1. Introduction and historical background

ALTERNATION AND THE ACKERMANN CASE OF THE DECISION PROBLEM [1]

by Martin FÜRER [2]

ABSTRACT. The Ackermann prefix class is the set of all formulas of predicate calculus (first order logic without function symbols) with quantifier prefix $\exists \ldots \exists\forall\exists \ldots \exists$. This is one of the few prefix classes for which satisfiability is decidable. Lower bounds for the computational complexity of this decision problem and the $\forall\exists$ sub-problem are presented. The tool to get the main result is the alternating Turing machine. An introduction to alternating Turing machines is given, because they are probably the most remarkable new subject of automata theory, and are well known only to computer scientists.

1. INTRODUCTION AND HISTORICAL BACKGROUND

From the beginning of this century to the thirties, the problem of deciding universal validity of first order formulas, moved slowly to the center of interest of mathematical logic. Especially Hilbert considered it to be a fundamental problem. As it seemed too hard to solve the decision problem (or Entscheidungsproblem) in general, the main approach was to restrict the class of formulas (for which a decision algorithm should work) by very simple syntactic criteria. An earlier example of this kind of restriction was the decidability result of Löwenheim [29] for the monadic (only unary predicate symbols) predicate calculus. Later the main such criterion was the form of the quantifier sequence for formulas in prenex form (see [14], [28], [43] for other syntactically defined classes). There is a duality between universal validity and satisfiability. A closed formula (i.e. a formula of predicate calculus without free variables) is universally valid, iff its negation is not satisfiable. Around 1930 the decidability of the satisfiability

---

[1]) Presented at the *Symposium über Logik und Algorithmik* in honour of Ernst SPECKER, Zürich, February 1980.

[2]) This work was supported by the British Science Research Council and by the Swiss National Fonds.

problem for closed formulas with the following prefixes has been shown ($\exists^*$ ($\forall^*$) stands for finite sequences of existential (universal) quantifiers).

Bernays and Schönfinkel [7]	$\exists^* \forall^*$
Ackermann [1]	$\exists^* \forall\exists^*$
Gödel [16, 17], Kalmár [23], Schütte [33, 34]	$\exists^* \forall\forall\exists^*$

Independent of Ackermann, Skolem [39] solved $\forall\exists^*$. Bernays and Schönfinkel [7] solved the $\forall\exists$ case before.

These results are for predicate calculus without equality. But for the first two cases, the methods can be extended to predicate calculus with equality (see [2] or [14]). Dreben has discovered that the same extension is not obvious in the Gödel-Kalmár-Schütte case. Dreben's conjecture that this case might be more difficult with equality was supported by Aanderaa and Goldfarb with various examples. Recently Goldfarb [18] (see also [14]) has shown this case not to be primitive recursive. It is not known, if it is decidable.

It took a long time to prove that for predicate calculus without equality the subclasses of $\exists^* \forall^*$ and $\exists^* \forall\forall\exists^*$ are the only decidable prefix classes. The major steps in this direction were: Church [12] showed that the predicate calculus is undecidable. Turing [41] gave a more direct proof of this fact using Turing machines. So in the thirties, it was known that not all prefix classes allow an algorithmic solution. Undecidability results have been obtained by several researchers for prefix classes containing prefixes of arbitrary length (see [2, p. 61]). The gap was narrowed by Surányi [40] who showed that the prefix classes obtained from $\forall\forall\exists \wedge \forall\forall\forall$ are undecidable. Here the refined classification according to conjunctions of formulas in prenex form is used. The formulas of the class $\forall\forall\exists \wedge \forall\forall\forall$ allow a straightforward transformation to formulas of both the prefix classes $\forall\forall\forall\exists$ and $\forall\forall\exists\forall$. With an elegant proof Büchi [8] showed the undecidability for $\exists \wedge \forall\exists\forall$. Wang [43, 44] has invented several versions of domino problems. They represent an intermediate step between computations and formulas. Infinite computations are technically harder to describe by formulas, than the corresponding domino problems. Using dominoes, Kahr, Moore and Wang [22] got rid of the additional $\exists$ (which seemed necessary to describe a start configuration) and showed the undecidability of $\forall\exists\forall$. With this result, the decidability problem for all prefix classes was solved. The undecidability of $\forall\exists \wedge \forall\forall\forall$ follows as a corollary, and all other undecidability results in the refined classification

follow immediately [8], while all decidable classes are contained in the classes of the form

$$\exists^*\forall^* \wedge \exists^* \forall^* \wedge \ldots \wedge \exists^* \forall^*$$

or

$$\exists^*\forall\forall\exists^* \wedge \exists^*\forall\forall\exists^* \wedge \ldots \wedge \exists^*\forall\forall\exists^* .$$

At the same time, as these purely syntactically defined subclasses of the predicate calculus have been investigated, many decidability and undecidability results for mathematical theories (i.e. for classes of sentences that are not selected primarily from a syntactical point of view) have been obtained.

