CSE596 Lecture Wed 9/26 Fall 2018

Consequences of the Universal RAM Simulator T[h]

① For every program P written in a (known/HL[L]) U
we can build a DTM M_P s.t. for all x on stdin

$M_P(x)$ has an emulating computation of $P(x)$

Proof: ⓐ Compile P to object code O_P in "my mini-assembly"

ⓑ Design M_P with an "initial bank" of states that

- First copy x as $[x]$ on Tape 3 of U.
- Use $|O_P|$ special one-off states to overwrite x on tape 1 with the textual code of O_P's instructions.

Now we have the initial setup for U running O_P on x
so go to the start state of U.

$$M_P = \boxed{\text{Init}} \xrightarrow{\varepsilon} \boxed{s \quad U}$$

$M_P(x) = U(P, x) = P(x)$. Size $\approx |O_P| + |U| \approx |P| + $con[st]

- The time used by a computation is its # of steps
★ The space used is the # of cells in which a char was changed to a different char.

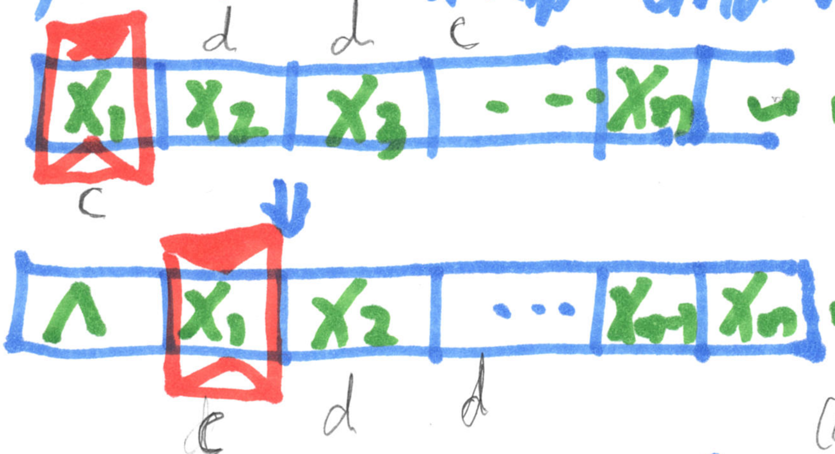
② NB: Theorem [Steve Cook, 1971]

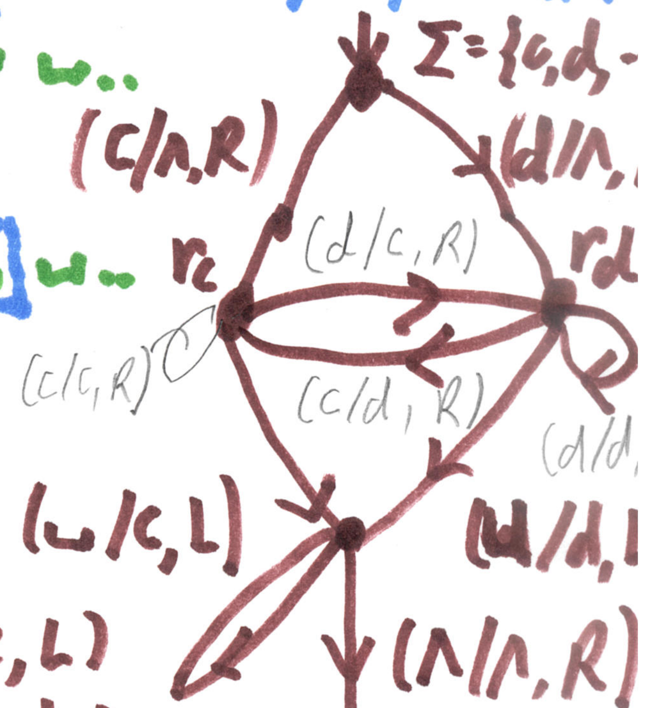
Not his Famous 1971 Thm on SAT!

There is a clever way to build U so that every t steps of P(x) judged at "fair cost" is simulated by $O(t^2)$ steps of U.

If you use an m-bit int you get charged m time uni

Ideas Involved. Suppose we get input $X = X_1 X_2 \cdots$ left-justified on a tape and want to prepend a





done: attach to start of rest of

Furthermore, this "shift over" routine can be called to make more room in the middle of a tape, e.g. to write to a "register" of U.

