

LECTURE NOTES FOR CMSC 651
TURNING A NON CONSTRUCTIVE ALGORITHM INTO A CONSTRUCTIVE ONE

Def 0.1 Let g_{SAT} be the function that does the following: On input ϕ (a formula),

- If $\phi \notin SAT$ the $g_{SAT}(\phi) = NO$.
- If $\phi \in SAT$ then $g_{SAT}(\phi)$ is a satisfying assignment of ϕ .

The following lemma we leave to the reader.

Lemma 0.2 *There exists program $PROG_{SAT}$ for g_{SAT} that calls a subroutine for SAT $O(n)$ times (where n is the number of variables) and, other than those calls, takes $O(n^2)$ steps.*

THOUGHT EXPERIMENT ONE: I have the code for $PROG_{SAT}$ and I have code for an ALLEGED SAT algorithm. I run $PROG_{SAT}(\phi)$ using that ALLEGED SAT algorithm for the subroutine. I get an assignment \vec{b} . I evaluate $\phi(\vec{b})$ and it is TRUE. I now KNOW that $\phi \in SAT$ since I have a satisfying assignment for it. The alleged SAT algorithm may be wrong, but it does not matter—I definitely know that $\phi \in SAT$.

Theorem 0.3 *Let \preceq be a well quasi order on formulas such that the following is true.*

1. *The following problem is in polynomial time: given (ϕ, ψ) , determine if $\phi \preceq \psi$. We also assume we have the actual code for this.*
2. *If $\phi \in SAT$ and $\psi \preceq \phi$ then $\psi \in SAT$.*
3. *There exists a finite set OBS (called the obstruction set) such that*
 - (a) *Every element of OBS is in \overline{SAT} .*

(b)

$$x \in SAT \text{ iff } (\forall y \in OBS)[y \not\preceq x].$$

Then one can write down a polynomial time algorithm for SAT.

Proof:

THOUGHT EXPERIMENT TWO: ‘ You somehow know that $\psi \notin SAT$. You then find out that $\psi \preceq \phi$. You now KNOW that $\phi \notin SAT$ since, by the premise of the theorem, if $\phi \in SAT$ and $\psi \preceq \phi$, then $\psi \in SAT$.

The key to the algorithm is that we will try to get a set OBS' that is just as good as OBS for our purposes. We will then use g_{SAT} using the algorithm for SAT obtained by pretending that OBS' is really OBS .

Let $FML = \{\psi_1, \psi_2, \dots\}$ be a list of all formulas.

ALGORITHM FOR SAT

Input ϕ (We want to know if ϕ is satisfiable.)

KNOW=FALSE.

$i = 1$.

While KNOW=FALSE DO

 Test $\psi_1, \psi_2, \dots, \psi_i$ for membership in SAT by brute force. Let OBS_i be the subset of $\{\psi_1, \dots, \psi_i\}$ that were found to NOT be in SAT .

 For all $y \in OBS_i$ test $y \preceq \phi$. If there is some $y \in OBS_i$ such that $y \preceq \phi$ then we KNOW $\phi \notin SAT$. Set KNOW to TRUE and ANSWER to NO.

 (If you are at this state we know that $(\forall y \in OBS_i)[y \not\preceq \phi]$.)

 Pretend that

$$x \in SAT \text{ iff } (\forall y \in OBS_i)[y \not\preceq x].$$

 This yields a (possibly incorrect) algorithm for SAT. Denote this algorithm SAT_{OBS_i} .

 Compute $g_{SAT}(\phi)$ using SAT_{OBS_i} as the alleged SAT algorithm. Let the answer be \vec{b} .

 If $\phi(\vec{b}) = TRUE$ then

 KNOW=TRUE

 ANSWER=TRUE

 else $i = i + 1$

Output ANSWER

END OF ALGORITHM FOR SAT

Claim 1: If the algorithm outputs YES then $\phi \in SAT$.

Proof: When the algorithm outputs YES it has actually found a satisfying assignment. Hence clearly $\phi \in SAT$.

End of Proof of Claim 1

Claim 2: If the algorithm outputs NO then $\phi \notin SAT$.

Proof: When the algorithm outputs NO it has actually found a $y \notin SAT$ such that $y \preceq \phi$. Hence clearly $\phi \notin SAT$.

End of Proof of Claim 2

The next claim is not about the algorithm, but we still need it. We leave its proof to the reader.

Claim 3: If OBS' is the union of OBS and some formulas that are NOT in SAT then

$$x \in SAT \text{ iff } (\forall y \in OBS')[y \not\preceq x].$$

Claim 4: There is a constant M such that, for all ϕ , the loop never goes more than M iterations.

Proof: Let OBS be the OBS in the premise, the true obstruction set. Let M be the least numbers such that $OBS \subseteq \{\psi_1, \psi_2, \dots, \psi_M\}$. Note that OBS_M is the union of OBS and some formulas that are NOT in SAT .

By Claim 3

$$x \in SAT \text{ iff } (\forall y \in OBS_M)[y \not\preceq x].$$

Hence in the M th iteration the algorithm will (without knowing it) be using a correct subroutine for SAT , and hence will find the correct answer and output it.

End of Proof of Claim 4

It is easy to see that the algorithm is correct and runs in poly time. Note that we know the degree of the polynomial but we do not know the order constant.

■

Given this reasoning, how come SAT this does not prove SAT in P ? Because we do not know of an ordering and an OBS set as specified in the premise.

Then why did we go through this exercise? Because it shows a principle that CAN be applied to other problems. The only property of SAT that I used was Lemma 1. A similar lemma holds for VC_k , as you proved on the HW. Hence the nonconstructive algorithm shown for VC_k can be turned into a constructive one.

With this in mind, let us recap what we now know

Theorem 0.4 *The set of graphs under the ordering \preceq_{minor} is a well quasi order. (This is called the GRAPH MINOR THEOREM and was proven in a series of papers by Robertson and Seymour. It is difficult.)*

Theorem 0.5 *If (X, \preceq) is any WQO and $Z \subseteq X$ is such that $(a \in Z) \wedge (b \preceq a) \rightarrow b \in Z$ then there exists a finite set $OBS \subseteq \overline{Z}$ such that*

$$z \in Z \text{ iff } (\forall y \in OBS)[y \not\preceq z].$$

(This proof was easy but NONCONSTRUCTIVE. All we get is that OBS exists, but not how to find it.)

Theorem 0.6 *Fix k . There exists a finite set $OBS \subseteq \overline{VC_k}$ such that*

$$z \in VC_k \text{ iff } (\forall y \in OBS)[y \not\preceq VC_k].$$

(This follows easily from Theorem 0.5 above and the fact that VC_k is closed downward under minors.)

Theorem 0.7 *The following problem is in $O(n^3)$ and we have the code for it: Given (H, G) , determine if H is a minor of G .*

From Theorems 0.5 and 0.7 we can obtain that THERE EXISTS a poly time algorithm for VC_k that is in time $O(n^3)$: Given G , for every $H \in OBS$ test if H is a minor of G . Since we do not know OBS this algorithm as stated cannot be used. But we can STILL get code for this algorithm by Theorem 0.3.

Theorem 0.8 *Fix k . There is an $O(n^3)$ algorithm for VC_k that we can write down. (This follows from Theorems 0.6, 0.7, and the HW that will show that a form of Lemma 0.2 holds for VC_k .)*