof of the previous theorem as it is already Δ_2^P-hard to determine if r_n is a conjunct of Y, i.e. that $Y^{wk} \models F$ with $F = r_n$.

For membership compute Y as usual and check with a single call to an oracle in $\mathbf{NP}$ whether $Y^{wk} \models F$. □

7.5 Restriction to a Single Top Level Modality

A linear number of oracle calls, namely two for each occurring modality are needed in the "simple" algorithm. However, due to results of Gottlob [1995c] we can show that the solution of this problem is closely related to $\mathbf{NP}$-trees and hence easier to solve than the general case since no fixed-point iteration occurs (to be exact, two (i.e. a fixed number independent from input) iteration steps occur, however the first step can be neglected).

$\mathbf{NP}$-trees are one of the most used concepts to show that a problem is in $\Delta_2^P[O(\log n)]$ – the proof that the evaluation $\mathbf{NP}$-trees is in this class is rather involved and can be found in the above cited paper. As soon as it is possible to display a problem as an $\mathbf{NP}$-tree, this is a tree where each node relates to an $\mathbf{NP}$ oracle call and then passes the value to its parent node until the root is reached. In this section we use a similar concept, however we need to alter the concept of $\mathbf{NP}$-trees to a more generalized concept and describe how the proof in the paper of Gottlob [1995c] needs to be altered to suffice our requirements.

Note that in Section 5.3 we used a different powerful technique to show that a problem is in $\Delta_2^P[O(\log n)]$ – guessing data-structures and applying binary search.

Consider the theory

$$T = \neg K \neg F(q_1, q_2, \ldots, q_{n-1}, \top)$$

with

$$q_i = \neg K \neg F(q_1, q_2, \ldots, q_{i-1}, \top, p_{i+1}, \ldots, p_n)$$

inductively defined. Observe that the q_i are not propositional atoms, but just abbreviations for negated modal formulae.

However, this theory T has length which is exponential in the length of the formula F, hence one cannot use this to show Δ_2^P-hardness. Indeed, the problem is not even Δ_2^P-hard, but a little bit easier.

Carnap's modal logic is not described in Section 2.4 as it is a somewhat different logic than the usual $S5$ etc. – Carnap's logic is not closed under substitution, and the modalities correspond to the meta-relation of provability. Details about Carnap's logic can be extracted from [Gottlob, 1997].

Theorem 7.5.1 *Determining the fixed-point of a theory T with only one top level modality is $\Delta_2^P[O(\log n)]$-hard.*

Proof. We show hardness by using a reduction to the $\Delta_2^P[O(\log n)]$-complete problem of deciding validity of a formula in Carnap's modal logic, namely we show that $\Box F$ is valid in C (which is the same as if the modal formula F is valid where F does not need to contain only one top-level modality) exactly iff $r \vee KF$, with r being a new propositional atom not occurring in F, has the fixed-point $Y = \mathbf{t}$. r has the task of enforcing a value $\mathbf{t}$ or $\mathbf{f}$ instead of $\mathbf{u}$.

Since only one top-level modality occurs, the fixed-point iteration yields a fixed-point after at most 2 steps (in general $n + 1$ steps with n modalities).

It is easy to see that after the first iteration of the formula $r \vee KF$, $Y = r \vee \mathbf{u}$ or $Y = r \vee \mathbf{t}$. The modality evaluates to $\mathbf{t}$ exactly if F^{str} is a tautology. We can treat the second case as a special case of the first one (where $\mathbf{t}$ is reached in the second iteration step).

In the first case, the next iteration step has $Y^{str} = r$ and $Y^{wk} = \mathbf{t}$. Hence, each modal subformula KG where G is a three-valued formula such that the inner modalities are already replaced with a three-valued truth value, is evaluated in the following way:

$$\begin{cases} \mathbf{t} & \text{iff } \mathbf{t} \vdash G^{str} \\ \mathbf{f} & \text{iff } r \nvdash G^{wk} \\ \mathbf{u} & \text{else} \end{cases}$$

Since r does not occur in any modal subformula, $r \nvdash G^{wk}$ is equal to $\mathbf{t} \nvdash G$, each subformula is forced to evaluate either to $\mathbf{t}$ or to $\mathbf{f}$.

In the second iteration, no $\mathbf{u}$ is left from higher nested modalities, so in this step, each G^{str} equals G^{wk}.

Hence, the corresponding modality evaluates to true exactly if G^{prop} (where the inner modalities are all inductively replaced) is a propositional tautology, and to $\mathbf{f}$ otherwise. I.e.

$$\begin{cases} \mathbf{t} & \text{iff } \mathbf{t} \vdash G^{prop} \\ \mathbf{f} & \text{iff } \mathbf{t} \nvdash G^{prop} \end{cases}$$

It remains to be shown that this second iteration step evaluates each sub-formula exactly like the subformulae in Carnap's modal logic.

