Deadline Turing Machine

Rudi Dittrich
rdittrich@cosy.sbg.ac.at

Werner Gitschthaler
wgitscht@cosy.sbg.ac.at

Harald Röck
hroeck@cs.uni-salzburg.at

***Abstract* — We report on a new computational model namely the *Deadline Turing Machine* DTM. The DTM is an extension, although less powerful, of the well known Turing Machine(TM) [2]. It uses a deadline within words of a given language can be recognized and decided. If a word cannot be decided within the deadline it is rejected by the DTM. This property implies that opposed to a TM it always halt. Moreover, a DTM can easily be mapped on a TM and is equal to it if and only if the deadline is infinite. Finally, we show that there exists a *Universal Deadline Turing Machine*(UDTM) which can be built of an universal TM with an additional tape and independent head.**

1 Introduction

The Turing Machine(TM) was introduce by Alan Turing in 1936[2]. A TM is an abstract mathematical entity that is composed of a (infinite) tape, a read-write head, and a finite state automaton. The head can either move to the left or to the right, after it wrote a symbol of the tape alphabet to the tape. The finite state machine keeps track in which state the TM is in. By reading the symbol at the heads position from the tape the finite state machine can determine what the next state will be, what to write to the tape and in which direction to move the head. The formal definition of a TM, taken from [1], is a seven tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the special *blank* symbol $\sqcup$,
3. Γ is the alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma \backslash \{\sqcup\}$,
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state
6. $q_{accept} \in Q$ is the accept state
7. $q_{reject} \in Q$ is the reject state, where $q_{accept} \neq q_{reject}$, and

.

We propose an extension to this model, which introduces a deadline to all computations performed by a TM. We call this model *Deadline Turing Machine*, DTM for short. The deadline could be any notion of time, e.g. real time. We chose a state transition as a clock tick, because it is easy to extend an existing TM to a DTM by counting the state transitions performed so far.

2 Formal Definition

A *Deadline Turing Machine* is an eight tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}, d)$ where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the special *blank* symbol $\sqcup$,
3. Γ is the alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma \backslash \{\sqcup\}$,
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state
6. $q_{accept} \in Q$ is the accept state
7. $q_{reject} \in Q$ is the reject state, where $q_{accept} \neq q_{reject}$, and
8. d is the deadline as an integer greater zero

.

A DTM D is a Turing Machine(TM) $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ with an additional deadline d: the maximum number of state transitions before a word is rejected. i.e. if the number of state transitions is d and M is not in the accept state the input word is rejected.

3 Properties

3.1 Halting Problem

A DTM always halts if $d < \infty$

Proof After a finite number of state transitions the DTM will reject the input, because of a deadline violation.

3.2 Power of DTM

Some words of a language are rejected by a DTM but accepted by a TM. i.e. the DTM is less powerful than a TM.

Proof There exists a language which is turing decidable, but not correctly decidable by a DTM, i.e. some languages can not be decided by a DTM although they are turing decidable. The language $L = 1^*$(every word is a sequence of ones) is clearly turing decidable. Eventually every DTM would reject a word, although the TM accepts it. Let the deadline of the DTM D be arbitrary but finite: $d = k; k < \infty$. The TM accepts the input word $w = 1^{k+1}$, but D rejects it.

3.3 Decidability

The set of languages decided by a DTM is a strict subset of the turing decidable languages.

Proof First we show that all languages decided by a DTM can be decided by a TM. Then we give an example of a language which can be decided by a TM but not by a DTM.

$\Longrightarrow$ All words in a language L accepted by DTM D have finite length, because of the deadline. The alphabet Σ is also a finite set. Hence we can enumerate all words accepted by D . Let the number of words of L be n and $L = \{l_1, l_2, \ldots, l_n\}$, hence a TM M will decide L by comparing an input word w with $l_1, l_2, \ldots, l_k; k \leq n$. If it finds a $l_k = w$ *accept* otherwise *reject*.

$\Longleftarrow$ follows directly from property 3.2.

3.4 Universal DTM

There exists a universal deadline turing machine, UDTM and it is possible to implement it with a (multi-tape) universal turing machine, UTM. Similar to a UTM, a UDTM gets a description of a DTM D and a word w as input $< D, w >$. It emulates D like a UTM would emulate D without a deadline. And it accepts $< D, w >$ if and only if the word is accepted before the deadline is reached. Unlike a UTM a UDTM can not emulate any DTM; it can only emulate these DTM with a smaller deadline than it self.

Proof For the implementation of a universal DTM UD we can use a universal TM UM with an additional tape and an independent head. Initially the extra tape contains the deadline in unary representation and the head is at the leftmost position. On every state transition of D simulated by UM the second head moves one step to the right. If it reads a *blank* symbol the deadline is reached and the input is rejected. Figure 1 illustrates the tapes of a implementation of the DTM with a multitape TM. This DTM

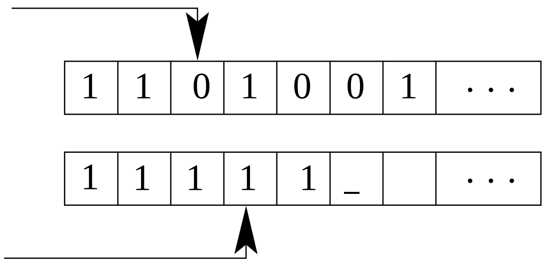

Figure 1: A implementation of a DTM with a multi-tape TM

has only one more state transition unitl the deadline is reached and the input word is rejected. For simplicity we ommited the state automaton.

Let d_u be the deadline of the UDTM and d_d the deadline of D. It is necessary that $d_u > d_d$. There is an integer k as the maximum number of state transition needed by UDTM to emulate one state transition of D. Hence, d_u must at least be $k \cdot d_d$, i.e. $d_u \geq k \cdot d_d$, to correctly simulate D with UD.

3.5 DTM to TM

If $d \longrightarrow \infty$ then DTM $\longrightarrow$ TM

Proof This follows from the definition. If the deadline is infinity, it is never reached and an input is never rejected because of a deadline violation. Hence it behaves like a regular TM.

4 Conclusion

We presented a new model of computation: The *Deadline Turing Machine*. It can be seen as an extension of the Turing Machine. Alltough it is strictly less powerfull as the TM it has some other interesting properties, e.g. it always halts. The DTM is also a more realistic model than the TM, because of the deadline it does not run forever and we think it would not need an infinity long tape. However, we were not able to proof these property yet.

References

[1] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1996.

[2] A.M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. LMS, Series 2*, 42:230–265, 1936-1937.