The Cook-Levin Theorem

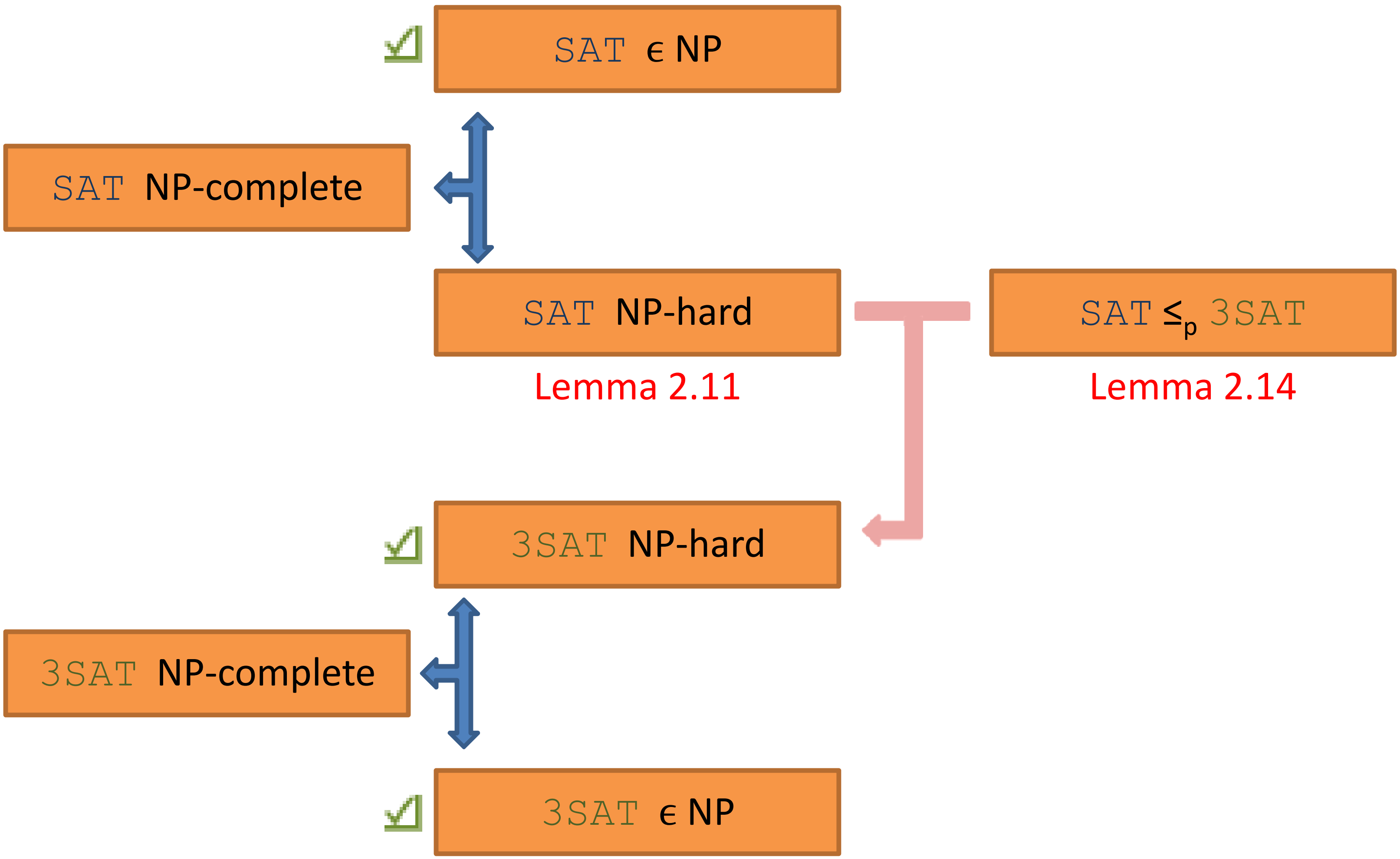
CS468/568

Author: David Costanzo, Yale University
Adapted from Computational Complexity: A Modern Approach by Arora and Barak

The Claim

- Claim 1: `SAT` is NP-complete
- Claim 2: `3SAT` is NP-complete

Proof Structure



Lemma 2.14: $\mathtt{SAT} \leq_p \mathtt{3SAT}$

- Convert each CNF clause of size $\mathtt{k}$ to equivalent clauses of sizes $\mathtt{k-1}$ and $\mathtt{3}$
  - $\longrightarrow$ repeat until all clauses are size $\mathtt{3}$

$$C = u_1 \vee u_2 \vee \cdots \vee u_k$$

$$C_1 = u_1 \vee u_2 \vee y$$ (fresh variable)

$$C_2 = u_3 \vee \cdots \vee u_k \vee \overline{y}$$

$$C \in SAT \Leftrightarrow C_1 \wedge C_2 \in SAT$$

Lemma 2.11: `SAT` is NP-hard

- Pick any `L` ϵ `NP`, and let M be a poly-time TM recognizing `L` on any input x and valid certificate `u`
- Assumptions:
  - M has 1 input tape and 1 work/output tape
  - M is oblivious