In Carnap's modal logic $\Box(\equiv \neg\Diamond\neg)$ can be considered as a strong provability operator that expresses the meta-predicate $\models_C$ in the language: When evaluating formulae in Carnap's modal logic each sub-formula is evaluated. If $H = \Box G$, then $\models_C H$ iff $\models_C G$ (and in a dual way for $\Diamond$). And if G contains further modalities, then the procedure is iteratively applied and each modal sub-formula $\Box I$ with propositional I is replaced with $\mathbf{t}$ iff $\vdash I$, otherwise with $\mathbf{f}$ [Gottlob, 1995c]:

$$\begin{cases} \mathbf{t} & \text{iff } \vdash I \\ \mathbf{f} & \text{iff } \nvdash I \end{cases}$$

The necessity-operator $\Box$ can be identified with the operator K. Hence, this is in essence the same procedure as with the evaluation algorithm given above. Therefore, G evaluates to true in the second iteration run, exactly if G^{prop} is a propositional tautology, which in turn holds exactly if $\models_C G$.

Therefore, $\Box F$ is valid in C exactly iff $Y = r \vee \mathbf{t} = \mathbf{t}$.

(The proof would follow a similar line w.r.t. $\Diamond$ as top-level modality (one could even immediately use a modal formula g with one arbitrary top-level modality instead of distinguishing two cases KF and $\neg K \neg F$, but for simplicity we chose $\Box F$, as it is sufficient to show hardness for one particular case like $KF \vee r$ and the corresponding problem in Carnap's logic is also $\Delta_2^P[O(\log n)]$-hard for $\Box F$ with F a modal formula). $\Box$

Example 7.5.2 *A very simple example shall highlight the facts of the proof. Consider the formula $\Box p$. This formula is not valid in C. In the second iteration step, we get $\mathbf{t} \vdash p$ resp. $r \not\vdash p$. Since p is not a tautology, this modality evaluates to false, the same as in Carnap's logic (and Y evaluates to r; Y always evaluates to r if the modal formula is not valid in C). However, consider $\Diamond p$, which is a valid formula in C, because p is satisfiable. In the second iteration step we get $\mathbf{t} \vdash \neg p$ and $r \not\vdash \neg p$. Hence we get $Y = r \vee \neg \mathbf{f} = \mathbf{t}$.*

In this proof we have furthermore shown some minor pleasant facts:

- By disjunctively adding a *new* atom r to T, every incomplete belief pair for a theory of the form KF or $\neg KF$ becomes a complete belief pair (i.e. every Y possibly containing $\mathbf{u}$ gets two-valued).
- We exhibited a very close relationship to Carnap's modal logic: $Y = \mathbf{t}$ for $T = r \vee Kg$ exactly if the modal formula g is valid in C (and exactly if $\Box g$ is valid of course, due to the definition) and due to [Gottlob, 1997], exactly if $g \in st(\emptyset)$ (g is in the stable set of the empty set).

Let us proceed with a definition to be used in a subsequent membership proof:

Definition 7.5.3 *A 3val-tree is defined by a triple $\langle Var, G, F_R \rangle$ consisting of*

- *a set of propositional variables $Var = \{v_1, \ldots, v_n\}$, called the linking variables.*
- *a tree $G = \langle V, E \rangle$, the nodes $V = \{F_1, \ldots, F_n\}$ representing three-valued propositional formulae and we identify the name F_i of a node with the name of the formula represented by the node. Each three-valued propositional formula F_i, in addition to zero or more "private" variables not appearing in any other node, contains the following linking variables: $\{v_j | \langle F_j, F_i \rangle \in E\}$.*
- *A distinguished terminal node F_R, for some $R \in \{1, \ldots, n\}$, called the result node, which is not evaluated (as we are interested in the complete formula after the top-level modality has been evaluated).*

For each 3val-tree the truth value assignment σ to the propositional linking variables $v_1, \ldots, v_n$ is defined as follows. $\sigma(v_i)$ is true if $A \vdash F_i^{\prime str}$, where $F_i^\prime$ is a three-valued formula where each linking variable v_j is replaced by $\mathbf{t}$ iff $\sigma(v_j) = true$, by $\mathbf{f}$ iff $\sigma(v_j) = false$ and by $\mathbf{u}$ else; $\sigma(v_i)$ is false if $B \not\vdash F_i^{\prime wk}$ and $\sigma(v_i)$ is undefined else.

A and B are two-valued propositional formulae with the additional requirement that $B \vdash A$.

3val-trees are extending the concept of $\mathbf{NP}$-trees. An $\mathbf{NP}$-tree is very similar to such a tree, however, the F_i are just two-valued propositional formulae, and depending on satisfiability of F_i the linking variable receives the value $\mathbf{t}$ or $\mathbf{f}$.

