NP-completeness

2 perspectives:

1. To show that some problems are (probably) hard to solve
2. A rich topic in theoretical computer science

To show a language is in NP:

2-input verification algorithms with polynomial running time

Examples:

L_{HAM} = {G | G has a Hamiltonian cycle}

$L_{COMPOSITE}$ = {n | n is a composite number}

What about

L_{PRIME} = {n | n is a prime number}?

$L_{NOT\text{-}HAM}$ = {G | G does <u>not</u> have a Hamiltonian cycle}?

A language L is in the complexity class **co-NP** if its complement is is NP

NP-completeness

The circuit satisfiability problem CIRCUIT-SAT

A combinatorial circuit of and/ or/ not gates

Represented as a directed acyclic graph

$L_{CIRCUIT\text{-}SAT}$ = {C | C is a satisfiable combinatorial circuit}

Lemma 34.5. $L_{CIRCUIT\text{-}SAT}$ is in NP

Lemma 34.6. $L_{CIRCUIT\text{-}SAT}$ is NP-hard

Proof - the standard reduction

Let L be any language in NP, accepted by the 2-input verification algorithm A(x,y)

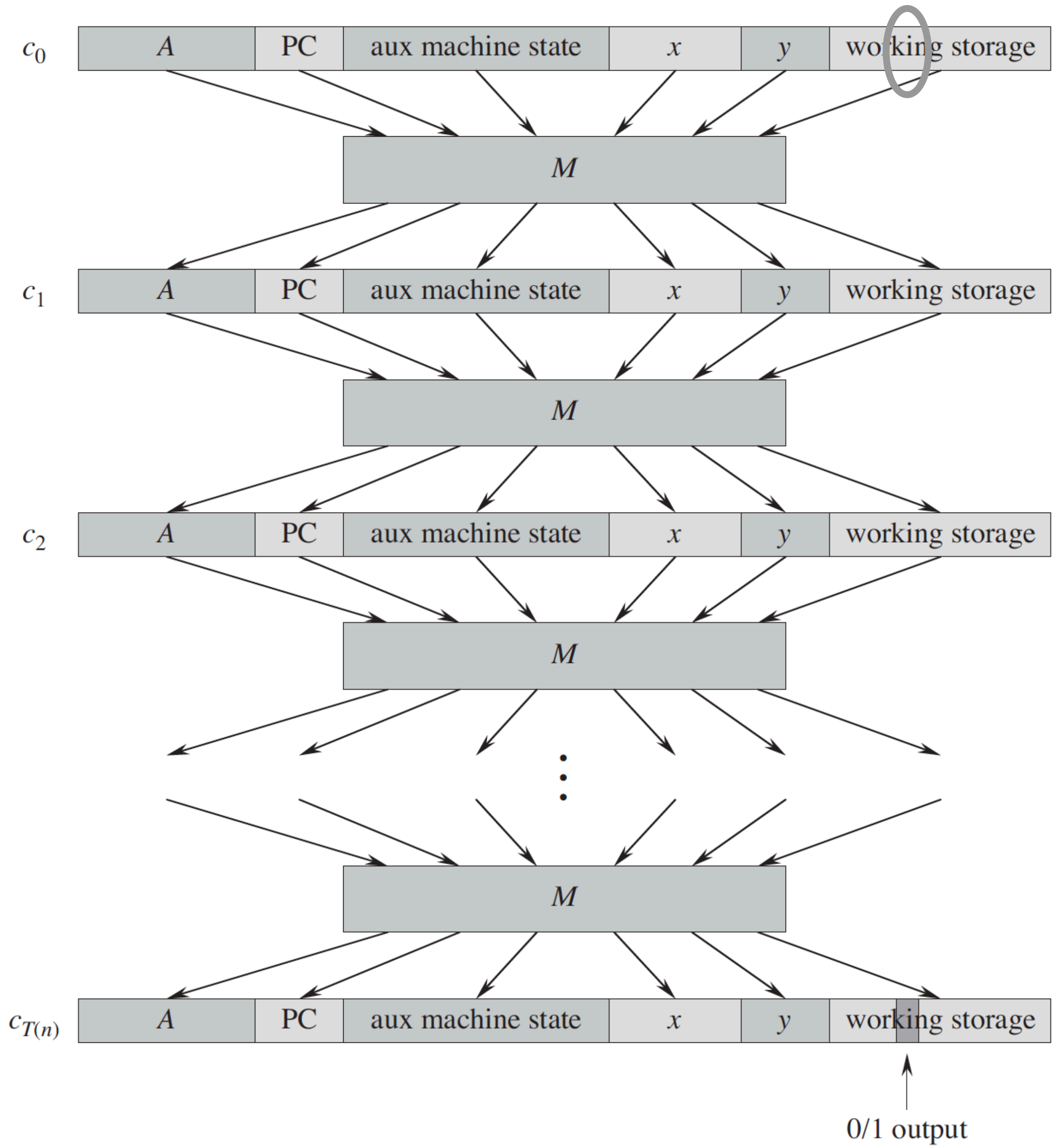
Given any input x (Is x in L?)

