Monash University
Faculty of Information Technology

FIT2014 Theory of Computation

Lecture 29
NP-completeness: the Cook-Levin Theorem

slides by Graham Farr

Overview

- Proof of the Cook-Levin Theorem
- This proof is non-examinable,
  although it uses ingredients from other parts of the unit (see, e.g., Tute 7, Q9).

Cook-Levin Theorem

Our first NP-complete language:

SATISFIABILITY:
the set of satisfiable Boolean expressions in CNF

Cook-Levin Theorem
SATISFIABILITY is NP-complete.

History: S. Cook (1971), L. Levin (1972)

https://commons.wikimedia.org/wiki/File:Stephen_A._Cook_1968_(enlarged_portion).jpg

Stephen Cook (b. 1939)
in 1968

https://www.cs.bu.edu/fac/lnd/

Leonid Levin (b. 1948)

Proof

Proof.

Let L be any language in NP.
We must give a polynomial-time reduction from L to SATISFIABILITY.

Let V be a Turing machine that is a polynomial-time verifier for L.

- input alphabet $\{\texttt{a},\texttt{b}\}$
- tape alphabet $\{\texttt{a},\texttt{b},\texttt{\#}\}$. Blank cells represented by Δ.
- p states

So, the tape of V initially contains two strings, x and y:

- x is the input string whose membership, or not, of L is under consideration;
- y is the certificate.
- Assume that x and y are separated on the tape by #.
  So the tape initially holds the string $x\#y$.

Proof

V being a **verifier** for L means:

$$x \in L \quad \text{if and only if} \quad \exists y : V(x,y) \text{ accepts.}$$

V running in **polynomial time** means:

$$\exists N, c, k \;\; \forall x \text{ such that } |x| \geq N \;\; \forall\, y \; : \quad t_V(x,y) \;\leq\; c\,|x|^k.$$

For convenience, put $T(n) := \lfloor c\, n^k \rfloor$.
This gives us an integer-valued polynomial upper bound for the time taken when $|x| = n$.

Proof

We will describe the *entire computation* of V starting with $x\#y$, by a Boolean expression φ_x in Conjunctive Normal Form.

We must ensure:

$$\exists y : V(x,y) \text{ accepts} \qquad \text{if and only if} \qquad \exists \text{ truth assignment} : \varphi_x \text{ is True}.$$

We will express the proposition

$$V(x,y) \text{ accepts}$$

as a conjunction of more specific propositions,
and keep doing so,
until we have the CNF expression we need.

Boolean variables

To begin with, we need Boolean variables that describe every possibility for every little piece of V at every possible time during the computation.

variable	intended meaning	
$Q_{t,q}$	At time t, the machine is in <u>state</u> q.	$0 \leq t \leq T(n), \quad 1 \leq q \leq p.$
$S_{t,s,\ell}$	At time t, tape <u>cell</u> s contains letter ℓ.	$0 \leq t \leq T(n), \quad 1 \leq s \leq T(n), \quad \ell \in \{\mathtt{a}, \mathtt{b}, \#, \Delta\}$
$H_{t,s}$	At time t, Tape <u>Head</u> is scanning tape cell s.	$0 \leq t \leq T(n), \quad 1 \leq s \leq T(n).$

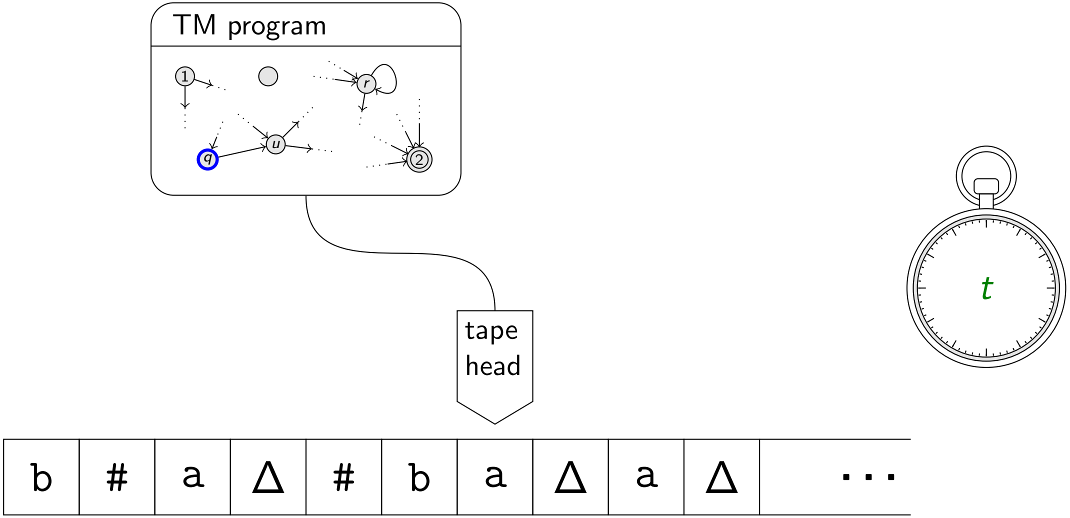
Boolean variables

To begin with, we need Boolean variables that describe every possibility for every little piece of V at every possible time during the computation.