Remember that evaluating an $\mathbf{NP}$-tree is in $\Delta_2^P[O(\log n)]$ (Section 3.10) – Gottlob [1995c] showed this fact with a rather involved proof. We do not go into the details of the corresponding proof of $\mathbf{NP}$-trees as we do not need them, however would like to present some intuition: A hypertree is a tree where some nodes are fusioned together into one node - intuitively, a hypernode H represents a set of nodes whose queries can be answered in parallel. $comb(H)$ represents the combination of output of the nodes in H and serves as output value of the hypernode.

The intuition beyond Gottlob's proof is the following: First, a tree is converted into a hypertree. The depth of the hypertree after using all possible fusions is lesser or equal to $\log n$ with n is the depth of the initial tree already hinting that not a polynomial number of oracle calls is required, however due to $O(n)$ nodes this not necessarily needs to be the case. Therefore, the hypertree is afterwards dissolved into a directed acyclic graph and furthermore, this is not a general directed acyclic graph but one with an "admissible" weighting function – and for such graphs the problem is only $\Delta_2^P[O(\log n)]$-complete which consummates that proof.

Theorem 7.5.4 *Determining the fixed-point of a theory T with only one top level modality is in $\Delta_2^P[O(\log n)]$.*

Proof. Instead of going to the basics of the corresponding proof of $\mathbf{NP}$-trees and generalising to 3val-trees, we solve a 3val-tree in a way which is analogous to evaluating an $\mathbf{NP}$-tree to show that this task can be completed with a logarithmic number of oracle calls only.

Let T be a modal theory with only one top-level modality. One top-level modality means two iteration steps, the first one starting with $Y = \mathbf{u}$ and the second one with T_Y, where the top-level modality is replaced with one of the three truth-values.

Hence, the fixed-point is reached after two iteration steps, in the first step, starting with $Y = \mathbf{u}$, the modality K can be evaluated to $\mathbf{u}$ or $\mathbf{t}$.

So, either we analyse two trees, show that both are in $\Delta_2^P[O(\log n)]$ and hence the overall procedure is in this class (both trees just use different A and B), or else, it is far easier to forget about the first step and treat the result of the first step as if the top-level modality were evaluated to $\mathbf{u}$. In the first step, the top-level modality could change to $\mathbf{t}$, but due to Lemma 7.2.1, it is valid that $Y_{u|t} \vdash Y^{wk}$ and $Y^{str} \vdash Y_{u|t}$: Hence: $Y^{wk} \vdash F^{str}$ implies that $Y_{u|t} \vdash F^{str}$, and $Y^{str} \not\vdash F^{wk}$ implies that $Y_{u|t} \not\vdash F^{wk}$. This means that even if in the first run the modality evaluates to true, we can use $\mathbf{u}$ instead in the second iteration, as the truth value that we gain with using weak/strong implies the truth value with using $\mathbf{t}$ instead of $\mathbf{u}$ (after the first iteration, the modality cannot be false, as $Y^{str} = \mathbf{f}$ implies everything).

Hence, we construct a 3val-tree as follows. $A = T^{str}_{K(\dots)|u}$ and $B = T^{wk}_{K(\dots)|u}$ with $K(\dots)$ being the only top-level modality (i.e. A and B are two-valued formulae; remember that $K(\dots)$ is defined as atom in the modal language).

The bottom nodes of the tree are the propositional formulae in the innermost modalities K. The other nodes are constructed iteratively with using the corresponding linking variables instead of the modalities. Additionally, a special result node is added, representing the formula T with the only top-level modality substituted by the corresponding linking variable (our aim is not to get a truth-value back as in standard-$\mathbf{NP}$-trees, but a formula where the truth-value for the only top-level modality is settled).

We display this 3val-tree as "almost" $\mathbf{NP}$-tree with the following algorithm: Instead of one satisfiability oracle call in the evaluation process two oracle calls for each F_i are made: $A \vdash F_i^{str}$ and $B \vdash F_i^{wk}$. Both yield $\mathbf{t}$ means that the value $\mathbf{t}$ should be given to the linking variable, if both yield $\mathbf{f}$ then $\mathbf{f}$ should be returned, and if one yields $\mathbf{t}$ (the second one) whereas the first one yields $\mathbf{f}$, then the value $\mathbf{u}$ should be given to the linking variable.

However, to allow that it remains the way that only true or false is passed to a linking variable (oracle calls of decision problems can only answer *yes* or *no*) we use for each F_i two linking variables v_i and w_i. In the node where usually v_i would appear, we use the formula $(w_i \wedge \mathbf{u}) \vee v_i$ and assign $\mathbf{t}$ to v_i if $A \vdash F_i^{'str}$ (and $\mathbf{f}$ else) and $\mathbf{t}$ to w_i if $B \vdash F_i^{'wk}$.

Each such formula F_i is three-valued, not two-valued like in $\mathbf{NP}$-trees. However, creating F^{str} and F^{wk} for the next two oracle calls is polynomial, hence this poses no problem.

