Chapter 1

Introduction

Computational complexity is an area of theoretical computer science in which one aims to classify a given problem with respect to its worst case complexity measured in required computation time and space. Here a completeness result is the most satisfying answer as, informally, it states that the problem's complexity is fully classified. Now such a result may prove that the problem will always stay intractable and therefore forbids the existence of a polynomial time algorithm. One of the possible approaches to overcome this fact is the restriction of the problem with the hope of getting a faster algorithm for the fragment of this problem.

Now suppose you want to visit r of your relatives in one big journey. As you are free from work for only one week you are interested whether there exists an efficient route in at most one week duration and how it would look like. Thus we consider two different kind of questions. On the one hand there is a decision problem to which we can answer with simply *yes* or *no*. On the other hand we want to compute one optimal solution. Without doubt, knowing an optimal solution implies answering the decision problem. Vice versa, it is not clear if this is possible. Visualizing our situation in a graph of nodes (one for each relative and one for you) having edges between every pair of nodes, and edge labels with a distance or travel duration. The naïve approach computes all routes and selects one of the best. The computational effort of this algorithm measured in runtime is limited by the factorial of r, that is, in $r^{O(r)}$ many steps. Having about twenty relatives and computing one billion routes in one second would still need approximately $2 \cdot 10^8$ years to finish the computation. Further, more computation power would not lower the waiting time significantly as the problem exhibits exponential runtime. Thus we either need to find a better algorithm which uses some intelligent approach in deducing one of the desired routes or we simplify the problem by making several restrictions[1]. This could be forcing the triangle inequality to hold, disallowing asymmetric paths, or, e.g., forbidding several connections between some nodes (possibly one cannot directly travel from city x to city y). These approaches may involve understanding which parts of the problem make it inherently hard to solve.

Another promising approach is a transfer to propositional logic which enlarges the field of possible applicable algorithms. The most prominent open problem in theoretical computer science is the P-NP-problem which essentially is the question whether there exists an algorithm running in polynomial time solving the question from above, or

[1]Other approaches that will not be discussed in detail are approximation algorithms (see, for instance, [ACG+99]), or randomized algorithms which have an error property connected to wrong answers (see [MR95] for more information about this topic).

equivalently, deciding the satisfiability of a propositional formula. At present such an algorithm is not known to exist. But propositional logic has been proven itself to be a very powerful tool for encoding several difficult problems into the satisfiability problem SAT. By this property many different algorithms have been exhibited. Another very interesting property of the problem SAT is that one can efficiently verify solutions of instances, that is, given assignments to the variables one can check in polynomial time if this is indeed a correct solution. This fact is the main property of problems in the class NP (which stands for nondeterministic polynomial time). Further, SAT has received great attention because a polynomial time algorithm for SAT implies polynomial time algorithms for every problem of the class NP [Coo71b, Lev73]. Therefore several restrictions of SAT have been investigated where k-SAT comprises one surprising characteristic. If we restrict propositional formulae to conjunctive normal form, i.e., any formula can be written as conjunctions of disjunctive clauses containing only k literals (which are variables or their negations), then for $k = 2$ the problem becomes tractable whereas for $k = 3$ it is intractable unless P equals NP.

Furthermore, an approach used by H. Lewis in 1979 is the origin of an auspicious technique for understanding the hardness of a problem involving propositional logic [Lew79]: H. Lewis used a tool investigated by E. Post 1941 [Pos41], which is a lattice of all Boolean functions wherefore it is also called *Post's lattice*. The main application of this tool is to fragment any problem which inherently uses propositional connectives into all parts by means of any possible set of Boolean functions. Thereby H. Lewis was able to connect the intractability (unless $\mathsf{P} = \mathsf{NP}$) of SAT to some specific Boolean function, i.e., the negation of the implication function $\nrightarrow$. Thus whenever a formula is composed of Boolean functions that are in some way able to express $\nrightarrow$, one works with an instance of the intractable version of SAT. Consequently if we would be able to write a propositional formula for the travel problem from above avoiding $\nrightarrow$ (and functions that can express $\nrightarrow$ as well) then we would have a polynomial time algorithm for our recent case (unless the constructed formula is of super-polynomial size). Unfortunately it is not known whether such a formula exists as this would answer the P-NP-question as well.