variable	intended meaning	
$\bigcirc_{t,q}$	At time t, the machine is in state q.	$0 \leq t \leq T(n), \quad 1 \leq q \leq p.$
$\square_{t,s,\ell}$	At time t, tape cell s contains letter ℓ.	$0 \leq t \leq T(n), \quad 1 \leq s \leq T(n), \quad \ell \in \{\mathtt{a}, \mathtt{b}, \mathtt{\#}, \Delta\}$
⬠$_{t,s}$	At time t, Tape Head is scanning tape cell s.	$0 \leq t \leq T(n), \quad 1 \leq s \leq T(n).$

How many variables altogether? Is this polynomially bounded, in n?

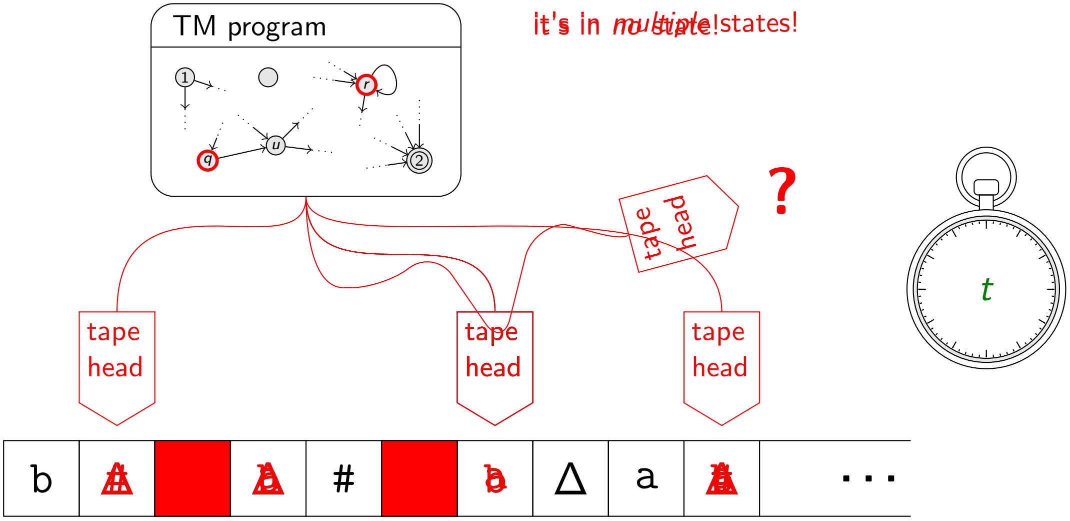
What we *want* these variables to describe:


TM program
1
r
u
q
2
tape head
t
b
#
a
Δ
#
b
a
Δ
a
Δ
...

What might actually happen, if we just let the variables loose:

it's in *no state*!

it's in *multiple* states!



Static conditions

For every time t: **The TM configuration is sane.**

- The machine is in exactly one state.
- The Tape Head is in exactly one position.
- For every tape cell s, the cell contains exactly one letter.

At time 0: **The initial set-up is correct.**

- The machine is in the Start state.
- The Tape Head is scanning the first tape cell.
- Tape cells 1 to n contain the letters of x, and tape cell $n+1$ contains #.

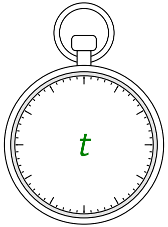
At time $T(n)$: **The TM has accepted.**

- The machine is in the Accept state.

Static conditions: time t

For every time t: **The TM configuration is sane.**

- The machine is in exactly one state.
- The Tape Head is in exactly one position.
- For every tape cell s, the cell contains exactly one letter.



For every time t, the machine is in exactly one state.

- For every time t, the machine is in <u>at least</u> one state.

$$\bigcirc_{t,1} \vee \bigcirc_{t,2} \vee \cdots \vee \bigcirc_{t,p}$$

- For every time t, $\underbrace{\text{the machine is in \underline{at most} one state.}}$