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} .\ M(x \cdot u) = 1$$

Lemma 2.11: SAT is NP-hard

- Need to describe a poly-time function mapping input x of length n to formula φ_x such that:

$$x \in L \Leftrightarrow \varphi_x \in SAT$$

Lemma 2.11: SAT is NP-hard

- Need to describe a poly-time function mapping input x of length n to formula φ_x such that:

$$x \in L \Leftrightarrow \varphi_x \in SAT$$

Or equivalently:

$$\exists u \in \{0,1\}^{p(|x|)} .\ M(x \cdot u) = 1 \Leftrightarrow \varphi_x \in SAT$$

Lemma 2.11: SAT is NP-hard

- Variables of φ_x:
  - y_1, y_2, …, y_n (first n bits of input $x \cdot$ u)
  - y_{n+1}, y_{n+2}, …, $y_{n+p(n)}$ (next p(n) bits of input $x \cdot$ u)
  - z_1, z_2, …, $z_{T(n)}$ (*snapshots* of TM M on input $x \cdot$ u)

(bit strings of length c)

Lemma 2.11: SAT is NP-hard

- Variables of φ_x:
  - $y_1, y_2, \ldots, y_n$ (first n bits of input $x \cdot$ u)
  - $y_{n+1}, y_{n+2}, \ldots, y_{n+p(n)}$ (next p(n) bits of input $x \cdot$ u)
  - $z_1, z_2, \ldots, z_{T(n)}$ (*snapshots* of TM M on input $x \cdot$ u)

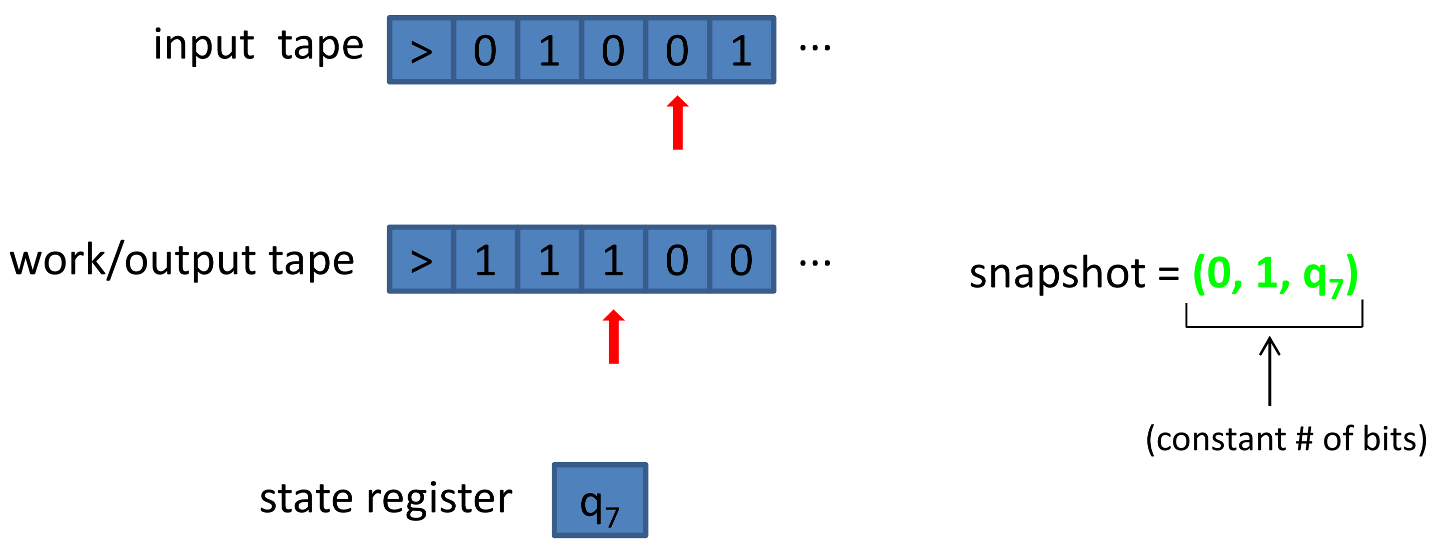
    (bit strings of length c)

- The goal:

  The formula φ_x is satisfied iff the input snapshots represent a valid execution of M on input y = $x \cdot$ u