The motivation of this thesis strictly encompasses this question. Which functions make a decision problem hard to solve? Why does the tractability of some problem depend on the availability of some Boolean function or operator? Here we will investigate several powerful extensions of propositional logic which are closely connected to modal logic.

1.1 Modal Logic

The connection of propositional logic to computer programs requires an adequate model where Kripke structures have been proven of great use. These structures are essentially directed node-labeled graphs simulating the behavior of a computer program in the means of different program states. Informally, modal logic is the extension of propositional logic by a new operator $\Diamond$ enabling formulae to express transitions between program states. 1918, modal logic has been firstly introduced by C. I. Lewis [Lew18] and has become very popular since the 1960s [Kri63, HC68] and until now [Gol06, BvW06]. Furthermore a complete study with respect to the Boolean fragments of modal logic has been done by Hemaspaandra et al. [HSS08] recently.

1.1.1 Temporal Logic

Enriching modal logic with concepts to interact more densely with computer programs leads to the field of temporal logics which have been introduced by A. N. Prior in 1957 who has been called "the founding father of temporal logic" by the Danish Centre for Philosophy and Science Studies [Pri57, Pri67, Aal11]. From 1971 to 1986 significant effort by Pnueli, Emerson, Halpern, and Clarke resulted in the definition of the linear time logic LTL and the computation tree logics CTL* and CTL [Pnu77, CE81, QS82, EH86]. These logics have been invented to be of great benefit in the process of software engineering for verifying non-terminating programs. Describing specifications through formulae results in an evaluation of the written programs which are modeled by the Kripke structures as explained above. In the course of time, temporal logics emerged as being useful with major relevance for practical experience [VS85a]. In this context the *model checking problem* and the *satisfiability problem* of these logics are of great interest. For the model checking problem one asks if a given formula is satisfied in a given world of a given Kripke structure. Thus essentially the question whether a computer program fulfills its specification. For the satisfiability problem the question is whether a Kripke structure (containing a world) exists which satisfies a given formula. Hence we occupy with the question if there exists a computer program fulfilling this specification. In other words we ask some kind of consistency question with respect to a specification modeled by a temporal logic formula.

These two problems with respect to the three logics have been completely classified with respect to their complexity and without making any restrictions to the problems in [FL79, VS85a, CES86, Eme90, EJ00]. A comparison of these results bare a tremendous gap between model checking and satisfiability of CTL: a polynomial time model checking algorithm (and also hardness for P) is accompanied by an exponential time algorithm for satisfiability with the proof that there can be no better one. For the other two temporal logics the gap between the complexity of satisfiability and model checking is similar huge but both problems are intractable unless $\mathsf{P} = \mathsf{PSPACE}$. Model checking in both logics is complete for polynomial space whereas satisfiability remains PSPACE-complete for LTL and jumps up to double exponential time for CTL*. Whilst for LTL the classification of all Boolean and modal operator fragments has been achieved by Bauland et al. [BMS$^+$11, BSS$^+$09], the fragments of the computation tree logics are still open and will be investigated in Chapter 3.

1.1.2 Description Logic

The concept of databases influenced the development of description logics significantly. The origin of research has been considered to have started with the work of Brachman and Levesque in 1984 [BL84], whilst some principles of these logics go back to semantic networks and the KL-ONE system [BS85]. An extensive introduction to this field of logics has been written by Baader et al. [BCM$^+$03]. Description logics (DLs) are usually defined as extension of the logic $\mathcal{AL}$ however some smaller fragments of $\mathcal{AL}$ recently received attention in the research community, namely the $\mathcal{FL}$- and $\mathcal{EL}$-family [Baa03, Bra04a, BBL05a, BBL08]. DLs are widely used in the semantic web in terms of the web ontology language OWL 2 [MPSP09]. Several terms from the web ontology language are synonyms of terms in the family of description logics and connect these two areas very closely. Further, DLs are used in the codification of medical knowledge by ontologies whose definition is explained below.