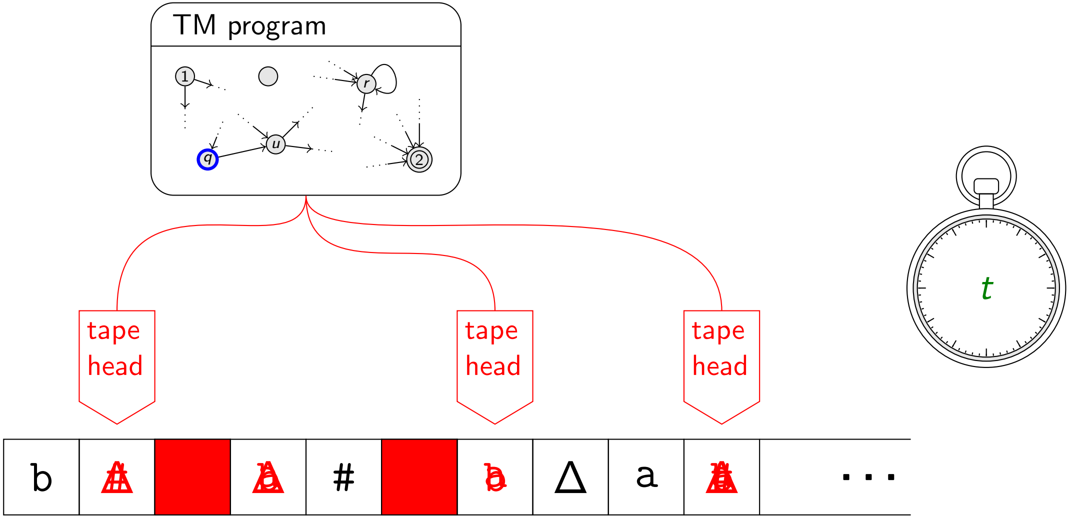
$\overbrace{\text{for each pair of states } q, r\text{, the machine is not in state } q \text{ or it's not in state } r.}$

$$(\neg\bigcirc_{t,q} \vee \neg\bigcirc_{t,r})$$

Joining them together, for time t:

$$\begin{aligned}
(\neg\bigcirc_{t,1} \vee \neg\bigcirc_{t,2}) \wedge (\neg\bigcirc_{t,1} \vee \neg\bigcirc_{t,3}) \wedge (\neg\bigcirc_{t,1} \vee \neg\bigcirc_{t,4}) \wedge \cdots \wedge (\neg\bigcirc_{t,1} \vee \neg\bigcirc_{t,p})\\
\wedge (\neg\bigcirc_{t,2} \vee \neg\bigcirc_{t,3}) \wedge (\neg\bigcirc_{t,2} \vee \neg\bigcirc_{t,4}) \wedge \cdots \wedge (\neg\bigcirc_{t,2} \vee \neg\bigcirc_{t,p})\\
\wedge (\neg\bigcirc_{t,3} \vee \neg\bigcirc_{t,4}) \wedge \cdots \wedge (\neg\bigcirc_{t,3} \vee \neg\bigcirc_{t,p})\\
\ddots \qquad \qquad \vdots\\
\wedge (\neg\bigcirc_{t,p-1} \vee \neg\bigcirc_{t,p})
\end{aligned}$$

Then form:
(expression for $t = 0$) $\wedge$ (expression for $t = 1$) $\wedge \cdots\cdots \wedge$ (expression for $t = T(n)$)


TM program
1
q
u
r
2
tape head
tape head
tape head
t
b
#
a
Δ
a
. . .

For every time t, the Tape Head is in exactly one position.

- For every time t, the Tape Head is in <u>at least</u> one position.

$$\bigtriangledown_{t,1} \vee \bigtriangledown_{t,2} \vee \cdots \vee \bigtriangledown_{t,T(n)}$$

- For every time t, $\underbrace{\text{the Tape Head is in \underline{at most} one position.}}$

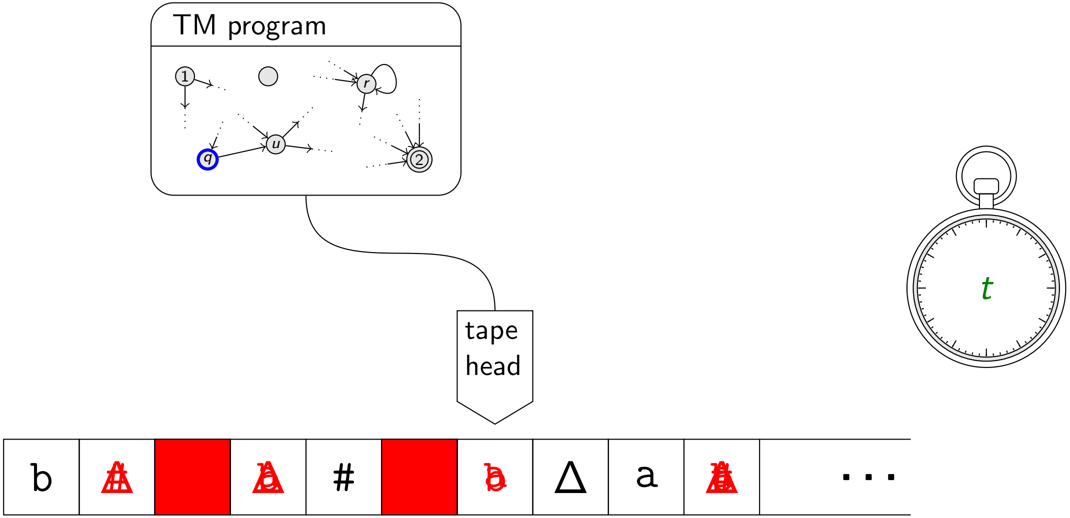
$$\overbrace{\text{for each pair of tape cells } s_1, s_2 \text{, the Tape Head is not at cell } s_1 \text{ or it's not at cell } s_2}$$