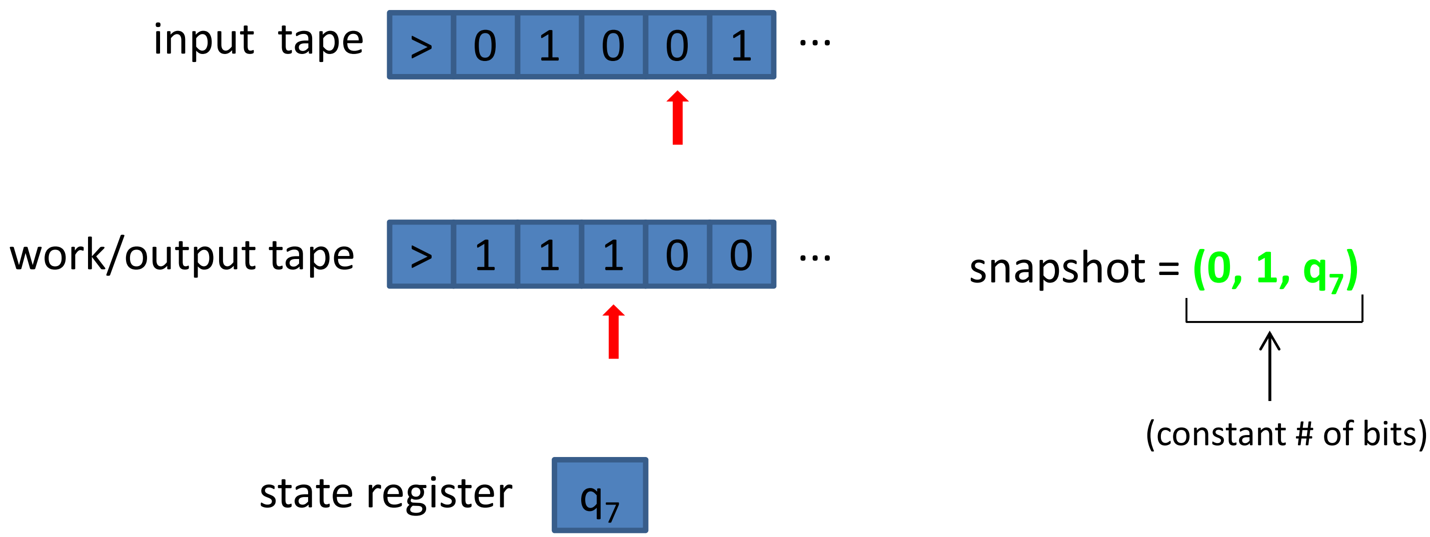
Lemma 2.11: SAT is NP-hard

- What is a snapshot?



Lemma 2.11: `SAT` is NP-hard

- What is a snapshot?



- How to determine the correct snapshot at time i?

Lemma 2.11: `SAT` is NP-hard

- Given input y and all snapshots $z_1, \ldots, z_{i-1}$, there exists *at most one* valid snapshot z_i
- This z_i depends *only* on z_{i-1}, $z_{prev(i)}$, and $y_{inputpos(i)}$
  - prev(i) = the most recent step preceding i at which the work/output head was at the same position as at step i
  - inputpos(i) = the position of the input head at step i
  - These are well-defined by obliviousness, and poly-time computable by simulating M on dummy input $0^{|x|}$

$$z_i = F(z_{i-1}, z_{prev(i)}, y_{inputpos(i)})$$

$$F: \{0,1\}^{2c+1} \rightarrow \{0,1\}^c$$

Lemma 2.11: SAT is NP-hard

$$\varphi_x = A \wedge B \wedge C \wedge D$$

- A : the first n bits of y are equal to the bits of x

  $(x_1 \vee \overline{y_1}) \wedge (\overline{x_1} \vee y_1) \wedge \cdots \wedge (x_n \vee \overline{y_n}) \wedge (\overline{x_n} \vee y_n)$

- B : z_1 correctly encodes the initial snapshot of M
- C : $z_{T(n)}$ correctly encodes a halting snapshot of M from which M outputs 1 (i.e. accept)
- D: For each i between 2 and T(n)-1,

  $z_i = F(z_{i-1}, z_{prev(i)}, y_{inputpos(i)})$

Lemma 2.11: SAT is NP-hard

How to express $z_i = F(z_{i-1}, z_{prev(i)}, y_{inputpos(i)})$ in CNF?

$$f: \{0,1\}^l \rightarrow \{0,1\}$$

$$\psi_x = \bigwedge_{v:f(v)=0} C_v(x_1, \dots, x_l)$$

$$C_{1101} = \overline{x_1} \vee \overline{x_2} \vee x_3 \vee \overline{x_4}$$

$$C_{00110} = x_1 \vee x_2 \vee \overline{x_3} \vee \overline{x_4} \vee x_5$$

etc...

$$\psi_x \in SAT \Leftrightarrow f(x) = 1$$

QED