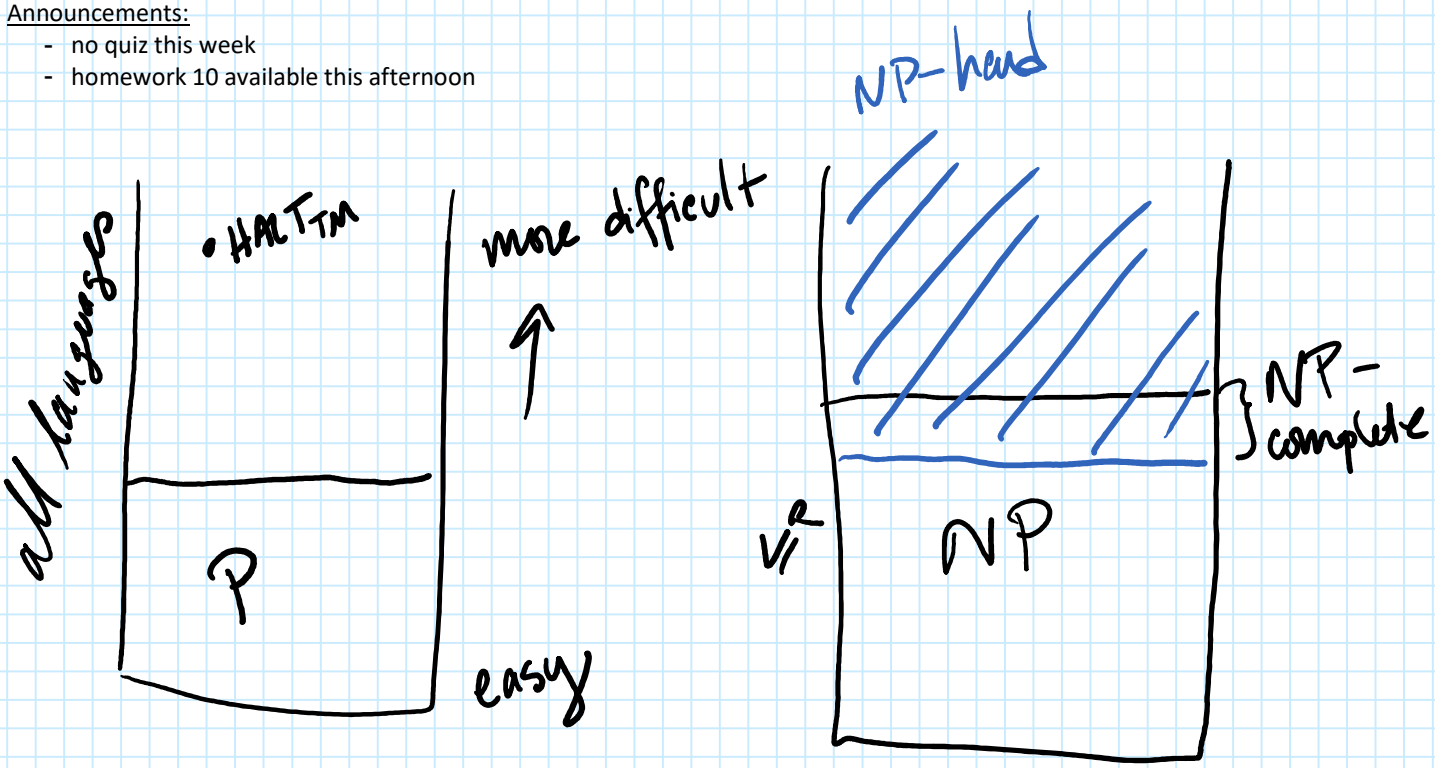
W13 L1 NP-completeness and Cook-Levin Theorem

Monday, April 27, 2020 9:06

Announcements:

- no quiz this week
- homework 10 available this afternoon



We want to combine diagrams with P & NP.

$A \leq_P B$

How do you prove a new problem B is NP-complete?

Need to show that $A \leq_P B$ for every $A \in NP$.

This seems challenging BUT

Thm: If $C \leq_P D$ and $D \leq_P E$ then $C \leq_P E$.

($\leq_P$ are transitive)

Alternately, need to show: $A \leq_P B$ for some NP-complete A

(Of course need $B \in NP$ as well.)

(Of course need $B \in NP$ as well.)

We need one NP-complete problem to get started.

