Computability and Complexity
COSC 4200

Search versus Decision

We know that SAT is NP-complete, so $\mathrm{SAT} \in \mathrm{P}$ if and only if $\mathrm{P} = \mathrm{NP}$. The following theorem shows that finding satisfying assignments is also equivalent to $\mathrm{P} = \mathrm{NP}$.

Theorem

The following are equivalent.

1. $\mathrm{P} = \mathrm{NP}$
2. *There is a polynomial-time algorithm A that for any formula ϕ outputs a satisfying assignment if ϕ is satisfiable, or "unsatisfiable" if it is not.*

Proof. Assume there is a polynomial-time algorithm A that for any formula ϕ outputs a satisfying assignment if ϕ is satisfiable, or "unsatisfiable" if it is not.

Proof. Assume there is a polynomial-time algorithm A that for any formula ϕ outputs a satisfying assignment if ϕ is satisfiable, or "unsatisfiable" if it is not.

Then A can be used to show that $\mathrm{SAT} \in \mathrm{P}$. Either A outputs a satisfying assignment, at which point we decide "yes," or it says "unsatisfiable," and we decide "no."

For the other direction, assume $\mathrm{P} = \mathrm{NP}$. Then $\mathrm{SAT} \in \mathrm{P}$.
Consider the following algorithm:

For the other direction, assume $\mathrm{P} = \mathrm{NP}$. Then $\mathrm{SAT} \in \mathrm{P}$.
Consider the following algorithm:

Algorithm *A*

input ϕ

let n be the number of variables in ϕ

if $\phi \notin \mathrm{SAT}$, *then output "unsatisfiable"*

For the other direction, assume $\mathrm{P} = \mathrm{NP}$. Then $\mathrm{SAT} \in \mathrm{P}$. Consider the following algorithm:

Algorithm A

input ϕ

let n *be the number of variables in* ϕ

if $\phi \notin \mathrm{SAT}$, *then output "unsatisfiable"*

let $\phi_0 = \phi$

for $i = 1$ *to* n

For the other direction, assume $\mathrm{P} = \mathrm{NP}$. Then $\mathrm{SAT} \in \mathrm{P}$.
Consider the following algorithm:

Algorithm A

input ϕ
let n *be the number of variables in* ϕ

if $\phi \notin \mathrm{SAT}$, *then output "unsatisfiable"*

let $\phi_0 = \phi$
for $i = 1$ *to* n
    if $\phi_{i-1} \wedge (x_i) \in \mathrm{SAT}$
        $\phi_i = \phi_{i-1} \wedge (x_i)$
        $\tau(x_i) = T$

For the other direction, assume $\mathrm{P} = \mathrm{NP}$. Then $\mathrm{SAT} \in \mathrm{P}$. Consider the following algorithm:

Algorithm A

input ϕ
let n *be the number of variables in* ϕ

if $\phi \notin \mathrm{SAT}$, *then output "unsatisfiable"*

let $\phi_0 = \phi$
for $i = 1$ *to* n
  if $\phi_{i-1} \wedge (x_i) \in \mathrm{SAT}$
    $\phi_i = \phi_{i-1} \wedge (x_i)$
    $\tau(x_i) = T$
  else
    $\phi_i = \phi_{i-1} \wedge (\neg x_i)$
    $\tau(x_i) = F$

For the other direction, assume $\mathrm{P} = \mathrm{NP}$. Then $\mathrm{SAT} \in \mathrm{P}$. Consider the following algorithm:

Algorithm A

input ϕ

let n be the number of variables in ϕ

if $\phi \notin \mathrm{SAT}$, *then output "unsatisfiable"*

let $\phi_0 = \phi$

for $i = 1$ *to* n

    if $\phi_{i-1} \wedge (x_i) \in \mathrm{SAT}$

        $\phi_i = \phi_{i-1} \wedge (x_i)$

        $\tau(x_i) = T$

    else

        $\phi_i = \phi_{i-1} \wedge (\neg x_i)$

        $\tau(x_i) = F$

output τ

Since $\mathrm{SAT} \in \mathrm{P}$, this is a polynomial-time algorithm. Observe that the algorithm always outputs "unsatisfiable" for any formula $\phi \notin \mathrm{SAT}$. Assume that $\phi \in \mathrm{SAT}$. We claim that the assignment τ output by the algorithm satisfies ϕ.

Since $\mathrm{SAT} \in \mathrm{P}$, this is a polynomial-time algorithm. Observe that the algorithm always outputs "unsatisfiable" for any formula $\phi \notin \mathrm{SAT}$. Assume that $\phi \in \mathrm{SAT}$. We claim that the assignment τ output by the algorithm satisfies ϕ.

