3. Some notions from computational complexity

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

alphabet Γ and the input alphabet Σ are any finite set of symbols with $\Sigma \subseteq \Gamma$ and $B \in \Gamma - \Sigma$, where B is the blank symbol.

A language L is a set of (finite) words over a finite alphabet. Σ^* is the set of all words over the alphabet Σ.

We write $f(n)$ if we mean the function $f : \mathbf{N} \to \mathbf{R}$. This unprecise notation is standard in computational complexity. In connection with Turing machines n denotes always $|w|$, the length of the input word,

$$f(n) = O(g(n)) \text{ means } \exists c\, \exists n_0\, \forall n \geqslant n_0\, f(n) \leqslant c\, g(n)$$

A language L_1 (over Σ) is *logspace transformable* to a language L_2 (over Σ) *via length order* $g(n)$, if there exists a function $f : \Sigma^* \to \Sigma^*$ such that:

$$f(w) \in L_2 \text{ iff } w \in L_1 \quad \text{for all } w \in \Sigma^*,$$

$|f(w)| = O(g(|w|))$, and there exists a multi-tape Turing machine which computes f, scanning only $O(\log n)$ tape squares of the work tapes.

We use the following complexity classes:

$DTIME(f(n)) = \{L \mid L$ is accepted by a deterministic Turing machine in at most $f(n)$ steps (for all $w \in \Sigma^*$ with $n = |w|)\}$

$NTIME(f(n)) = \{L \mid L$ is accepted by a nondeterministic Turing machine in at most $f(n)$ steps (for all $w \in L$ with $n = |w|)\}$

$DSPACE(f(n)) = \{L \mid L$ is accepted by a deterministic Turing machine using at most $f(n)$ tape squares on each work tape$\}$

$NSPACE(f(n)) = \{L \mid L$ is accepted by a nondeterministic Turing machine using at most $f(n)$ tape squares on each work tape$\}$

$$P = \bigcup_{c,k \in \mathbf{N}} DTIME(c+n^k)$$

$$NP = \bigcup_{c,k \in \mathbf{N}} NTIME(c+n^k)$$

$$POLYSPACE = \bigcup_{c,k \in \mathbf{N}} DSPACE(c+n^k) = \bigcup_{c,k \in \mathbf{N}} NSPACE(c+n^k)$$

It is easy to see that $DSPACE(f(n)) = DSPACE(\lceil c\, f(n) \rceil)$ for all positive constants c. This linear *speed-up* is done by increasing the alphabet size. The same theorems hold for nondeterministic and alternating (defined below) Turing machines. In the corresponding theorems for time complexity (and for one-tape space complexity) $c\, f(n)$ is replaced by $\max(n+1, c\, f(n))$, and they hold if $\lim n/f(n) = 0$.

A *configuration* of a Turing machine consists of its state, the position(s) of its head(s), and the contents of the tape(s).

Configurations are described by instantaneous descriptions (ID). An *ID* of a one-tape (infinite only to the right) Turing machine with finite tape contents (i.e. almost everywhere is the blank symbol) is the representation of its configuration by a word, built from an initial segment of the tape inscription which contains the non-blank part. In this segment the scanned symbol s is replaced by (s, q), where q is the state of the configuration.

4. Alternating Turing machines

We assume that the reader is familiar with (one version of) deterministic Turing machines. In nondeterministic Turing machines (see e.g. [3]), the scanned symbol(s) do not determine a move (new symbol(s) and shift of head(s)), but a finite set of moves. By choosing any move of this set, the Turing machine follows a computation path. The nondeterministic Turing machine accepts, iff at least one computation path leads to an accepting configuration (i.e. a configuration with accepting state). Chandra and Stockmeyer [10] and Kozen [24] have extended the concept of nondeterministic Turing machines to alternating Turing machines [9]. Nondeterministic machines involve an existential quantification (there exists a path). Alternating machines are a natural extension involving universal as well as existential quantification. This extension from nondeterministic to alternating machines, works for all kinds of abstract machine models, but we look here only at alternating Turing machines.

Definitions (Automata theory)

Alternating Turing machines have two disjoint sets of states, existential and universal states. Configurations and successor configurations (reachable in one move) are defined as for nondeterministic Turing machines, but the conditions for acceptance are different.

An *accepting computation tree* of an alternating Turing machine M with input w is a finite tree T whose nodes are labeled with configurations of M according to the conditions:

a) The root of T is labeled with the start configuration of M with input w.

b) If a node is labeled with a configuration C, then all descendants are labeled with successor configurations of C.

c) Nodes labeled with a non accepting existential configuration have at least one descendant.