Definitions:

- a **variable** is something like x, y, z (these are logical variables, so they can take on the values true/false)
- a **literal** is a variable or its negation, for example $x, \overline{x}$
- a **disjunction** is an "or" and we write it $\vee$, for example "x or y" is written $x \vee y$
- a **conjunction** is an "and" and we write it $\wedge$, for example "x and y" is written $x \wedge y$
- a **clause** is a disjunction of several literals, for example $(x \vee y \vee z)$, and $(x \vee \overline{x} \vee y \vee w)$ and (y)
- a formula is in **conjunctive normal form** if it is the conjunction of some clauses
  for example $(x \vee y) \wedge (z \vee \overline{x}) \wedge (w \vee y \vee \overline{z}) \wedge (y)$
- a **satisfying assignment** is a way to give each variable a Boolean value to make the overall expression evaluate to TRUE

Define $SAT = \{ \langle \varphi \rangle \mid \varphi$ is a conjunctive normal form formula and also φ has a satisfying assignment$\}$.

For example $(x) \wedge (\overline{x})$ is not in SAT, but $(x \vee y) \wedge (\overline{z})$ is in SAT.

Theorem: (Cook and Levin) SAT is NP-complete.

Proof:

Need to show that $SAT \in NP$ and also that for any $A \in NP, A \leq_p SAT$.

$SAT \in NP$: The certificate is the truth assignment, and we just check that the input is a properly-formatted formulae φ in conjunctive normal form and that the given truth assignment evaluates to TRUE.

Take any language $A \in NP$. We need to come up with a polynomial-time reduction $A \leq_p SAT$.

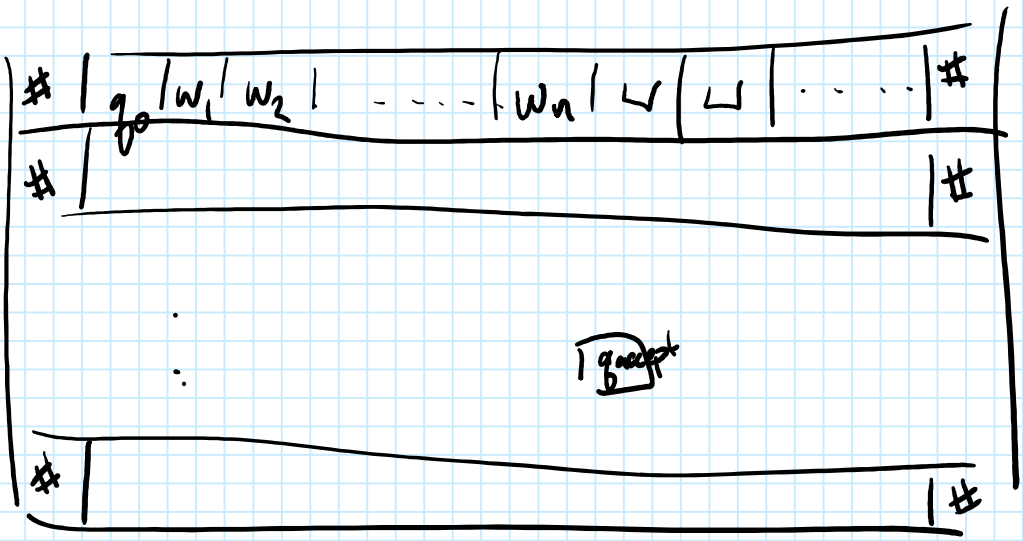
From the assumption that $A \in NP$, we know there is some nondeterministic polynomial-time TM N which decides A. Let's say that N runs in time n^k for some $k \in \mathbb{N}$. We know that on input string w, some branch of N's computation will accept string w.

On that branch, we will go through configurations of N:
first one is the starting configuration: $q_0 w$
next one will be some configuration we get to by following a transition of N based on the last configuration
next one will be some configuration we get to by following a transition of N based on the last configuration
next one will be some configuration we get to by following a transition of N based on the last configuration
next one will be some configuration we get to by following a transition of N based on the last configuration
...
next one is an accepting configuration where N is in state q_{accept}.

at most n^k

longest configuration has length at most n^k

Let's think about this accepting computation as a table of $n^k \times n^k$ many squares:



Now we know that $w \in A$ if and only if there is some table like the above, which has the following properties:

- it is $n^k \times n^k$
- the first row is $\#q_0 w$ (then blanks) #
- the i^{th} row is one step of computation after the $i-1^{th}$ row (or is exactly identical, if the TM has stopped computing) (see book)
- there is a row where we reached q_{accept}

Our goal is now to write a formula φ which is satisfiable if and only if this table can exist.
(Want that $w \in A$ if and only if $\varphi \in SAT$.)

