Chapter 1

INTRODUCTION

1.1 Understanding Dependence

In order to understand what dependence means, it is necessary to know where dependence works on. To make statements about dependence of certain parameters in a situation (that can e.g. be a measurement in experimental science or some configuration in a computational system) one has to compare at least two or more situations with respect to a property. Clearly, one can not make a statement about dependence in a computational system by only knowing one single configuration. If for example we talk about the computational path of any machine model, a situation can be the configuration of the machine at a certain time. Here, a configuration has certain parameters. We say t is dependent on $t_1, \ldots, t_n$ if in every situation $t_1, \ldots, t_n$ has the same value it follows that also t has the same value.

In this thesis we call t a *variable* or *term* according to the logic and a situation is called an *assignment*, hence an assignment is a function that maps a variables to a value. A set of assignments is called a *team*. Statements about dependence are expressed with the *dependence atom* denoted by

$$=(t_1, \ldots, t_n, t)$$

if we want to express that the value of t only depends on the values of $t_1, \ldots, t_n$.

Trivially, every statement about dependence holds, if the team is small enough (at least every dependence holds on singleton teams). This is an interesting property that is captured by the dependence atom. Thus, an obvious question is, whether a given formula expressing any property of

dependence holds on a model (*model checking*), or given a formula verifying whether there exists a model that makes the formula true (*satisfiability*). These are problems that are considered on different types and extensions of dependence logics.

In most classical logics there are only assignments to variables that map one single value to a variable. In dependence logic we have to be able to formulate several situations, thus we have to talk about several assignments of variables. A semantics that handles different assignments is called *team semantics*. This semantics was introduced firstly by Hodges in [Hod97a] and [Hod97b]. It works as follows: If we want to evaluate a formula φ on a team T we have to evaluate it on the team as a whole, thus the assignments can conflict each other. That means in dependence logic (and also in every extension of it studied in this thesis) a formula φ can be true on every single assignment but false if it is evaluated on several assignments of the team. In fact, team semantics is a nonclassical extension to logic, but it is natural since it even appears in our everyday life as the following example shows.

Example 1.1.1. *Assume one organizes a meeting of five people and only the organizer knows the exact number of people invited to the meeting. At the beginning of the meeting the organizer asks "Are we complete?". No one can answer this question correctly by himself because no one except the organizer knows whether there is still a person missing.*

Definitely, this question is evaluated in team semantics, because we have to evaluate the question on the team as a whole. After all the organizer did not want to know whether any *single person* is complete by himself or herself, he wanted to know, whether the team is complete. Thus, the answer can only be given, by someone that has information about the whole group of people. This example illustrates the naturality of team semantics.

We go one step further. In the next example we illustrate that statements about dependence seem very likely in this context.

Example 1.1.2. *We could think of a response to the organizers question like "If person A joins the meeting depends on whether person B joins the meeting". Note, that this gives no information about* how *it depends, it only states that there* is *a dependence at all. Thus, two scenarios are possible.*

a) Person A and B like *each other, that means, person A only joins the meeting, if person B joins the meeting as well.*

b) Person A and B do not like *each other, i.e. A only joins the meeting, if B does not.*

1.2 A Historical Outline on Dependence Logic

Dependence and independence is a topic of broad interest in every time of human history. Understanding dependencies of another party such as another nation or simply another person in many cases leads to a better understanding of the way of acting of this other party. In negotiations it is of major importance to have knowledge about the way the opponent acts. This is in many cases based on dependencies the opponent is inflicted with. In a poker game for example, it is always a good strategy to play unpredictable. This is only achieved when knowing about my own dependence of my strategy. If I only play for high stakes while having good cards and risk minimizing while having bad cards makes the strategy predictable. This is however the first idea to optimize the strategy of course but in case you have a strong opponent the strategy gets obvious. Thus, one has to vary the own strategy on one hand and on the other hand one has to be able to predict the way of the opponents acting, basing on e.g. dependencies.

Since dependence is such an important property there have been several approaches to formalize it. Those approaches are necessary in order to make general statements about dependence, that means what is expressible in a certain logic with dependence and how difficult is it to express it or to check if a statement about dependence is true. In the early 1960's there is an approach of Henkin. He considered the dependence of choosing values for quantified variables. Usually e.g. in the following first order logic ($\mathcal{FO}$) formulae

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 \varphi$$

we have, that the value of, e.g., y_2 can be chosen dependent on the value of x_1 or y_1. In other words y_2 "knows" these values. In order to make this kind of quantification independent, Henkin introduced a new kind of quantifier in [Hen61]. This quantifier uses independent lines written as a matrix to quantify variables with only a partial dependence:

$$\begin{pmatrix} \forall x_1 & \exists y_1 \\ \forall x_2 & \exists y_2 \end{pmatrix} \varphi. \tag{1.1}$$

In (1.1) we have that y_1 solely depends on x_1 and y_2 only depends on x_2. Thus we still have the partial dependence from right to left, but the lines are independent of each other.

