Allowing blank symbols in words makes a difference.

Abstract. We construct a language with possible blank symbols in its words. This language is decided by a polynomial time non deterministic Turing machine. However, this language is decided by no deterministic Turing machine.

1. Introduction.

Proving that $\mathbf{P} \neq \mathbf{NP}$ is difficult for many reasons. Here are three of them.

(a) A deterministic Turing machine (DTM) is not very different from a non deterministic Turing machine (NDTM).

(b) The set of polynomial time DTMs is not recursively enumerable. Hence it is very difficult to check whether a word codes a polynomial time DTM.

(c) It is even more difficult for a single polynomial time NDTM (that runs for instance in $O(n^2)$ steps) to catch the set of polynomial time DTMs that run in $O(n^k)$ steps for arbitrarily large integers k.

Unfortunately, you will not find in this paper a short proof that $\mathbf{P} \neq \mathbf{NP}$. However we will try to deal with the difficulties we have mentioned. According to the point **(b)**, it is easier to consider all the DTMs instead of the polynomial time DTMs only. Allowing blank symbols in inputs will enable us to prove a result that goes in the direction of $\mathbf{P} \neq \mathbf{NP}$.

2. Extended languages.

Definition. *We fix a finite alphabet Σ which is common to every TMs and a coding of DTMs by finite words C that represent all the possible transitions of the coded DTM $M(C)$ like it is usually done in the definitions of universal machines. Let T be the set of these codes C. Initially the tapes are fulfilled with blank symbols $\circ$. Hence an input w is viewed as $w\circ^\infty$. We allow that a code C can be extended with a finite sequence of blank symbols $\circ$ followed by a symbol $\bullet$. Let S be the set of these codes C or $C\circ^k\bullet$ for every finite integer $k \geq 0$. We say that a language like S, which can contain words that end with a finite sequence of blank symbols followed by a $\bullet$, is an extended language. Let $\mathbf{NP}^\circ$ be the set of extended languages decided by polynomial time NDTMs and let $\mathbf{DTIME}^\circ$ be the set of extended languages decided by DTMs.*

The extensions of codes enable to artificially increase their lengths in order to integrate a kind of time counter. That will be a possible answer for the point **(c)**.

3. $\mathbf{NP}^\circ$ versus $\mathbf{DTIME}^\circ$.

We are going to separate the complexity classes $\mathbf{NP}^\circ$ and $\mathbf{DTIME}^\circ$. The method is quite artificial but it is not very complicated.

Theorem. *There exists an explicit extended language L with $L \in \mathbf{NP}^\circ$ and $L \notin \mathbf{DTIME}^\circ$.*

Proof. This extended language L is a subset of the extended language S. We construct a NDTM N that decides this language L and we explain how works N on an input w. First, N enters in a non deterministic stage : N guesses a word x with x empty or of the form $\circ^k\bullet$. Then N checks whether $w = Cx$ where $C \in T$. If this guessing stage succeeds, then $w = C$ or $w = C\circ^k\bullet$ and w lies in S. Then C is copied on the second tape, w is copied on the third tape and 1^n is written on the first tape. Then N checks by simulation whether the DTM $M(C)$ (coded on the second tape) accepts or rejects w (written on the third tape) in at most $n = |w|$ steps (one step for each symbol 1 on the first tape). If that is not the case then N rejects w. Otherwise N returns the

opposite answer of $M(C)$: if $M(C)$ rejects w then N accepts w and if $M(C)$ accepts w then N rejects w. Let L be the subset of S made of all the inputs accepted by N. The computation of N shows that $L \in \mathbf{NP}^{\circ}$: the guessing stage takes $O(n^2)$ steps because codes with non deterministic transitions must be rejected and the simulation of at most n steps of $M(C)$ on w takes also $O(n^2)$ steps.

Assume that for some $D \in T$, the DTM $M(D)$ decides the language L. Two cases are possible :

1) If $D \in L$ then N accepts D. There exists for N an accepting computation on the input D if N guesses an empty word x, recognize that $D \in S$ and only if the simulation of $M(D)$ on D ends by a rejection in at most $|D|$ steps. Hence $D \notin L$ because $M(D)$ is supposed to decide L. But D was supposed to belong to L.

2) If $D \notin L$ then $M(D)$ rejects D in at most K steps for some finite integer K because $M(D)$ is supposed to decide L in finite time. Now, we consider the extended code $D\circ^K\bullet$. The DTM $M(D)$ will also reject $D\circ^K\bullet$ because $M(D)$ will not be able to make a difference between D and $D\circ^K\bullet$ since $M(D)$ takes at most K steps on the tape with the word $D\circ^{\infty}$. Hence $M(D)$ will reject $D\circ^K\bullet$ in at most $K < |D\circ^K\bullet|$ steps. There exists for N an accepting computation on the input $D\circ^K\bullet$. It is sufficient for N to guess the word $x = \circ^K\bullet$, recognize that $D\circ^K\bullet \in S$ and verify that $M(D)$ rejects $D\circ^K\bullet$ in at most $|D\circ^K\bullet|$ steps. Hence N accepts $D\circ^K\bullet$ and $D\circ^K\bullet \in L$. But $D\circ^K\bullet$ is rejected by $M(D)$.

In both cases there is a contradiction. Hence, the hypothesis that some DTM $M(D)$ can decide the language L is false. Of course, one could imagine another DTM D' that would try to decide the language L by taking more time and care on inputs like D and $D\circ^K\bullet$ in order to give then the correct answers. But the contradictions would appear then with D' itself. Hence $L \notin \mathbf{DTIME}^{\circ}$.

■

4. Conclusion.

First, one can see that the difference between the NDTM N and a DTM is quite small. That corresponds to the point **(a)**. However, in the context of this work, this difference makes a single polynomial time NDTM stronger than every DTMs. Secondly, the author hopes that the ideas of this paper will be improved for obtaining stronger results. Thirdly, this short paper justifies the research in the field of quantum computers and for new models of computation like our recent work based on linear algebra : "Machines Matricielles"