Therefore, this *adapted 3val-tree* is very similar to a standard $\mathbf{NP}$-tree, with the only difference that two oracle calls are required at each node and that each F_i is three-valued. However, it is straightforward that these changes do not raise the complexity (formally, modifying Gottlob's proof in [Gottlob, 1995c] in some minor issues will suffice). □

Example 7.5.5 *Consider the theory $T = K(K(\neg p \wedge q) \Rightarrow Kp) \vee p$. As stated in the proof, without restricting generality we are allowed to use $\mathbf{u}$ for creating A and B and only run the second iteration. $Y = \mathbf{u} \vee p$ establishes $A = \mathbf{t}$ and $B = p$.*

To display this formula as a 3val-tree, let F_1 be $\neg p \wedge q$ and F_2 be p (F_1 and F_2 are the leaves of this tree, corresponding to the innermost modalities). Their father node F_3 is $v_1 \Rightarrow v_2$, and, in the adapted 3val-tree $F_3 = ((w_1 \wedge \mathbf{u}) \vee v_1) \Rightarrow ((w_2 \wedge \mathbf{u}) \vee v_2))$. F_R, the father of F_3, is the special result node. F_R always has a single child only, and we are interested which truth value this single child returns (only one top-level modality). Hence, this is a decision problem, although we return a formula as result, namely the formula T where the top-level modality is replaced with one of $\mathbf{t}$, $\mathbf{f}$ or $\mathbf{u}$.

The linking variables v_1 and w_1 of F are evaluated as $\mathbf{f}$ and $\mathbf{f}$ (meaning that $B \not\vdash F_i^{wk}$ is true). The ones of F_2 are evaluated as $v_2 = \mathbf{f}$ and $w_2 = \mathbf{t}$. Hence, $F_3' = \mathbf{u}$ (Now, you could proceed with $F_3^{'str} = \mathbf{f}$ etc. to get F_R, however, after having F_3 it is clear how F_R looks like). So $Y = \mathbf{u} \vee p$ and the corresponding belief pair $B = (\mathcal{A}, \{\{p, q\}, \{p, \neg q\}\})$.

Observe that it is not possible in the above proof to view two independent $\mathbf{NP}$-trees, one always evaluating F^{wk} (as in the proof to false or true only) and one always evaluating F^{str} and in the end, if one results in $\mathbf{f}$ and one in $\mathbf{t}$ to derive $\mathbf{u}$ for the top-level modality. This would only be possible if it would hold that $F_{\mathbf{u}|\mathbf{f}} \Rightarrow F^{str}$ – because undefined values are passed as false ones (in one of the trees), however in the original tree the truth value of F^{str} is required in the next step – then it would be sufficient to look for $\mathbf{u}$ in the final step only (however, only the other direction holds, namely that $F^{str} \Rightarrow F_{\mathbf{u}|\mathbf{f}}$). Thus the outcome of the two independent trees could be $(\mathbf{f}, \mathbf{t})$, although the outcome of the tree of the proof may be $(\mathbf{t}, \mathbf{t})$ or $(\mathbf{f}, \mathbf{f})$. Hence the proof cannot be simplified to two $\mathbf{NP}$-trees.

As in the previous section, we state the problem as decision problem:

Corollary 7.5.6 *Let F be a two-valued propositional formula and T a modal theory involving only one top level modality. Both deciding if $B_T \models F$ and if $B_T \not\models F$ is $\Delta_2^P[O(\log n)]$-complete.*

Proof. We only give a proof for the case of $\models$, the other proof is analogue. Hardness follows from the proof of the previous theorem as it is already $\Delta_2^P[O(\log n)]$-hard to determine if the top level modality evaluates to $\mathbf{t}$: Consider $T = F \vee \neg KG$ with F and G propositional two-valued formulae. Then deciding $Y^{wk} \models F$ is $\Delta_2^P[O(\log n)]$-hard, too (only if KG evaluates to true, then $Y^{wk} = F$, else $Y^{wk} = \mathbf{t}$).

For membership compute Y as in Theorem 7.5.4 and check with a single call to an oracle in $\mathbf{NP}$ whether $Y^{wk} \models F$. □

With these proofs we definitely settled the complexity of fixed-point 3-valued semantics for autoepistemic logic in the general case (and for the case of no nested modalities) and for the restricted setting of one top-level modality.

This concludes the discussion of new results; the next section recapitulates the results presented in this report.

Chapter 8

Overview of Results

In the first four chapters, we gave an introduction to the topics of NMR and complexity theory. In the following chapters, we exhibited a number of new results of which the most important are presented in the following tables and summarized in the following.

A summary of our results concerning model checking for default logic for standard default logic is presented in Table 8.1, together with the results in [Liberatore and Schaerf, 1998] (which are marked with "•"). In Table 8.2 the results concerning weak model checking are presented.

