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.

d) If C' is a successor configuration of a universal configuration C, and a node N is labeled by C, then at least one descendant of N is labeled with C'.

e) All leaves are labeled with accepting configurations.

The *language* $L(M)$ accepted by an alternating Turing machine M is the set of all words w, such that there exists an accepting computation tree of M with input w.

Hence a nondeterministic Turing machine is an alternating Turing machine with only existential states.

A very intuitive way of looking at alternating Turing machines is: Two players A and E make moves (maybe not strictly alternating) beginning in the start configuration of M with input w. Player A moves from universal configurations, and E from existential configurations to successor configurations. E wins if (after finitely many moves) an accepting configuration is reached. The input w is in $L(M)$ iff E has a winning strategy.

One might first think alternating Turing machines accept all arithmetic sets and even more. But naturally, exactly the recursively enumerable sets are accepted by alternating Turing machines, because every deterministic Turing machine is an alternating Turing machine, and alternating Turing machines can easily be simulated by deterministic Turing machines.

What goes wrong if we want a player (A or E) of an alternating Turing machine to choose any natural number, is that this player could decide for ever that he wants to choose an even bigger number (computation trees have only finite branching).

The situation changes drastically if Turing machines do not accept by entering one accepting state, but by infinitely often entering accepting states. Then nondeterministic Turing machines accept exactly the Σ_1^1-sets of the analytical hierarchy, while deterministic Turing machines accept exactly the Π_2^0-sets of the arithmetical hierarchy. (It is not known, if the sets accepted by alternating Turing machines in this way have such a nice characterisation.) This remark is just to indicate that automata theory might be useful for non-recursive sets too.

Alternating Turing machines are important for several reasons. First they are a very natural extension of nondeterministic Turing machines, and they are closely related to the fundamental concept of quantifiers. Second they are a basic model of parallel computation, which is of growing importance with modern technology. And third, there are beautiful relations

between the power of time and space bounded versions of ordinary and alternating Turing machines. Some versions of alternating Turing machines with restricted alternating power (see Berman [6], Ruzzo [32]) are able to bridge the gaps between deterministic and nondeterministic time and space complexity classes. And the problems about the relation of these classes (e.g. $P = NP$? $P = POLYSPACE$?) are still the most challenging open questions in computational complexity.

Definitions (Complexity)

$ATIME(T(u))$ is the class of languages L accepted by alternating Turing machines M, such that for each input w, there exists an accepting computation tree (of M with input w) of depth $\leqslant T(n)$ (for $n = |w|$) if $w \in L$, and there exists no accepting computation tree if $w \notin L$.

$ASPACE(S(n))$ is the class of languages L accepted by alternating Turing machines M, such that for each input w, there exists an accepting computation tree (of M with input w), whose labels are $S(n)$-space bounded configurations (for $n = |w|$) if $w \in L$, and there exists no accepting computation tree if $w \notin L$. (A configuration is $S(n)$-space bounded if at most $S(n)$ tape squares on work tapes are used.)

The fundamental complexity relations between alternating and nonalternating Turing machines are (Chandra and Stockmeyer [10], Kozen [24]):

For $S(n) \geqslant n$

$$ATIME\ (S(n)) \subseteq DSPACE(S(n))$$

and

$$NSPACE(S(n)) \subseteq ATIME\ (S(n)^2)$$

For $T(n) \geqslant \log n$

$$ASPACE(T(n)) = \bigcup_{c>0} DTIME(c^{T(n)})$$

We sketch the proof of $DTIME(c^{T(n)}) \subseteq ASPACE(T(n))$, because we use this fact for the complexity result about the Ackermann case.

Let $C_{t,s}$ be the s-th symbol in the ID (instantaneous description) of the configuration at time t, of a $c^{T(n)}$ time bounded deterministic one-tape Turing machine M with input w. The computation of M is simulated backwards.

Player E says, M accepts w by entering the accepting state q_a at time t with headposition s_t. And to prove this, he presents $t-1$, a_{t-1}, b_{t-1}, c_{t-1} and $s = s_{t-1}$ ($t-1$ and s in binary), claiming that $C_{t-1,s-1} = a_{t-1}$,

$C_{t-1,s} = b_{t-1}$ and $C_{t-1,s+1} = c_{t-1}$. (Naturally these values must imply the state q_a and the headposition s_t at time t.) Now player A is allowed to doubt one of these three claims, by playing the integer $s' \in \{s-1, s, s+1\}$, and player E has to justify his claim for $C_{t-1,s'}$ by claiming values for $C_{t-2's'-1}$, $C_{t-2,s'}$ and $C_{t-2,s'+1}$ which imply his value for $C_{t-1,s'}$ etc. Finally the value claimed for $C_{0s''}$ is checked by comparison with the s''-th input symbol. If it is correct, then player E, otherwise player A wins.

If w is accepted by M, then the winning strategy for player E is to make always correct claims. If w is not accepted by M, then player A has a winning strategy. He always doubts one of the wrong claims of player E.

5. Upper bounds

Proposition. 1. *For all* $p \geqslant 0$, *the* $\exists^p \forall \exists^*$ *class is logspace transformable to the monadic* $\exists \forall \exists^*$ *class via length order* n.

2. *The* $\exists^* \forall \exists^*$ *class is logspace transformable to the monadic* $\exists^* \forall \exists^*$ *class via length order* $n^2/\log n$.

Proof. The main ideas of this proof are due to Lewis [27, Lemma 7.1] and Ackermann [2, Section VIII.1]. Given a formula F of the class $\exists^p \forall \exists^q$ with prefix $\exists x_1 \ldots \exists x_p \forall y \exists z_1 \ldots \exists z_q$ and matrix M, let S be the set of atomic formulas in M. We define the set S' by $S' = S \cup \{A[y/x_i] \mid A \in S \text{ and } 1 \leqslant i \leqslant p\}$.

Let $S' = \{A_1, \ldots, A_r\}$.
Then $|S'| = r \leqslant (p+1)|S|$.

Now we change the matrix M of F to get the formula F' with matrix M' by replacing (for $j = 1, \ldots, r$) all occurrences of the atomic formula A_j by $P_j(y)$ (for a new monadic predicate symbol P_j) and by adding — as a conjunct to M — a set B of biconditionals.

The set B is constructed to ensure that every Herbrand model α' of the functional form of the formula F' (with matrix M') defines immediately a model α of the functional form of F by $|\alpha| = |\alpha'|$,

$c_k^\alpha = c_k^{\alpha'} = c_k$, $k = 1, \ldots, p$ (where c_k is the replacement of x_k in the functional forms of F and F'),