We'll have a variable $x_{i,j,s}$ for every possible value $(i,j) \in [n^k] \times [n^k]$ and $s \in \Gamma \cup Q$.
We want is that if tape square i,j has character a in it, then $x_{i,j,a}$ =TRUE and $x_{i,j,b}$=FALSE for all other characters $b \in \Gamma$.

In order to enforce that every tape square has at least one character in it, we'll have a clause for each i,j pair which is: $(\bigvee_{s\in\Gamma} x_{i,j,s}) = (x_{i,j,a} \vee x_{i,j,b} \vee \cdots \vee x_{i,j,\#})$
This clause is satisfied when at least one of these variables is true (this place in the table has at least one character assigned to it).

In order to enforce that every tape square has at most one character in it, we'll have a clause for each i,j and each pair of different letters $a,b \in \Gamma \cup Q$: $(\overline{x_{i,j,a}} \vee \overline{x_{i,j,b}})$.
This clause is satisfied when at least one of a or b is NOT the letter assigned to tape square i,j.

Altogether, these two pieces enforce that every square of our table will have exactly one letter assigned to it.

} formatting

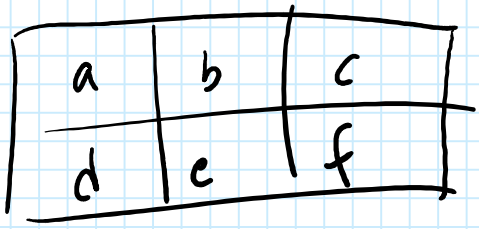
In order to enforce that the first row is $\#q_0 w$ (then blanks)#:
we add the clauses $(x_{1,1,\#}) \wedge (x_{1,2,q_0}) \wedge (x_{1,3,w_0}) \wedge \cdots \wedge (x_{1,n^k,\#})$.

} first row is # q_0 w ␣␣␣...#

In order to enforce that we eventually reach q_{accept} in the table, we need that somewhere, we have q_{accept} in a configuration.
We can add the clause $(\bigvee_{i,j} x_{i,j,q_{accept}})$.
This clause is satisfied when at least one square of the table, somewhere, is q_{accept}.

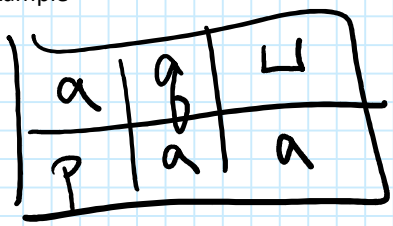
} we eventually reach q_{accept}

In order to enforce that each row of the table is one (or zero) step of computation after the last row, we need to look at "windows":



- always ok if a=d, b=e, c=f (no change)
- if they're different, what could have happened?
  One of a/b/c must have been a state, and in the bottom row, the state moved by at most one place, and maybe changed the tape.

For example



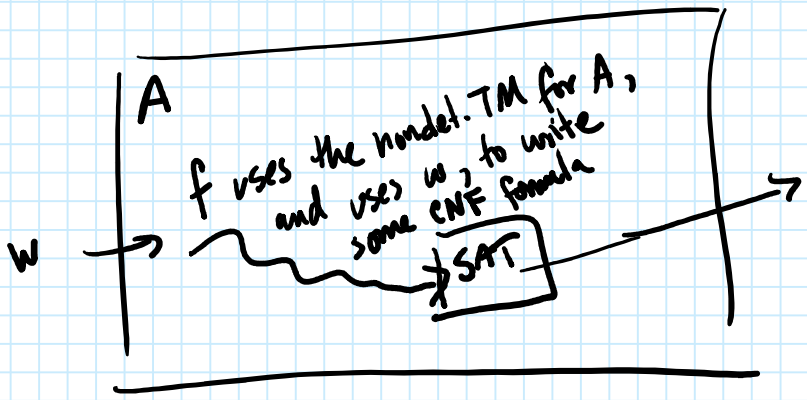
$\delta(q, \sqcup) = (p, a, L)$

overall idea: for every 2x3 window, come up with every possibly valid window (based on the transitions of the TM) and encode those as clauses.