To show that model checking for normal default theories is in $\Delta_2^P[O(\log n)]$ we improved techniques for guessing data-structures Gottlob used in [Gottlob, 1995] (about stationary default logic): Some steps in the main proof of that chapter consisted of guessing a whole structure, including for instance some interpretations; this allowed us to use the model relation instead of the inference relation and hence to "move the hardness of classical propositional inference into the previous guessing of a structure". The other main tool we used was binary search, a very useful means to obtain a logarithmic number of calls to an oracle.

Probably this result and the achieved results for other presented default logics can in a similar way be generalised to other variants of default reasoning considered in [Liberatore and Schaerf, 1998].

The class of normal default theories in extended Horn normal form is an ample class admitting a polynomial algorithm for model checking and hence is of enormous interest for practical applications.

	General	Semi-Normal	Normal
General	Σ_2^P-complete •	Σ_2^P-complete •	$\Delta_2^P[O(\log n)]$-cmpl.
Prerequisite-free	Σ_2^P-complete •	Σ_2^P-complete •	$\mathbf{coNP}$-complete •
Df-EHNF	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{P}$-complete
Df-EHNF/Prer.-free	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{P}$-complete

Table 8.1: *Complexity of Model Checking for Default Theories*

	General	Semi-Normal	Normal
General	Σ_2^P-complete	Σ_2^P-complete	Σ_2^P-complete
Prerequisite-free	Σ_2^P-complete	Σ_2^P-complete	$\mathbf{coNP}$-complete
Df-EHNF	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete
Df-EHNF/Prer.-free	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{P}$-complete

Table 8.2: *Complexity of Weak Model Checking for Default Theories*

	General	Semi-Normal	Normal
General	Σ_2^P-complete	Σ_2^P-complete	Σ_2^P-complete
Prerequisite-free	Σ_2^P-complete	Σ_2^P-complete	Σ_2^P-complete
Df-EHNF	Σ_2^P-complete	Σ_2^P-complete	Σ_2^P-complete
Df-EHNF/Prer.-free	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete

Table 8.3: *Complexity of Model Checking for Disjunctive Default Theories*

Weak model checking has been shown to coincide w.r.t. complexity with model checking for all considered cases except for normal default theories with prerequisites; the non-constructive nature of weak extensions forbids an easier procedure for model checking. This result is also connected to the fact that no modular translation from default logic into autoepistemic logic exists [Gottlob, 1995b], as the prerequisites are treated in a very different way.

We used our results to show that model checking in autoepistemic logic (logic **N**) is Σ_2^P-complete, since the objective parts of stable expansions (**N**-expansions) correspond to weak extensions (extensions) in default logic.

Our results are also useful for further comparisons of default logic with other non-monotonic formalisms (as indicated in Section 5.3).

The collection of Tables 8.3 to 8.6 present the results we derived for complexity of model checking for disjunctive default logic. We showed that even model checking with disjunction-free and EHNF ddts is Σ_2^P-complete; however, if furthermore prerequisite-free, the complexity decreases one level and turns the problems into $\mathbf{NP}$-complete ones - this result is still valid if every consequent and all justifications are limited to single literals, and hence equally valid for model checking (not answer

	General	Semi-Normal	Normal
General	Σ_2^P-complete	Σ_2^P-complete	Σ_2^P-complete
Prerequisite-free	Σ_2^P-complete	Σ_2^P-complete	Σ_2^P-complete
Df-EHNF	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete
Df-EHNF/Prer.-free	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete

Table 8.4: *Complexity of Model Checking for Limited Disjunctive Default Theories*

	General	Semi-Normal	Normal
General	Σ_2^P-complete	Σ_2^P-complete	Σ_2^P-complete
Prerequisite-free	Σ_2^P-complete	Σ_2^P-complete	$\mathbf{coNP}$-complete
Df-EHNF	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete
Df-EHNF/Prer.-free	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{P}$-complete

Table 8.5: *Complexity of Weak Model Checking for Disjunctive Default Theories*

	General	Semi-Normal	Normal
General	Σ_2^P-complete	Σ_2^P-complete	Σ_2^P-complete
Prerequisite-free	Σ_2^P-complete	Σ_2^P-complete	$\mathbf{coNP}$-complete
Df-EHNF	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete
Df-EHNF/Prer.-free	$\mathbf{NP}$-complete	$\mathbf{NP}$-complete	$\mathbf{P}$-complete

Table 8.6: *Complexity of Weak Model Checking for Limited Disjunctive Default Theories*

set checking!) in EDLPs (see Section 5.8). Using limited ddts the complexity of model checking in case of disjunction-free and EHNF ddts turns out to be only as hard as for standard default logic for almost all considered cases due to creating an independence of two intermingled sources of complexity. However, one case remains harder, namely the case of prerequisite-free and normal ddts, since the complexity introduced with effective disjunctions remains.