In recent years many researchers have investigated the computational complexity of decidable theories. But very few have looked at this problem for syntactically (in the above vague sense) defined sets of formulas, except for propositional calculus. One of these few is Kozen [25] who got the surprising result that predicate calculus without negation is *NP*-complete. The other is Lewis [27] who has investigated the computational complexity, just of the (above defined) decidable prefix classes and the monadic predicate calculus. (The latter was investigated before by Rackoff [31].) The work of Asser [4], Mostowski [30], Bennett [5], Jones and Selman [21], and of Christen [11] about spectra is related to this field.

The reason, why the computational complexity of these problems is of interest is not that we would like to know how many hours we have to spend, in order to decide if a certain formula is satisfiable. It is very much the same, as the logicians have not been interested to use their decidability results to decide for many formulas, if they are satisfiable. Nevertheless the decidability problem was considered to be a fundamental question of deep mathematical significance. In the same sense, we claim that the precise asymptotic computational complexity of a natural class of formulas is a fundamental mathematical property. Naturally this does not mean that complexity results are of no importance for the computational practice. But at least some results (mostly huge lower bounds) are not so directly applicable. What they can do, is to improve our understanding of the investigated problem, show connections to other problems, and give us hints for a better understanding of the reasons for the complexity of certain problems, and for the different qualities of complexity.

From a practical computational point of view, the deterministic time complexity is certainly the most important complexity measure. But other measures have been developed, such as nondeterministic and space measures

and certain more complex measures concerning alternating Turing machines. The claim that the computational complexity is an important mathematical notion is supported by the fact that natural problems are in nice complexity classes.

Lewis [27] has given good complexity bounds for the decidable prefix classes of the predicate calculus. The largest gap is in his result about the Ackermann class $\exists^* \forall\exists^*$. Lewis claims an upper deterministic time bound $c^{n/\log n}$. His lower bound is polynominal space. Naturally he conjectures that the lower space bound is the correct bound, as problems containing quantifiers do not tend to have good deterministic time bounds. And in any case, there are very few problems known with good deterministic time bounds. The usual method to prove lower time bounds is to describe Turing machine computations. So just as well a nondeterministic Turing machine can be chosen, yielding even a nondeterministic lower time bound.

But there are a few methods to prove deterministic lower time bounds, using tools of automata theory, namely the auxiliary pushdown automaton of Cook [13], or the auxiliary stack automaton of Ibarra [20], or the alternating versions of them, investigated by Ladner, Lipton and Stockmeyer [26], or the alternating Turing machine. The latter seems to be the most interesting, but so far it has not had too many applications. And most applications are to problems involving games or sequences of quantifiers with an unbounded number of quantifier alternations. We apply alternating Turing machines to get a $c^{n/\log n}$ deterministic lower time bound for the Ackermann case of the decision problem. This is an application of alternating Turing machines in a new field, where it is not obvious that this tool can be successful.

Here is a summary of the rest of this paper. In the next section a short presentation of some notions from logic, and in section three from computational complexity is given. In section four, the alternating Turing machine is introduced.

In section five, a transformation from the full Ackermann class to the monadic Ackermann class is described. This is a good transformation for the classes $\exists^p \forall\exists^*$ (only p existential quantifiers in front of the universal quantifier) with constant p. But for the class $\exists^* \forall\exists^*$ this transformation is via length order $n^2/\log n$ instead of n. It is not clear, if a better transformation exists. Alternating Turing machines can test the satisfiability of $\exists^* \forall\exists^*$ formulas more directly than deterministic Turing machines. They are used here to get the optimal upper bound of Lewis [27] for the monadic case, because with alternating Turing machines a polynomial

space upper bound for the $\exists^* \forall \exists$ subcase is obtained at the same time. It is easy to see that the class $\exists^* \forall$ is *NP*-complete.

Section six contains the main result, namely the $c^{n/\log n}$ lower bound for the $\forall \exists \exists$ case, and also a tight lower bound for the $\forall \exists$ case, as well as some *NP*-complete problems. In the last section are some conclusions.

2. Some notions from logic

The formulas of first order logic (see e.g. Shoenfield [36]) are built from:

— variables $y, x_1, x_2, \ldots z_1, z_2, \ldots$

— function symbols $f, g, f_L, f_R, f_1, f_2, \ldots$
(we use $c, c_1, c_2, \ldots$ for 0-any function symbols, i.e. constants)

— predicate symbols $P, P_1, P_2, \ldots$ (and other capitals)

— auxiliary symbols (,)

— equality symbol $=$

— propositional symbols $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$

— quantifiers $\forall, \exists$

We use $F[x/t]$ to denote the result of the *substitution* of the term t for the variable x in the formula F.

A formula $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n F_0$ with Q_i quantifiers (for $i = 1, \ldots, n$) and F_0 quantifier-free is in *prenex form*. F_0 is called the *matrix* of the formula.

We are investigating decision procedures for formulas of first order logic without equality and without function symbols. But we use the functional form of formulas.

The *functional form* of a formula in prenex form is constructed by repeatedly changing

$\forall x_1 \forall x_2 \ldots \forall x_n \exists y F$ (F may contain quantifiers) to

$\forall x_1 \forall x_2 \ldots \forall x_n F[y/f_i(x_1, \ldots, x_n)]$

using each time a new n-ary function symbol f_i until no more existential quantifiers appear.

A formula is satisfiable, iff its functional form is satisfiable. In addition, both are satisfiable by structures of the same cardinality.