[label # val]

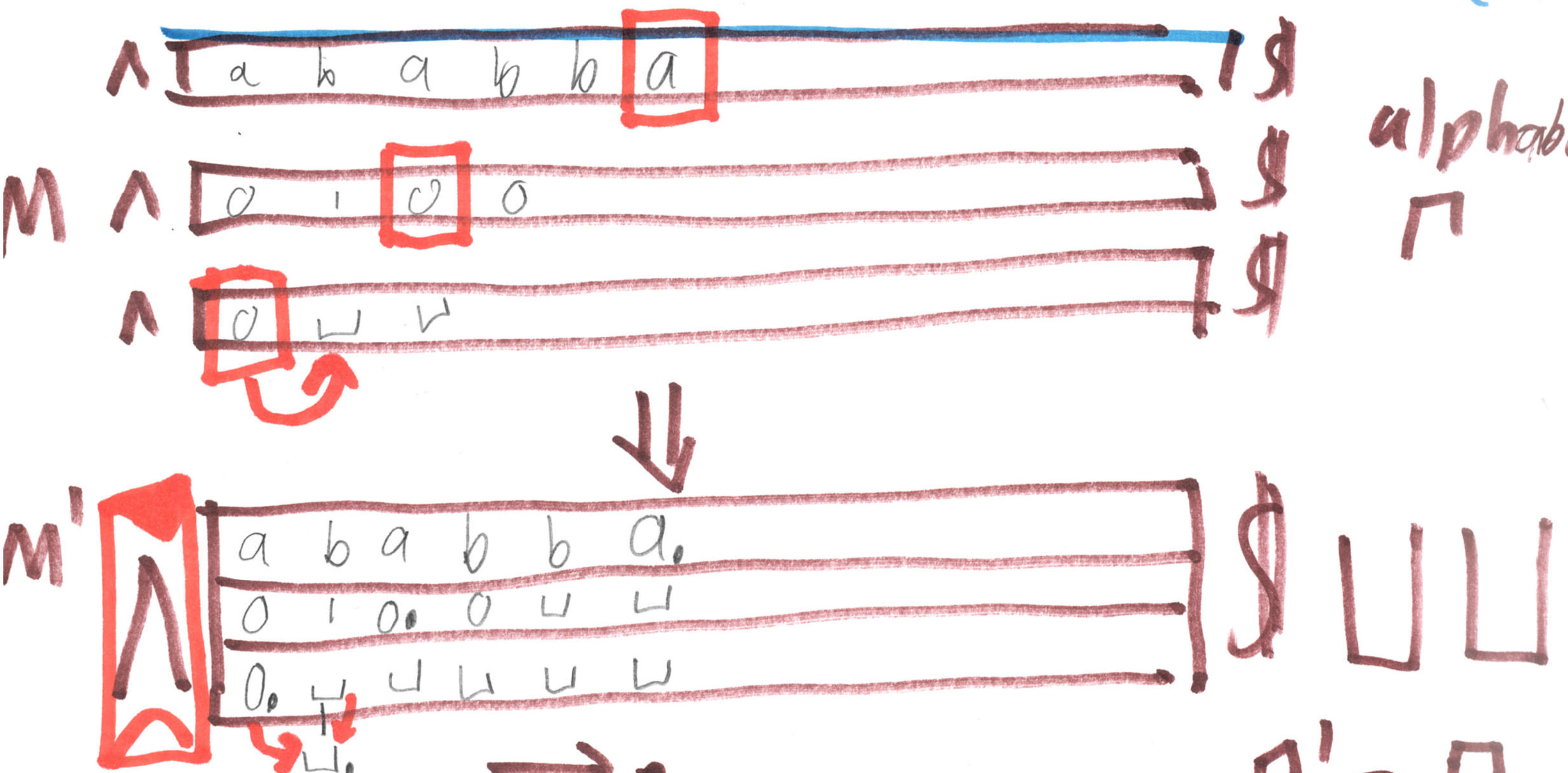
We can attach a separate copy of "sh.over", entered on `]` at any state q of A. Can picture it like a unix "daemon".

daemon

Doing this simplistically gives time $\approx O(t^4)$

NB A clever doubling scheme like C++ vector gives the $O(t^2)$. ∎

③ Thm. A TM with 3 (or any number K) tapes can be simulated by a one-tape TM in time $O(t^2)$

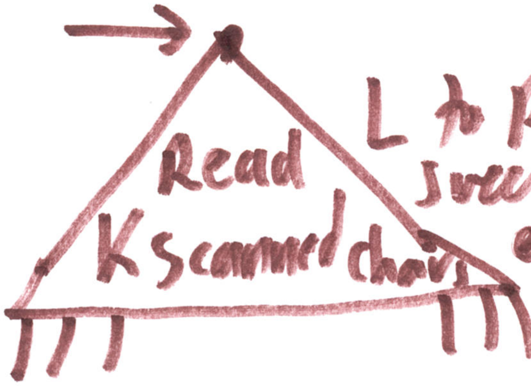


alphabet Γ

$\Gamma' = \Gamma \; \cancel{\cup\, \Gamma^K}$

$\cup\, (\Gamma \cup \dot{\Gamma})^K$

$|(\Gamma \cup \dot{\Gamma})^K|$ possible $c_1 \dots c_k$



L to R sweep end at $

Go back to ∧ (execute each one in R to L sweep

Overhead scales as # steps already taken ∴ time is $O(t^2)$, space is same.

Added: The notes mention that Sipser's text gives a different method that manages K registers on one tape [like my "registers" but separated by # not [ ]]. That uses a similar "shift over" routine and (hence) is less efficient as well as "messy."