$$(\neg\bigtriangledown_{t,s_1} \vee \neg\bigtriangledown_{t,s_2})$$

Joining them together, for time t:

$$\begin{aligned}
&(\neg\bigtriangledown_{t,1} \vee \neg\bigtriangledown_{t,2}) \wedge (\neg\bigtriangledown_{t,1} \vee \neg\bigtriangledown_{t,3}) \wedge (\neg\bigtriangledown_{t,1} \vee \neg\bigtriangledown_{t,4}) \wedge \cdots \wedge (\neg\bigtriangledown_{t,1} \vee \neg\bigtriangledown_{t,T(n)}) \\
&\qquad \wedge (\neg\bigtriangledown_{t,2} \vee \neg\bigtriangledown_{t,3}) \wedge (\neg\bigtriangledown_{t,2} \vee \neg\bigtriangledown_{t,4}) \wedge \cdots \wedge (\neg\bigtriangledown_{t,2} \vee \neg\bigtriangledown_{t,T(n)}) \\
&\qquad\qquad \wedge (\neg\bigtriangledown_{t,3} \vee \neg\bigtriangledown_{t,4}) \wedge \cdots \wedge (\neg\bigtriangledown_{t,3} \vee \neg\bigtriangledown_{t,T(n)}) \\
&\qquad\qquad\qquad\qquad \ddots \qquad\qquad \vdots \\
&\qquad\qquad\qquad\qquad \wedge (\neg\bigtriangledown_{t,T(n)-1} \vee \neg\bigtriangledown_{t,T(n)})
\end{aligned}$$

Then form:
(expression for $t = 0$) $\wedge$ (expression for $t = 1$) $\wedge \cdots\cdots \wedge$ (expression for $t = T(n)$)


TM program
1
r
u
q
2
tape head
b
#
Δ
a
...
t

For every time t and tape cell s, the cell contains exactly one letter.

- For every time t and cell s, the cell contains <u>at least</u> one letter.

$$\square_{t,s,\mathtt{a}} \vee \square_{t,s,\mathtt{b}} \vee \square_{t,s,\#} \vee \square_{t,s,\Delta}$$

- For every time t and cell s, $\underbrace{\text{the cell contains } \underline{\text{at most}} \text{ one letter}}_{\text{for each pair of letters } \ell,\, m, \text{ the cell doesn't contain } \ell \text{ or it doesn't contain } m}$.

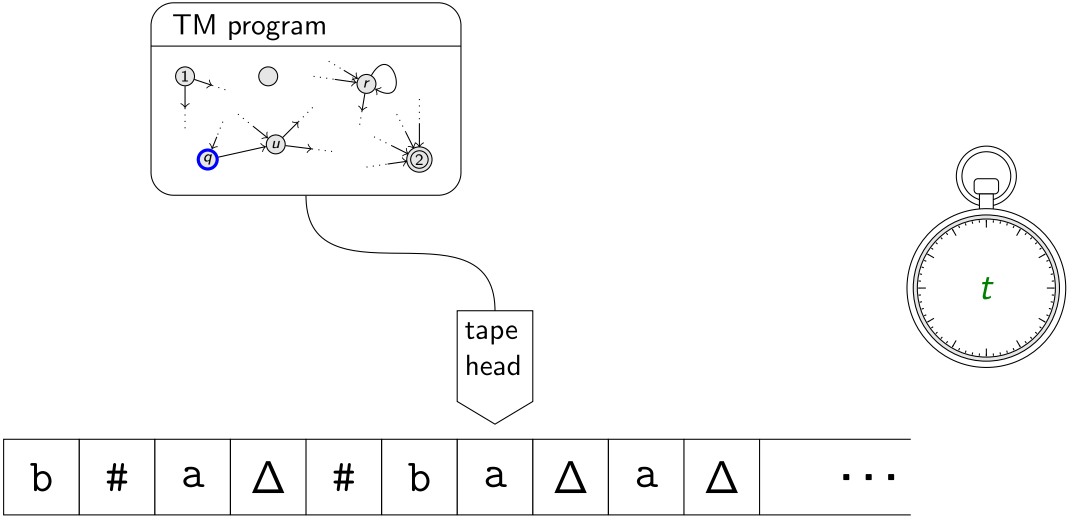
$$(\neg\square_{t,s,\ell} \vee \neg\square_{t,s,m})$$

Joining them together, for time t and cell s:

$$\begin{aligned}(\neg\square_{t,s,\mathtt{a}} \vee \neg\square_{t,s,\mathtt{b}}) \wedge (\neg\square_{t,s,\mathtt{a}} \vee \neg\square_{t,s,\#}) \wedge (\neg\square_{t,s,\mathtt{a}} \vee \neg\square_{t,s,\Delta})\\ \wedge (\neg\square_{t,s,\mathtt{b}} \vee \neg\square_{t,s,\#}) \wedge (\neg\square_{t,s,\mathtt{b}} \vee \neg\square_{t,s,\Delta})\\ \wedge (\neg\square_{t,s,\#} \vee \neg\square_{t,s,\Delta})\end{aligned}$$