In the disjunctive default logic setting the complexity of weak model checking turns out to be the favourable one w.r.t. complexity, at least in the way we generalised it. The difference between limited and not limited vanishes, because one source of complexity disappears. This allows also for an easier complexity in case of normal prerequisite-free theories, for whom model checking is even polynomial if disjunction-free or EHNF. The results of weak model checking for ddts hence are the same as for weak model checking with standard default theories.

Table 8.7 summarises the results for stationary default logic. Although model checking is somehow related to brave reasoning, because an interpretation needs to be a model of at least one extension, the fact that a smallest extension exists of course allows an easier model checking algorithm.

And finally, in Chapter 7, we solved an open problem left by Denecker, Marek and Truszczyński

	General	Semi-Normal	Normal
General	Δ_2^P-complete	Δ_2^P-complete	$\Delta_2^P[O(\log n)]$-complete
Prerequisite-free	Δ_2^P-complete	Δ_2^P-complete	$\mathbf{coNP}$-complete
Df-EHNF	$\mathbf{P}$	$\mathbf{P}$	$\mathbf{P}$
Df-EHNF/Prer.-free	$\mathbf{P}$	$\mathbf{P}$	$\mathbf{P}$

Table 8.7: *Complexity of Stationary Model Checking for Default Theories*

about the exact complexity classification of computing a belief pair of a modal theory with three valued semantics. We distinguished the following cases: First, the case of no nested modalities, for which the problem still remains Δ_2^P-complete, and on the other hand, the case of one single top-level modality (with nested modalities allowed) which slightly decreases the complexity (because no fixed-point iteration is needed) to $\Delta_2^P[O(\log n)]$-completeness.

Further work may include investigation of the complexity of model checking for other widely used variants of default reasoning, and most importantly for practical applications, to isolate and enlarge further tractable classes by imposing only minor restrictions on the defaults. Moreover, our results make it possible to draw various new parallels with other knowledge representation formalisms.

Bibliography

[Baral and Subrahmanian, 1992] C. R. Baral and V. S. Subrahmanian. Stable and Extension Class Theory for Logic Programs and Default Logics. *Journal of Autom. Reasoning* 8 (1992) 345-366.

[Baumgartner and Gottlob, 1999] R. Baumgartner and G. Gottlob. On the Complexity of Model Checking for Propositional Default Logics: New Results and Tractable Cases. *Proc. of IJCAI-99* (1999) 64-69.

[Baumgartner and Gottlob, 1999b] R. Baumgartner and G. Gottlob. On the Complexity of Model Checking for Propositional Default Logics: New Results and Tractable Cases (Extended Version). *Technical Report DBAI 6/99* (1999) 33 pages.

[Baumgartner and Gottlob, 1999c] R. Baumgartner and G. Gottlob. On the Complexity of Fixpoint 3-valued Semantics for Autoepistemic Logic. *Technical Report DBAI 8/99* (1999) 16 pages.

[Bidoit and Froidevaux, 1991] N. Bidoit and C. Froidevaux. General Logical Databases and Programs: Default Logic Semantics and Stratification. *Information and Computation* 91 (1991) 15-54.

[Brachmann and Levesque, 1984] N. Bidoit and C. Froidevaux. What Makes a Knowledge Base Knowledgeable? A View of Databases from the Knowledge Level. *First EDS Workshop* (1984) 69-78.

[Brewka, 1991] G. Brewka. Cumulative Default Logic: In Defense of Nonmonotonic Inference Rules. *Artificial Intelligence* 50(2) (1991) 183-205.

[Brewka and Gottlob, 1997] G. Brewka and G. Gottlob. Well-Founded Semantics for Default Logic. *Fundamenta Informaticae* XX (1997) 1-16.

[Cadoli, 1992] M. Cadoli. The complexity of model checking for circumscriptive formulae. *Information Processing Letters* 44 (1992) 113-118.

[Cadoli and Schaerf, 1992] M. Cadoli and M. Schaerf. A Survey on Complexity Results for Nonmonotonic Logics. *J. Logic Program.* 17 (1993) 127-160.

[Charniak and Mc Dermott, 1985] E. Charniak and D. Mc Dermott. *Introduction to Artificial Intelligence* Addison-Wesley (1985).

[Christian, 1996] C. Christian. Ein Äquivalent zum ersten Unvollständigkeitssatz. *Sitzungsberichte der Österreichischen Akademie der Wissenschaften, Abt. II 205* (1996) 35-55.

[Dantsin *et al.*, 1997] E. Dantsin, T. Eiter, G. Gottlob and A. Voronkov. Complexity and Expressive Power of Logic Programming. *Proc. of the IEEE Conf. on Computational Complexity '97* (1997).

[Denecker *et al.*, 1998] M. Denecker, V. Marek, M. Truszczyński. Fixpoint 3-valued semantics for autoepistemic logics. *Proc. of AAAI-98* (1998) 840-845.