First we show by induction that for all i, $0 \leq i \leq n$, $\phi_i \in \mathrm{SAT}$. The base case is trivial: $\phi_0 = \phi$, so $\phi_0 \in \mathrm{SAT}$. Assume $\phi_i \in \mathrm{SAT}$. Then $\phi_i \wedge (x_{i+1}) \in \mathrm{SAT}$ or $\phi_i \wedge (\neg x_{i+1}) \in \mathrm{SAT}$, for a satisfying assignment of ϕ_i sets x_{i+1} to true or false. In either case, $\phi_{i+1} \in \mathrm{SAT}$.

Since $\mathrm{SAT} \in \mathrm{P}$, this is a polynomial-time algorithm. Observe that the algorithm always outputs "unsatisfiable" for any formula $\phi \notin \mathrm{SAT}$. Assume that $\phi \in \mathrm{SAT}$. We claim that the assignment τ output by the algorithm satisfies ϕ.

First we show by induction that for all i, $0 \leq i \leq n$, $\phi_i \in \mathrm{SAT}$. The base case is trivial: $\phi_0 = \phi$, so $\phi_0 \in \mathrm{SAT}$. Assume $\phi_i \in \mathrm{SAT}$. Then $\phi_i \wedge (x_{i+1}) \in \mathrm{SAT}$ or $\phi_i \wedge (\neg x_{i+1}) \in \mathrm{SAT}$, for a satisfying assignment of ϕ_i sets x_{i+1} to true or false. In either case, $\phi_{i+1} \in \mathrm{SAT}$.

Now we know that $\phi_n \in \mathrm{SAT}$. For each i, $1 \leq i \leq n$, let

$$l_i = \begin{cases} x_i & \text{if } \tau(x_i) = T \\ \neg x_i & \text{if } \tau(x_i) = F \end{cases}$$

Since $\mathrm{SAT} \in \mathrm{P}$, this is a polynomial-time algorithm. Observe that the algorithm always outputs "unsatisfiable" for any formula $\phi \notin \mathrm{SAT}$. Assume that $\phi \in \mathrm{SAT}$. We claim that the assignment τ output by the algorithm satisfies ϕ.

First we show by induction that for all i, $0 \leq i \leq n$, $\phi_i \in \mathrm{SAT}$. The base case is trivial: $\phi_0 = \phi$, so $\phi_0 \in \mathrm{SAT}$. Assume $\phi_i \in \mathrm{SAT}$. Then $\phi_i \wedge (x_{i+1}) \in \mathrm{SAT}$ or $\phi_i \wedge (\neg x_{i+1}) \in \mathrm{SAT}$, for a satisfying assignment of ϕ_i sets x_{i+1} to true or false. In either case, $\phi_{i+1} \in \mathrm{SAT}$.

Now we know that $\phi_n \in \mathrm{SAT}$. For each i, $1 \leq i \leq n$, let

$$l_i = \begin{cases} x_i & \text{if } \tau(x_i) = T \\ \neg x_i & \text{if } \tau(x_i) = F \end{cases}$$

Let $\psi = \bigwedge_{i=1}^{n}(l_i)$. Since $\phi_n = \phi \wedge \psi$ is satisfiable and the only satisfying assignment to ψ is τ, we have that τ satisfies ϕ_n. This means it must also satisfy ϕ. $\square$

To highlight how this works, consider the formula

$$\phi = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3)$$

Our decision algorithm for SAT will tell us that this is satisfiable.

To highlight how this works, consider the formula

$$\phi = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3)$$

Our decision algorithm for SAT will tell us that this is satisfiable.

So we set $\phi_0 = \phi$ and ask if

$$\phi_0 \wedge (x_1) = \phi \wedge (x_1) \in \mathrm{SAT}.$$

The algorithm says it is (which tells us that a satisfying assignment beginning with $x_1 = T$ exists), so we let $\phi_1 = \phi_0 \wedge (x_1)$ and $\tau(x_1) = T$.

For the next step, we ask if

$$\phi_1 \wedge (x_2) = \phi \wedge (x_1) \wedge (x_2) \in \mathrm{SAT}.$$

The algorithm says that this is not satisfiable, so we let $\phi_2 = \phi_1 \wedge (\neg x_2)$ and $\tau(x_2) = F$.

Finally, we ask if

$$\phi_2 \wedge (x_3) = \phi \wedge (x_1) \wedge (\neg x_2) \wedge (x_3) \in \mathrm{SAT}.$$

The algorithm says that this is satisfiable, so we let $\phi_3 = \phi_2 \wedge (x_3)$, and $\tau(x_3) = T$. It can be verified that the assignment τ we have obtained satisfies ϕ.

Finally, we ask if

$$\phi_2 \wedge (x_3) = \phi \wedge (x_1) \wedge (\neg x_2) \wedge (x_3) \in \mathrm{SAT}.$$

The algorithm says that this is satisfiable, so we let $\phi_3 = \phi_2 \wedge (x_3)$, and $\tau(x_3) = T$. It can be verified that the assignment τ we have obtained satisfies ϕ.

$$\phi = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3)$$

$$\tau(x_1) = T, \tau(x_2) = F, \tau(x_3) = T$$