Then form:

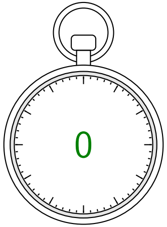
$$\begin{array}{ccccc}
(\text{expression for } t=0,\ s=0) & \wedge & \cdots\cdots & \wedge & (\text{expression for } t=0,\ s=T(n)) \quad \wedge \\
(\text{expression for } t=1,\ s=0) & \wedge & \cdots\cdots & \wedge & (\text{expression for } t=1,\ s=T(n)) \quad \wedge \\
\vdots & & \vdots & & \vdots \\
(\text{expression for } t=T(n),\ s=0) & \wedge & \cdots\cdots & \wedge & (\text{expression for } t=T(n),\ s=T(n))
\end{array}$$


TM program
1
r
u
q
2
tape head
t
b
#
a
Δ
#
b
a
Δ
a
Δ
...

Static conditions: time 0

At time 0: **The initial set-up is correct.**

- The machine is in the Start state.
- The Tape Head is scanning the first tape cell.
- Tape cells 1 to n contain the letters of x, and tape cell $n+1$ contains #.



At time 0, the machine is in the Start state.

$$(\bigcirc_{0,1})$$

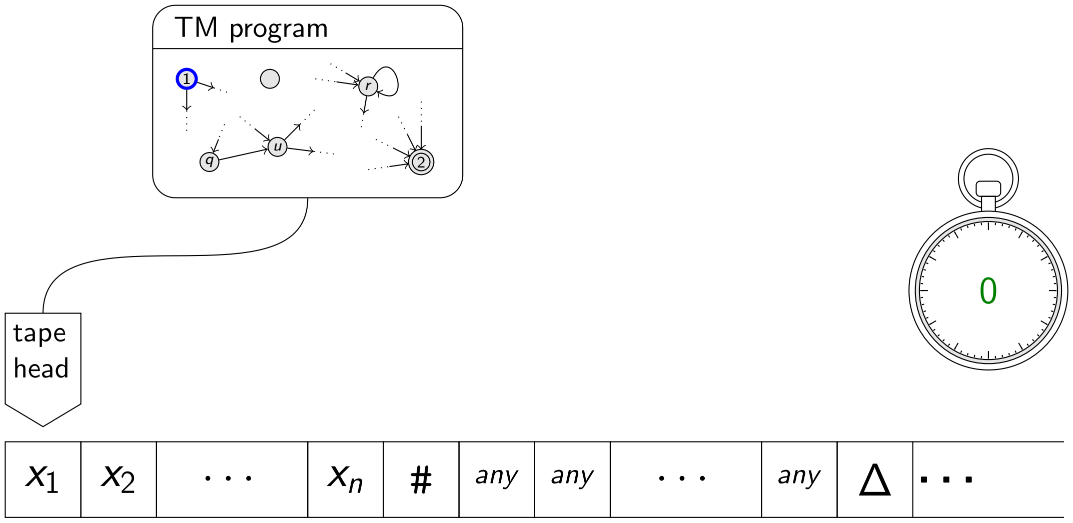
At time 0, the Tape Head is scanning the first tape cell.

$$(⬠_{0,1})$$

At time 0, tape cells 1 to n contain the letters of x, and tape cell $n+1$ contains `#`.

- Suppose $x = x_1 x_2 \cdots x_n$, where each $x_i \in \{\texttt{a}, \texttt{b}\}$.

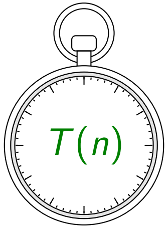
$$(\square_{0,1,x_1}) \wedge (\square_{0,2,x_2}) \wedge (\square_{0,3,x_3}) \wedge \cdots \wedge (\square_{0,n,x_n}) \;\wedge\; (\square_{0,n+1,\texttt{\#}})$$


TM program
1
r
u
q
2
tape head
0
x_1
x_2
$\cdots$
x_n
#
any
any
$\cdots$
any
Δ
$\cdots$