In 1989 Hintikka and Sandu introduced independence-friendly logic ($\mathcal{IF}$). While quantifying this logic allows to explicitly state on which variables a certain variable is independent on. The formula

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 / \{x_1, y_1\} \varphi$$

states that the value for y_2 can be chosen independently of x_1 and y_1. The variable y_2 depends only on x_2 here. That means while choosing the value for y_2, the values for x_1 and y_1 are not known to y_2.

Recently, in 2007 Jouko Väänänen introduced dependence logic [Vää07]. Here dependence is explicitly stated but in contrast to dependence-friendly logic dependence here is expressed inside the formula by the dependence atom which is a new kind of atomic formula. Dependencies can be expressed more general as in $\mathcal{FO}$. Namely one can define a function that gives the dependence between variables without defining the function explicitly. Dependence logic here behaves like existential quantifying a function or a relation in existential second order logic $\mathcal{ESO}$, respectively.

It has been shown that $\mathcal{IF}$ and $\mathcal{D}$ have the same expressive power as $\Sigma^1_1 \equiv \mathcal{ESO}$. In terms of descriptive complexity both $\mathcal{IF}$ and $\mathcal{D}$ capture the complexity class NP. This means that every NP-complete problem is definable in $\mathcal{D}$ and $\mathcal{IF}$.

In 2008 a modal variant of dependence logic ($\mathcal{MDL}$) was introduced by Väänänen in [Vää08]. Merlijn Sevenster showed that satisfiability is NEXPTIME-complete for $\mathcal{MDL}$ in general [Sev09]. This result was refined by Peter Lohmann and Heribert Vollmer for sublogics of $\mathcal{MDL}$ [LV10]. These fragments were built by restricting the set of operators used in the formulae. In 2011 E. and Lohmann gave an almost complete analysis of the model checking for $\mathcal{MDL}$ and its operator fragments [EL12].

The framework of dependence logic turned out to be flexible enough to introduce a new kind of implication, namely *intuitionistic implication*. This was introduced by Jouko Väänänen and Samson Abramsky [AV09]. Fan Yang showed [Yan13] that dependence logic with intuitionistic implication is as equivalent in the sense of expressive power as full second order logic. In other words by adding intuitionistic implication we gain universal second order quantification.

The aim of this thesis is to give new generalizations and extensions to first order dependence logic as well as for the modal variant of dependence logic. We investigate the complexity of several well known problems in computer science on these logics and also compare the logics in the sense of expressive power.

1.3 Dependence Logic with Quantifier Extensions

Classically in logic as well as in dependence logic there is universal and existential quantification. There are different kinds of those quantifiers. First order and second order existential quantifiers are used in order to express that, e.g., a certain property holds in a certain case (existential quantification), or in every case (universal quantification). But in many fields there are statements made about most cases or statements that use counting. Therefore one could think about statements made about the parity of a quantified variable, as for example

There is an even number of primes below 10.

On the other hand, one could think of statements about majorities like

For most natural numbers it holds that they are not primes.

On one hand we give an extension of first order dependence logic that can quantify the majority of functions that extend a team with a quantifier called M and on the other hand, we give an extension of dependence logic that counts parity of team extending functions. Since $\mathcal{D}$ itself is equivalent to existential second order logic ($\mathcal{ESO}$) we compare those quantifier extensions to fragments of second order logic ($\mathcal{SO}$). It turns out, that dependence logic extended by a majority quantifier M (denoted by $\mathcal{D}(\mathsf{M})$) is equivalent to second order logic extended by a quantifier Most (denoted by $\mathcal{SO}(\mathsf{Most})$), which is a quantifier that holds if for most relations the formula is true. It has been well studied in [Kon09] that $\mathcal{SO}(\mathsf{Most})$ is equivalent to its first order counterpart $\mathcal{FO}(\mathsf{Most})$ (since the notion of second order existential quantification can be simulated by the Most quantifier) and thus equivalent to the counting hierarchy CH which was firstly introduced by Wagner [Wag86]. Hence, the quantifier M adds counting capabilities to dependence logic.

Counting is an interesting and widely studied property in logic as well as in complexity theory [HO98, Kon09, KN11]. Thus understanding counting is very fundamental. Adding counting capabilities has deserved a lot of attention in theoretical computer science.

Now consider the circuit class TC^0, which is defined by a polynomial-sized circuits with constant depth plus majority gates [Vol99]. There are strict lower bound separations inside TC^0 but not above. It is known that TC^0 can be separated from the second level of the exponential time hierarchy, but still there is no separation to any lower class known. Thus a very