[Ebbinghaus *et al.*, 1996] H.-D. Ebbinghaus, J. Flum, W. Thomas. *Einführung in die mathematische Logik. 4. Auflage* (1996). Spektrum Akad. Verlag.

[Eiter and Gottlob, 1992] T. Eiter and G. Gottlob. Reasoning with Parsimonious and Moderately Grounded Expansions. *Fundamenta Informaticae* 17 (1992) 31-55.

[Eiter and Gottlob, 1993] T. Eiter and G. Gottlob. Propositional circumscription and extended closed world reasoning are Π_2^P-complete. *Theoretical Computer Science* 114 (1993) 231-245.

[Eiter and Gottlob, 1995] T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Math. and AI* 15 (1995) 289-323.

[Eiter and Gottlob, 1995b] T. Eiter and G. Gottlob. The Complexity of Logic-Based Abduction. *Journal of the ACM* 42 (1995) 3-42.

[Eiter and Veith, 1999] T. Eiter and H. Veith. On the Complexity of Data Disjunctions. *Technical Report DBAI-TR-99-29* (1999).

[Ben-Eliyahu and Dechter, 1991] R. Ben-Eliyahu and R. Dechter. Default logic, propositional logic and constraints. *Proc. of AAAI-91* (1991) 379-385.

[Fagin, 1974] R. Fagin. Generalized first-order spectra and polynomial recognizable sets. *Complexity of Computation, ed. by R.M. Karp* (1974) 43-73.

[Garey and Johnson, 1979] M. Garey and D. S. Johnson. Computers and Intractability: *A Guide to the theory of* $\mathbf{NP}$*-completeness* (1979) W.H. Freeman.

[Gelfond *et al.*, 1991] M. Gelfond, H. Przymusińska, V. Lifschitz and M. Truszczyński. Disjunctive defaults. *Proc. of KR-2* (1991).

[Gogic *et al.*, 1995] G. Gogic, H. Kautz, C. Papadimitriou, B. Selman. The Comparative Linguistics of Knowledge Representation. *Proc. of IJCAI'95* (1995).

[Gottlob, 1992] G. Gottlob. Complexity Results for Nonmonotonic Logics. *J. Logic and Comput.* 2(3) (1992) 397-425.

[Gottlob, 1995] G. Gottlob. The Complexity of Default Reasoning under the Stationary Fixed Point Semantics. *Inform. and Comput.* 121 (1995) 81-92.

[Gottlob, 1995b] G. Gottlob. Translating Default Logic into Standard Autoepistemic Logic. *J. of the ACM* 42/4 (1995) 712-740.

[Gottlob, 1995c] G. Gottlob. NP-Trees and Carnap's Modal Logic. *Journal of the ACM* 42/2 (1995) 421-457.

[Gottlob, 1997] G. Gottlob. Review of a Carnapian Extension of S5. *Logic at Work, ed. Ewa Orlowska* (1999). Physica-Verlag.

[Gottlob and Mingyi, 1994] G. Gottlob and Z. Mingyi. Cumulative default logic: finite characterization, algorithms and complexity *Artificial Intelligence* 69 (1994) 329-345.

[Gottlob and Truszczyński, 1996] G. Gottlob and M. Truszczyński. Approximating the Stable Model Semantics is Hard. *Fundamenta Informaticae* 28 (1996) 123-129.

[Halpern and Moses, 1992] J.Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence Journal* 54 (1992) 311-379.

[Halpern and Vardi, 1991] J. Y. Halpern and M. Y. Vardi. Model Checking vs. Theorem Proving: A Manifesto. *Proc. of KR-2* (1991) 325-333.

[Hoek, 1993] W. Hoek. Systems for knowledge and belief. *Journal of Logic and Computation* 3 (1993) 81-122.

[Hopcroft and Ullman, 1979] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation* (1979). Addison-Wesley.

[Imielinski and Vadaparty, 1989] T. Imielinski and K. Vadaparty. Complexity of query processing in databases with or-objects. *Proc. of PODS'89* (1989) 51-65.

[Inoue and Sakama, 1993] K. Inoue and C. Sakama. Relating Disjunctive Logic Programs to Default Theories. *Logic Programming and Non-monotonic Reasoning* (1993) 266-282.

[Inoue and Sakama, 1997] K. Inoue and C. Sakama. On Positive Occurrences of Negation as Failure. *Proc. of KR-97* (1997).

[Janhunen, 1998] T. Janhunen. On the Intertranslatability of Autepistemic, Default and Priority Logics, and Parallel Circumscription. *Proc. of JELIA'98* (1998).

[Kautz and Selman, 1991] H. A. Kautz and B. Selman. Hard Problems for Simple Default Logics. *Artificial Intelligence* 49 (1991) 243-279.

[Konolige, 1988] K. Konolige. On the relationship between default and autoepistemic logic. *Artificial Intelligence Journal* 35 (1988) 343-382.