Static conditions: time $T(n)$

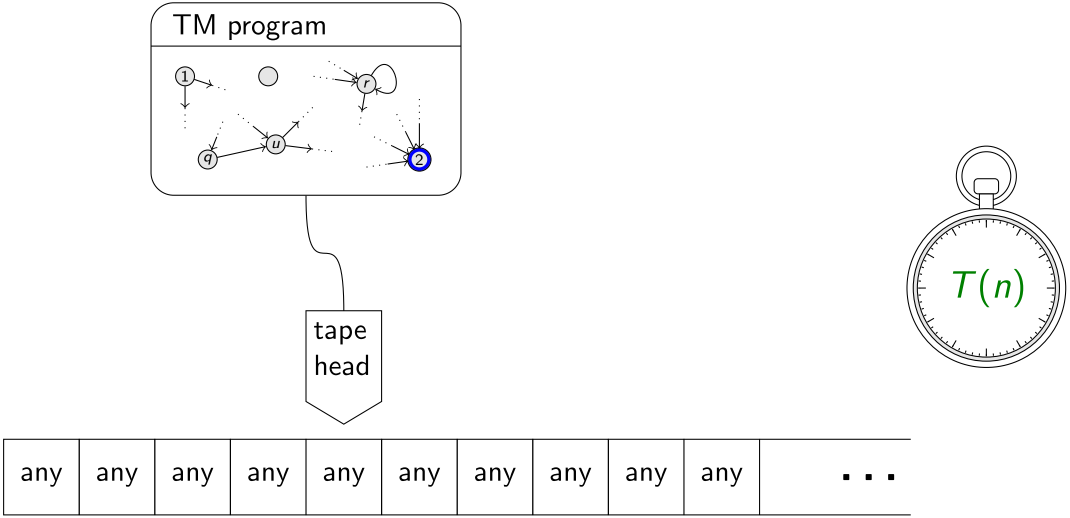
At time $T(n)$: **The TM has accepted.**

- The machine is in the Accept state.

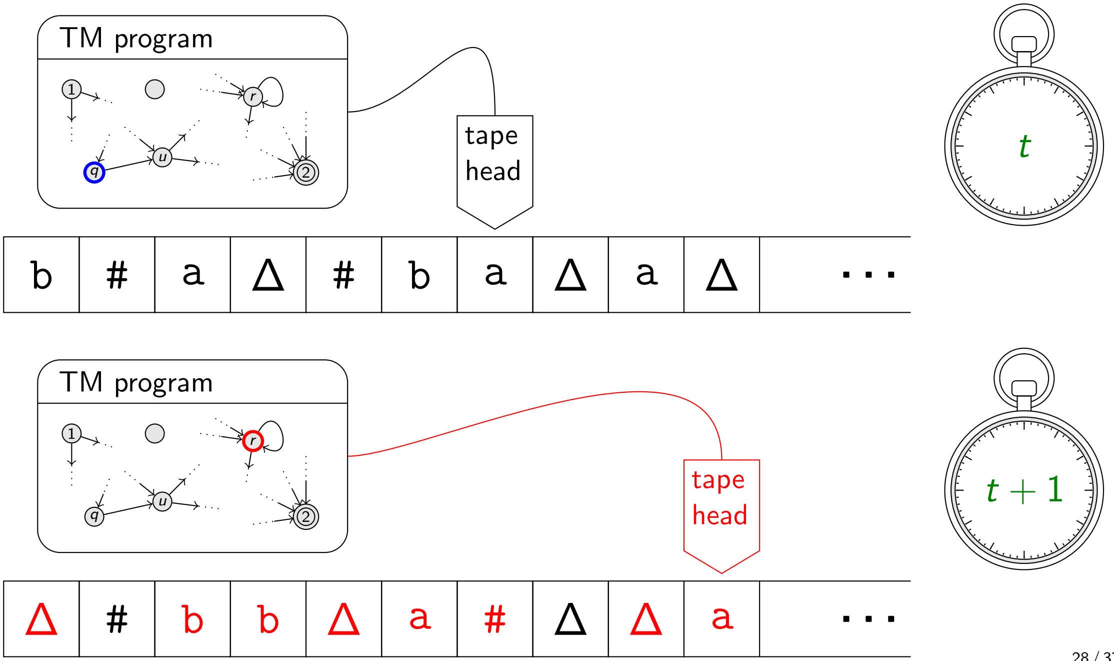


At time $T(n)$, the machine is in the Accept state.

$$(\bigcirc_{T(n),2})$$


TM program
1
r
u
q
2
tape
head
$T(n)$
any
any
any
any
any
any
any
any
any
any
...

Now, what about going from time t to time $t + 1$?


TM program
1
r
u
q
2
tape head
b # a Δ # b a Δ a Δ ...
t
TM program
tape head
Δ # b b Δ a # Δ Δ a ...
t + 1

Dynamic conditions

Conditions to describe how the TM changes from time t to $t+1$:

Cell content can only change at the tape head.
For every time t and tape cell s,
if the machine is <u>not</u> scanning tape cell s,
then the letter in this tape cell stays the same from time t to $t+1$.

Things change according to transitions.
For every time t, tape cell s, state q and letter ℓ,
if the machine is in state q, reading letter ℓ, and scanning tape cell s,
then at time $t+1$,

- the state and letter are as given by the transition for q and ℓ,
- the tape cell being scanned is $s-1$ or $s+1$
  according to the direction (Left or Right) specified by that transition.

Cell content can only change at the tape head.

For every time t and tape cell s,
if the machine is <u>not</u> scanning tape cell s,
then the letter in this tape cell stays the same from time t to $t+1$.