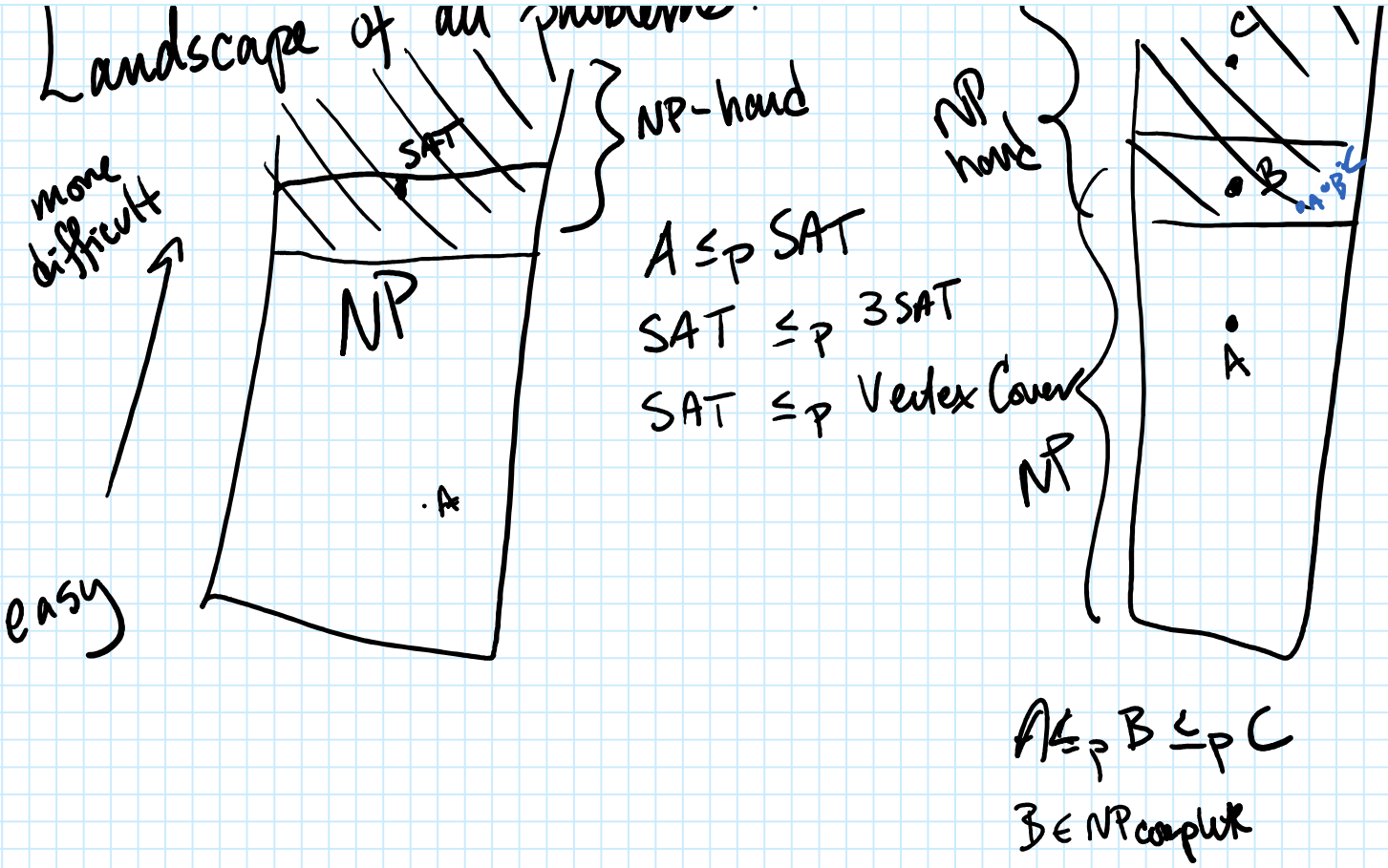
We'll add the clauses that enforce "these clauses are satisfiable if and only if every window is a valid window, based on the TM transitions".

(refer to the textbook for details of how exactly to write all these clauses)

Reduction overall



Landscape of all problems:


Landscape of all problems
more difficult
easy
SAT
NP
A
NP-hard
$A \leq_p SAT$
$SAT \leq_p 3SAT$
$SAT \leq_p$ Vertex Cover
NP hard
NP
C
B
A
$A \leq_p B \leq_p C$
$B \in$ NP complete