[Kraus *et al.*, 1990] S. Kraus, D. Lehmann, M. Magidor. Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelligence Journal* 44 (1990) 167-207.

[Lenzen, 1980] W. Lenzen. *Glauben, Wissen und Wahrscheinlichkeit. Systeme der epistemischen Logik*. Springer (1980).

[Liberatore and Schaerf, 1998] P. Liberatore and M. Schaerf. The Complexity of Model Checking for Propositional Default Logics. *Proc. of ECAI'98* (1998).

[Makinson, 1989] D. Makinson. General Theory of Cumulative Inference. *Non-Monotonic Reasoning, ed. Reinfrank et al.* (1989). Springer Verlag.

[Marek and Truszczyński, 1989] W. Marek and M. Truszczyński. Relating Autoepistemic and Default Logic. *Proc. of KR-1* (1990).

[Marek and Truszczyński, 1991] W. Marek and M. Truszczyński. Autoepistemic Logic. *Journal of the ACM* 38 (1991) 588-619.

[Marek and Truszczyński, 1993] W. Marek and M. Truszczyński. *Non-monotonic Logics – Context Dependent Reasoning* (1993). Springer Verlag.

[Mc Carthy, 1980] J. Mc Carthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence Journal* 13 (27-39).

[Mc Dermott and Doyle, 1980] D. Mc Dermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence* 13 (1980) 41-72.

[Mc Dermott and Doyle, 1982] D. Mc Dermott and J. Doyle. Non-monotonic logic II: nonmonotonic modal theories. *Journal of the ACM* 29 (1982) 33-57.

[Meyer and Hoek, 1995] J.C. Meyer and W. v.d.Hoek. *Epistemic logics for AI and Computer Science*. Cambridge University Press (1995).

[Moore, 1985] R. Moore. Semantical Considerations on nonmonotonic logic *Artificial Intelligence* 25 (1985) 75-94.

[Moore, 1990] R. Moore. Possible-world semantics for autoepistemic logic. *Readings on nonmonotonic reasoning, ed. M. Ginsberg* (1990) 137-142.

[Niemelä, 1991] I. Niemelä. Towards Automatic Autoepistemic Reasoning. In *Logics in AI* (1991) 428-443. Springer Verlag.

[Oberschelp, 1993] A. Oberschelp. *Rekursionstheorie* BI (1993).

[Papadimitriou and Sideri, 1992] C. H. Papadimitriou and M. Sideri. On Finding Extensions of Default Theories. *ICDT '92* (1992) 276-281. Springer LNCS 646.

[Papadimitriou, 1993] Christos H. Papadimitriou. *Computational Complexity* (1993). Addison Wesley.

[Przymusinska and Przymusinski, 1995] H. Przymusinska and T. Przymusinski. Stationary default extensions. *Fund. Informat.* 21 (1994) 67-87.

[Reiter, 1978] R. Reiter. On closed world databases. *Logic and Databases, ed. H. Gallaire and J. Minker* (1978) 55-76.

[Reiter, 1980] R. Reiter. A logic for Default Reasoning. *Artificial Intelligence* 13 (1980) 81-132.

[Rosati, 1999] R. Rosati. Model checking for non-monotonic logics: algorithms and complexity. *Proc. of IJCAI-99* (1999) 76-81.

[Russell and Norvig, 1995] S. Russell, P. Norvig, *Artificial Intelligence: A modern approach* (1995). Prentice Hall International Editions.

[Schwarz and Truszczyński, 1992] G. Schwarz and M. Truszczyński. Nonmonotonic reasoning is sometimes easier. *Technical Report 213-92, Dpt. of Computer Science, University of Kentucky* (1992)

[Selman, 1990] B. Selman. Tractable Default Reasoning. *Doctoral Theses. University of Toronto* (1990).

[Selman and Kautz, 1990] B. Selman and H. A. Kautz. Model-preference default theories. *Artificial Intelligence* 45 (1990) 287-322.

[Stillman, 1990] J. Stillman. It's not my Default: The Complexity of Membership Problems in Restricted Propositional Default Logic. *Proc. of AAAI-90* (1990) 571-578.

[Stillman, 1992] J. Stillman. The Complexity of Propositional Default Logics. *Proc. of AAAI-92* (1992) 794-799.

[Turing, 1937] A. Turing. On Computable Numbers, with an application to the Entscheidungsproblem. *Proc. of the London Math. Society 42/43* (1937). (1937).

[Turner, 1985] R. Turner. *Logics for Artificial Intelligence* (1985). Horwood.

[Veith, 1996] H. Veith. Succinct Representation and Leaf Languages. *Proc. of the 11th IEEE Conf. on Computational Complexity* (1996) 118-126.

[Wagner, 1990] K. Wagner. Bounded query classes. *SIAM J. Comput.* 19(5) (1990) 833-846.