CS F364
Design & Analysis of Algorithms

ALGORITHMS - COMPLEXITY

Complexity Class NP-complete

- Definition
- Circuit-SAT is NP-complete

COMPLEXITY CLASSES – ℕℙ-COMPLETE

- A problem is said to be **ℕℙ-hard** if all problems in ℕℙ (polynomially) reduce to it
  - i.e. π is ℕℙ-hard if
    - for each π' in ℕℙ: $\pi' \precsim \pi$
- A problem π is said to be **ℕℙ-complete**
  - if π is ℕℙ-hard and
  - if π is in ℕℙ.

PROBLEMS – COMPLEXITY – CIRCUIT-SAT

- A Boolean (combinational) circuit can be modeled as a directed graph:
  - Gates are vertices
  - Input lines to a gate are incoming edges and are labeled with input values
  - Outputs of a gate driven to input lines of other gates are outgoing edges
    - All outgoing edges of a given gate have the same label.
- For simplicity, we assume:
  - Gates are unary/binary i.e. at most two inputs per gate
  - The final output is not driven.
    - i.e. Gates forming the final outputs of the circuit have no outgoing edges.

PROBLEMS – COMPLEXITY – CIRCUIT-SAT [2]

- Definition:
  - Given a Boolean circuit (modeled as a graph as mentioned in the previous slide),
  - find whether there exists a sequence of input values such that the result is an output of 1.
- Cook-Levin Theorem :
  - CIRCUIT-SAT is $\mathbb{NP}$-complete.
- Proof:
  - CIRCUIT-SAT is in $\mathbb{NP}$. *(Lemma 1)* and
  - For any problem π in $\mathbb{NP}$, π reduces in polynomial time to CIRCUIT-SAT. *(Lemma 2)*

Problems – Complexity – Circuit-SAT Is In $\mathbb{NP}$

- Lemma 1:
  - CIRCUIT-SAT is in $\mathbb{NP}$.
- Proof:
  - Given a certificate
    - i.e. a combination of Boolean values assigned to the input variables,
  - it can be verified whether the circuit outputs 1
    - by executing the circuit
  - in time (linearly) proportional to the number of gates (i.e. input size).

Problems – Complexity – Circuit-SAT Is In ℕℙ- Hard

- Lemma 2:

  For any problem π in ℕℙ, π reduces to CIRCUIT-SAT.

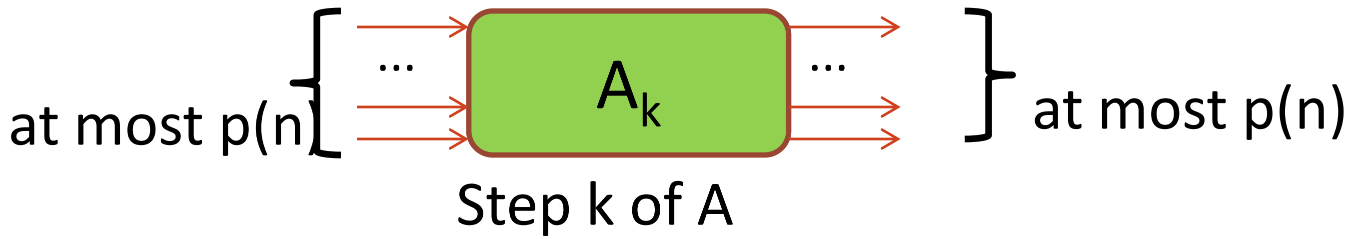
- Proof:
  - Consider any problem π in ℕℙ.
    - We need to reduce π to **CIRCUIT-SAT** i.e.
      - we need a polynomial-time mapping:
        - *that converts an (input) instance x of π to a Boolean combinational circuit S*
      - such that
        - π(x) is 1 iff S is satisfiable
  - We design such a mapping in the following slides.

REDUCING ANY $\mathcal{NP}$ PROBLEM TO **CIRCUIT-SAT**

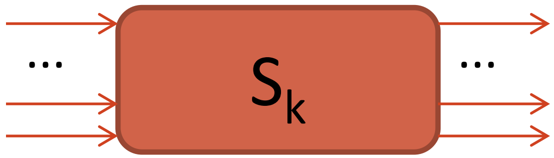
- Reduction:
  - Consider any problem π in $\mathbb{NP}$.
    - There exists a deterministic algorithm $\mathbf{A}_\pi$ for verifying a certificate c(x) of any input x of size n for π such that
      - $\mathbf{A}_\pi$ takes p'(n) time where p' is a polynomial function.
    - If $\mathbf{A}_\pi$ takes p(n) time, it will be using
      - at most p(n) bits of space at any step of the computation
        - where p(n) = c*p'(n) for some constant c
  - We design a mapping procedure, say ***cc*** (for *circuit constructor*):
    - that takes $\mathbf{A}_\pi$ and constructs a Boolean combinational circuit **S** in polynomial time such that
      - π(x) is 1 iff **S** is satisfiable

DESIGN OF A CIRCUIT CONSTRUCTOR

- cc(A):
  - Consider one step of computation in A:
    - at most p(n) bits of input and at most p(n) bits of output

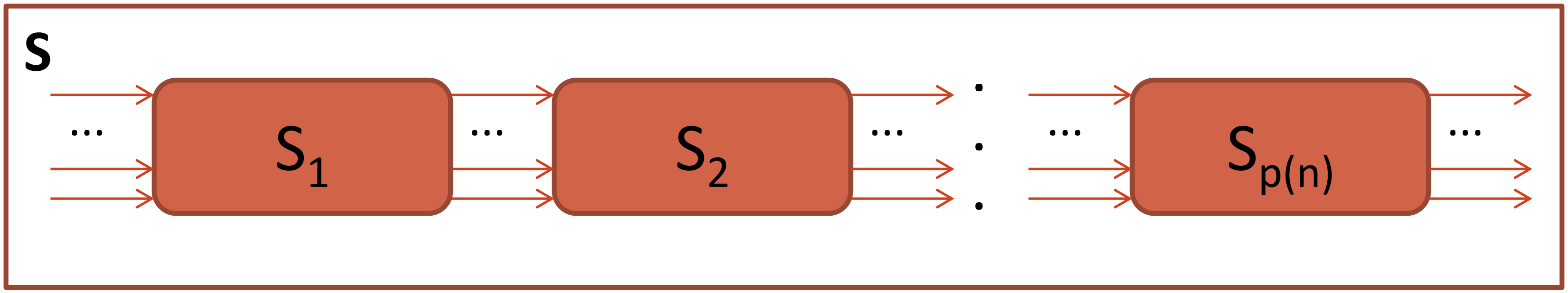


  - Construct a circuit S_k to implement the single step k of A
    - as a combinational circuit mapping the input (of a step) to the output (of a step)



DESIGN OF A CIRCUIT CONSTRUCTOR CONTINUED

- cc(A):
  - Construct a circuit S which is a cascade of S_k circuits – p(n) of them – such that the output of j^{th} circuit is the input of $(j+1)^{st}$ circuit.



  - S' is the circuit implementing algorithm A i.e. it has 2 inputs:
    - x (which is an input instance of problem π) and
    - c(x) (which is a certificate for x)
  - and it produces a 1 if the certificate is valid.

This ensures that <u>S' is satisfiable iff $\pi(x)$ is 1</u>

Problems – Complexity – Circuit-SAT Is In NP- Hard

- Proof (of Lemma 2) - continued:
  - Verify these claims:
    - Circuit S’ is satisfied if and only if there exists a certificate y such that A(x,y) = 1.
    - Time complexity of cc is polynomial in n, where n is the size of the input instance to π.