Regarding the connection to databases the two main formalisms in DLs are terminology and assertional boxes which are abbreviated by the terms *TBox* and *ABox*. The union of both is referred to as an *ontology*. An ABox is essentially a relational database with its pairs whereas the TBox expresses constraints for the database in form of rules (*axioms*, or without restrictions, *general concept inclusions* GCI). These rules are pairs of formulae which are composed of the functions *and* $\sqcap$, *or* $\sqcup$, and *not* $\neg$ as well as the *role quantifiers* which can be *existential* $\exists R$ or *universal* $\forall R$ for some role R. The different kinds of used symbols for expressing the Boolean functions base on the origin of the logics which was disjoined from modal logic as described above. However, the connection to first order logic is immediate but DLs have more efficient decision problems. With respect to the ability to express arbitrary Boolean functions $\mathcal{ALC}$ can be considered as best suitable for the use with Post's lattice due to the availability of $\wedge$, $\vee$, and $\neg$ in this logic. The remarkable part for the decision problem with respect to TBoxes is the following. A TBox $\mathcal{T}$ is said to be consistent for the corresponding model if and only if every axiom in $\mathcal{T}$ is consistent with every world in the model. By virtue of this definition this problem is already complete for exponential time [BBL05a, Hof05] and thus prohibits the existence of a polynomial time algorithm.

More formally the decision problems of interest for DLs are

- the satisfiability problem of TBoxes,
- the concept satisfiability problem with respect to a given TBox,
- the satisfiability problem of an ontology, and
- the subsumption problem with respect to a given TBox.

The latter problem is a special kind of the *implication problem* in the sense of description logics. Further, a method similar to logical deduction which is called *structural comparison* has been deployed but not proven itself to always state correct results. Lacking the completeness it has been shown to be weaker than logical subsumption recently [NB03]. Thus subsumption can be seen as one of the central problems in the area of DLs.

The unrestricted versions of the aforementioned decision problems have been classified previously by their correspondence to propositional dynamic logics [Pra78, VW86, DM00]. To the best of the author's knowledge a complete classification of these problems with respect to all possible Boolean functions has not been done yet and will be the topic of Chapter 4. Especially the study of less commonly used operators as the negation of implication $\nrightarrow$ or the binary exclusive-or $\oplus$ will give an insight to the influence of Boolean functions on tractability.

The classification of all Boolean function and operator fragments of the concept satisfiability problem for the description logic $\mathscr{ALC}$ immediately follows from the work of Hemaspaandra et al. [HSS08] due to the equivalence to modal logic. They obtained a trichotomy which comprises of complexity degrees from contained in P, through coNP-complete to PSPACE-complete fragments.

1.1.3 Post's Lattice

As motivated above, our approach is to follow Lewis' technique for getting the most fine granulated and complete classification with respect to all possible Boolean functions and operators for each of the decision problems which have been mentioned above. Previously this approach has been followed extensively in the areas of constraint satisfaction [Bau07, Sch07, Sch08], nonmonotonic logics [BMTV09a, Tho09, CMTV10, Tho10], modal and propositional logics [Rei01, HSS08, BMTV09b], abduction, and argumentation [CST10, CSTW10]. These studies have one goal in common. They want to understand which Boolean functions play the role of $\nrightarrow$ in the therein studied extended propositional logics. This is the main goal in this thesis as well.

More formally let B be a finite set of Boolean functions. Then we define the *clone* $[B]$ of B as the set of all Boolean functions which can be constructed by arbitrary composition and projection of functions from B. B is called a *base* of $[B]$ in this context. Post constructed the infinite lattice comprising of all possible clones and proved the existence of a finite base for each of these clones. Usually one aims to achieve a complete classification with respect to Post's lattice. Therefore one needs to overcome the infinity within the lattice by stating matching upper and lower bounds ranging from both ends of the infinite chains in the lattice (see Figure 2.2 on page 16). By definition of the lattice those results state completeness results for any decision problem fragment with respect to each clone within the infinite chain.