For each ℓ:

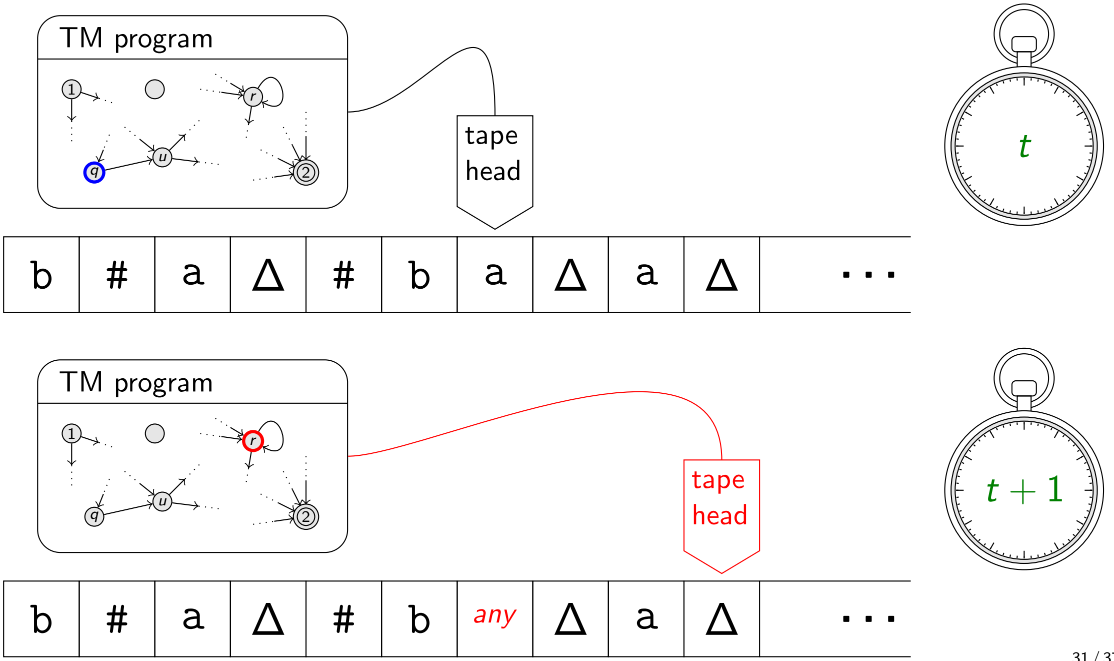
$$(\neg \bigtriangledown_{t,s} \wedge \square_{t,s,\ell}) \Longrightarrow \square_{t+1,s,\ell}$$

In CNF, for ℓ:

$$\bigtriangledown_{t,s} \vee \neg \square_{t,s,\ell} \vee \square_{t+1,s,\ell}$$

Altogether:

$$\begin{array}{ll} & (\bigtriangledown_{t,s} \vee \neg \square_{t,s,\mathtt{a}} \vee \square_{t+1,s,\mathtt{a}}) \\ \wedge & (\bigtriangledown_{t,s} \vee \neg \square_{t,s,\mathtt{b}} \vee \square_{t+1,s,\mathtt{b}}) \\ \wedge & (\bigtriangledown_{t,s} \vee \neg \square_{t,s,\#} \vee \square_{t+1,s,\#}) \\ \wedge & (\bigtriangledown_{t,s} \vee \neg \square_{t,s,\Delta} \vee \square_{t+1,s,\Delta}) \end{array}$$


TM program
1
q
u
r
2
tape head
t
b # a Δ # b a Δ a Δ ...
TM program
1
q
u
r
2
tape head
t + 1
b # a Δ # b any Δ a Δ ...

Things change according to transitions.

For every time t, tape cell s, state q and letter ℓ,
if the machine is in state q, reading letter ℓ, and scanning tape cell s,
then at time $t+1$,

- the state and letter are as given by the transition for q and ℓ,
- the tape cell being scanned is $s-1$ or $s+1$
  according to the direction (Left or Right) specified by that transition.

Things change according to transitions.

$$\begin{tikzcd} q \arrow[r, "\ell \to m, d"] & u \end{tikzcd}$$

$$\sigma := \begin{cases} +1, & \text{if } d \text{ is Right;} \\ -1, & \text{if } d \text{ is Left.} \end{cases}$$

$$\begin{aligned}
(\bigcirc_{t,q} \wedge \square_{t,s,\ell} \wedge \triangledown_{t,s}) &\implies \bigcirc_{t+1,u} \\
(\bigcirc_{t,q} \wedge \square_{t,s,\ell} \wedge \triangledown_{t,s}) &\implies \square_{t+1,s,m} \\
(\bigcirc_{t,q} \wedge \square_{t,s,\ell} \wedge \triangledown_{t,s}) &\implies \triangledown_{t+1,s+\sigma}
\end{aligned}$$

