2. Some notions from logic

Objekttyp: **Chapter**

Zeitschrift: **L'Enseignement Mathématique**

Band (Jahr): **27 (1981)**

Heft 1-2: **L'ENSEIGNEMENT MATHÉMATIQUE**

PDF erstellt am: **25.05.2024**

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.

We use α, α' to denote structures. A *structure* α for a first order language L consists of:

— a nonempty set $|\alpha|$ (the universe of α),
— a function $f^\alpha : |\alpha|^n \to |\alpha|$ for each n-ary function symbol f of L, (in particular an individual (= element) c^α of $|\alpha|$ for each constant c of L),
— a predicate $P^\alpha : |\alpha|^n \to \{\text{true, false}\}$ for each n-ary predicate symbol P in L.

f^α and P^α are called interpretations of f and P.

A structure for a language L defines a truth-value for each closed formula (i.e. formula without free variables) of L in the obvious way (see e.g. [36]). A structure α is a *model* of a set of closed formulas, if all the formulas of the set get the value true (i.e. are *valid* in α). A formula F is *satisfiable*, if its negation $\neg F$ is not valid.

Let α be the following structure for a language L without equality:

The universe $|\alpha|$ (the Herbrand universe) is the set of terms built with the function symbols of L (resp. of L together with the constant c, if L contains no constants (= 0-ary function symbols)). Each function symbol f is interpreted by f^α with the property: For each term t, $f^\alpha(t)$ is the term $f(t)$. We call such an α a Herbrand structure. If a formula F (in the language L) is valid in α, then we call α a *Herbrand model* of F.

The following version of the Löwenheim Skolem theorem is very useful for our investigations.

THEOREM. *The functional form of a closed formula without equality is satisfiable iff it has a Herbrand model.* □

This theorem can be proved with the methods developed by Löwenheim [29] and completed as well as simplified by Skolem [38]. The version of Skolem [37] which uses the axiom of choice, has less connections with this theorem. Also in Ackermann [2] and Büchi [8] versions of the above theorem are present. Probably for the first time, Ackermann [1] constructs a kind of Herbrand model, the other authors use natural numbers instead.

3. Some notions from computational complexity

We use one-tape Turing machines and multi-tape Turing machines with a two-way read-only input tape and, if necessary, a one-way write-only output tape. The other tapes are called work tapes. The Turing machine