1.2 Results

In the first part of Chapter 3 we visit the satisfiability problem of CTL and classify the temporal operator and Boolean fragments. There we show how they form a trichotomy ranging through NP-, PSPACE-, and EXP-complete cases (see Figure 3.6 on page 50) whereas the Boolean fragments, without respect to the temporal operators, lead to TC^0-, NC^1-, and EXP-complete cases (see Figure 3.7 on page 51). Section 3.1.4 aims to describe the problems occurring when working with affine cases which resisted getting fully classified for this decision problem in temporal logic. Furthermore, we will visit extensions of the temporal logics CTL and CTL*, particularly, *(i)* CTL$^+$ which behaves similarly as CTL-SAT (and we also classify in parallel the fragment LTL$^+$) and *(ii)* the fairness extension ECTL where all relevant operator fragments are either PSPACE- or EXP-complete.

The second part of this chapter covers the research on the model checking problems of CTL, CTL*, and the same extensions as above. As the model checking problem for CTL is tractable, and in fact P-complete, we will follow an approach by Sistla and Clarke: we investigate three different kinds of fragments in terms of allowed negation symbols, starting with monotone, atomic negations only, and positive fragments (for an explicit definition see page 52). The latter one are fragments where operators (not Boolean functions) may not occur in the scope of a negation. Surprisingly, we will show that these three problems are actually computationally equivalent (see Theorem 3.24), and are NC^1-complete if no temporal operator is available, LOGCFL-complete if either we have a non-empty subset of $\{\mathrm{EX}, \mathrm{EF}\}$ or $\{\mathrm{AX}, \mathrm{AG}\}$, and P-complete otherwise. Hence, most fragments of the CTL model checking problem are inherently sequential (see Figure 3.11 on page 73). Thus there is no way to develop parallel algorithms for these cases. While ECTL behaves analogously to CTL, the other extensions exhibit different properties, and their classifications range through six different complexity classes (see Theorem 3.27 and Corollary 3.28). As a starting point for further research, we will achieve a classification for all operator/quantifier fragments of cardinality at most two. We will show how fragments which are easy for this problem, use some CTL-algorithms, and how intractable cases depict parallels to the model checking problem of LTL, in Theorem 3.29.

Finally in Chapter 4 we turn towards the area of description logics, an extension which is widely used by the semantic web community. There we visit all fragments with respect to the possible subsets of the quantifiers $\exists$ and $\forall$, and all Boolean clones. Given a single terminology $\mathcal{T}$ using both quantifiers, we will see how the connected satisfiability problem is either EXP-complete or trivial, i.e., always having satisfiable terminologies. The latter holds if and only if only c-reproducing functions for $c \in \{\top, \bot\}$ are used in $\mathcal{T}$. Allowing only one quantifier turns the fragments which use conjunctions or disjunctions tractable, i.e., P-complete. Without any quantifiers we reach NLOGSPACE-completeness for the fragments using only unary functions. The classification for the decision problems asking about the satisfiability of a concept with respect to a terminology behaves similarly but with two exceptions. First, the $\bot$-reproducing cases are not trivial any longer. Secondly, the lower complexity bounds can be improved to hold without using the constant $\top$. An overview of the results is depicted in Table 4.1 on page 94.

Lastly, in Section 4.2 we classify the implication problem adjusted to description logics, which is the subsumption problem with respect to all quantifier sets and Boolean function sets. There we will show that whenever we are able to express one of the constants besides having access to all quantifiers, the complexity of the fragment remains EXP-complete. By the use of only one quantifier the problem becomes tractable (P-complete) if either conjunctions or disjunctions are allowed—depending on which quantifier is existent. Disallowing quantifiers in general leads to a similar classification as previously has been achieved by Beyersdorff et al. for the propositional implication problem [BMTV09b] with a slight exception for the affine cases involving the function exclusive-or $\oplus$. The complete arrangement in Post's lattice is visualized in Figures 4.1 and 4.2 on pages 105 and 106.

1.3 Publications

Sections 3.1.1 to 3.1.3 have been previously published in [MMTV09] but the proof of Theorem 3.4 (1.) is new. Sections 3.1.4 and 3.1.5 contain unpublished results about the affine cases and extensions. Section 3.2 contains published results from [BMM$^+$11] but Section 3.2.3 contains unpublished results about fragments of the model checking problem for CTL*. Section 4.1 has been published in [MS11a, MS11b]. Section 4.2 contains new and unpublished results.