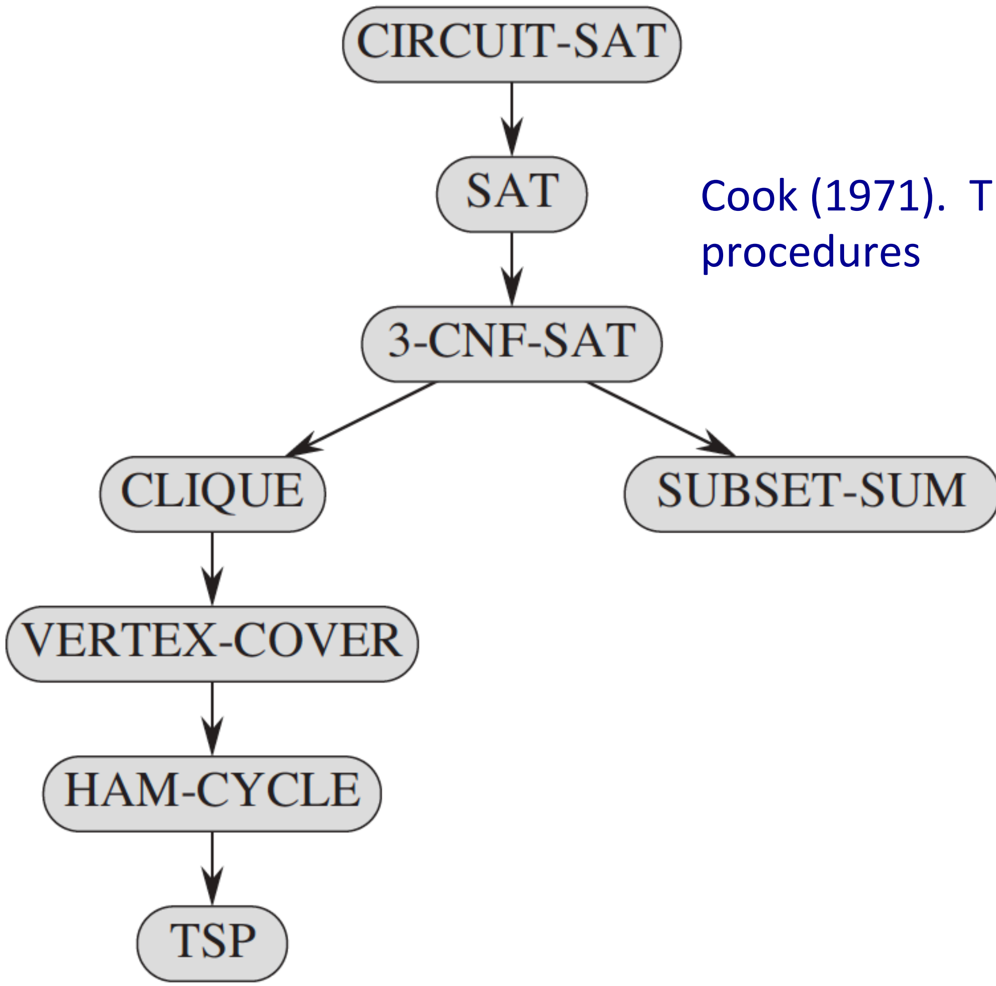
-Compute f(|x|), where f(|x|) is the running time of A(x,y)

-Make f(|x|) copies of the comb circuit of a computer, and f(|x|)+1 copies of the memory of the computer, and wire the copies together

-Initialize the first memory copy to A, x, and y

-Ignore all bits except the output bit on the f(|x|)+1 'th memory copy


c_0
A
PC
aux machine state
x
y
working storage
M
c_1
A
PC
aux machine state
x
y
working storage
M
c_2
A
PC
aux machine state
x
y
working storage
M
M
$c_{T(n)}$
A
PC
aux machine state
x
y
working storage
0/1 output



Cook (1971). The complexity of theorem-proving procedures