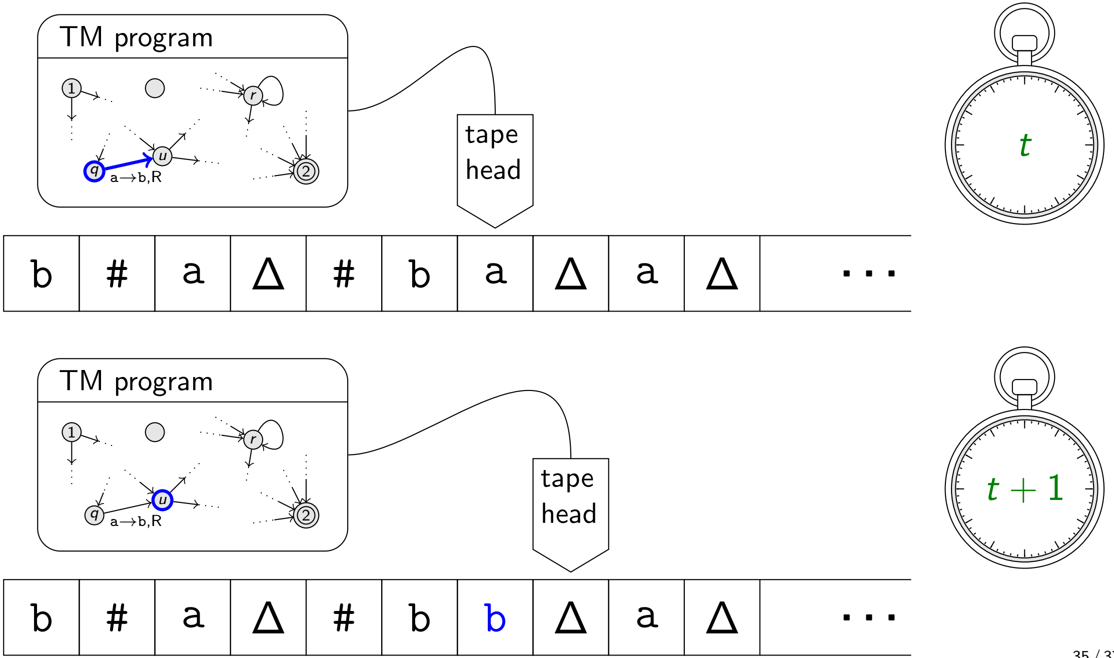
Things change according to transitions.

Then convert to CNF-clauses:

$$\begin{aligned}
&(\neg\bigcirc_{t,q} \vee \neg\square_{t,s,\ell} \vee \neg\text{⬠}_{t,s} \quad\vee\quad \bigcirc_{t+1,u})\\
&(\neg\bigcirc_{t,q} \vee \neg\square_{t,s,\ell} \vee \neg\text{⬠}_{t,s} \quad\vee\quad \square_{t+1,s,m})\\
&(\neg\bigcirc_{t,q} \vee \neg\square_{t,s,\ell} \vee \neg\text{⬠}_{t,s} \quad\vee\quad \text{⬠}_{t+1,s+\sigma})
\end{aligned}$$

. . . and combine them with conjunction:

$$\begin{aligned}
&\quad (\neg\bigcirc_{t,q} \vee \neg\square_{t,s,\ell} \vee \neg\text{⬠}_{t,s} \quad\vee\quad \bigcirc_{t+1,u})\\
\wedge&\quad (\neg\bigcirc_{t,q} \vee \neg\square_{t,s,\ell} \vee \neg\text{⬠}_{t,s} \quad\vee\quad \square_{t+1,s,m})\\
\wedge&\quad (\neg\bigcirc_{t,q} \vee \neg\square_{t,s,\ell} \vee \neg\text{⬠}_{t,s} \quad\vee\quad \text{⬠}_{t+1,s+\sigma})
\end{aligned}$$


TM program
1
r
u
q
2
a→b,R
tape head
b # a Δ # b a Δ a Δ ...
t
TM program
1
r
u
q
2
a→b,R
tape head
b # a Δ # b b Δ a Δ ...
$t+1$

Conclusion

φ_x := the conjunction of all the expressions we've made so far.

The algorithm that takes input x and constructs φ_x as above, $x \longmapsto \varphi_x$,
is our polynomial transformation from L to SATISFIABILITY.

$$x \in L \quad \text{if and only if} \quad \varphi_x \in \text{SATISFIABILITY}.$$

The construction can be done in polynomial time.

- lengthy, but routine, to prove.
- To gain insight on this:
  find upper bounds for # variables and # clauses created, in terms of n and k,
  where the Verifier TM's time complexity is $O(n^k)$.

$\square$

Revision

Things to think about:

- *One detail omitted:*
  Our construction assumes that the computation accepts *at* time $T(n)$.
  What if the TM accepts *before* time $T(n)$?
  We need to include some extra clauses in φ_x to deal with this. How?
- Now that we *know* that SATISFIABILITY is NP-complete,
  how can we use it to show that other problems are NP-complete,
  without going to the same amount of trouble all over again?
- How to show that SATISFIABILITY $\leq_P$ 3SAT ?

Reading:

- Sipser, section 7.4, pp. 304–311.
- M. R. Garey and D. S. Johnson,
  Computers and Intractability: A Guide to the Theory of NP-Completeness,
  W. H. Freeman & Co., San Francisco, 1979.
  See especially §2.6.