

# Recall From Last Time

The following problems are **NP** complete:

- 3-SAT
- 3-COLOR
- CLIQUE
- SUBSET SUM
- KNAPSACK

**Working assumption:** 3-SAT requires  $2^{\epsilon n}$  time.

When we reduce 3-SAT directly to problem  $A$ , we can see what makes  $A$  hard.

# Alternation

The concept of a nondeterministic acceptor of a boolean query has a long and rich history, going back to various kinds of nondeterministic automata.

It is important to remember that these are **fictitious machines**: we suspect that they cannot be built.

**Open question:**  $\mathbf{NP} \stackrel{?}{=} \mathbf{co-NP} = \{\overline{A} \mid A \in \mathbf{NP}\}$

If one could really build an **NP** machine, then one could, with a single gate to invert its answer, also build a **co-NP** machine.

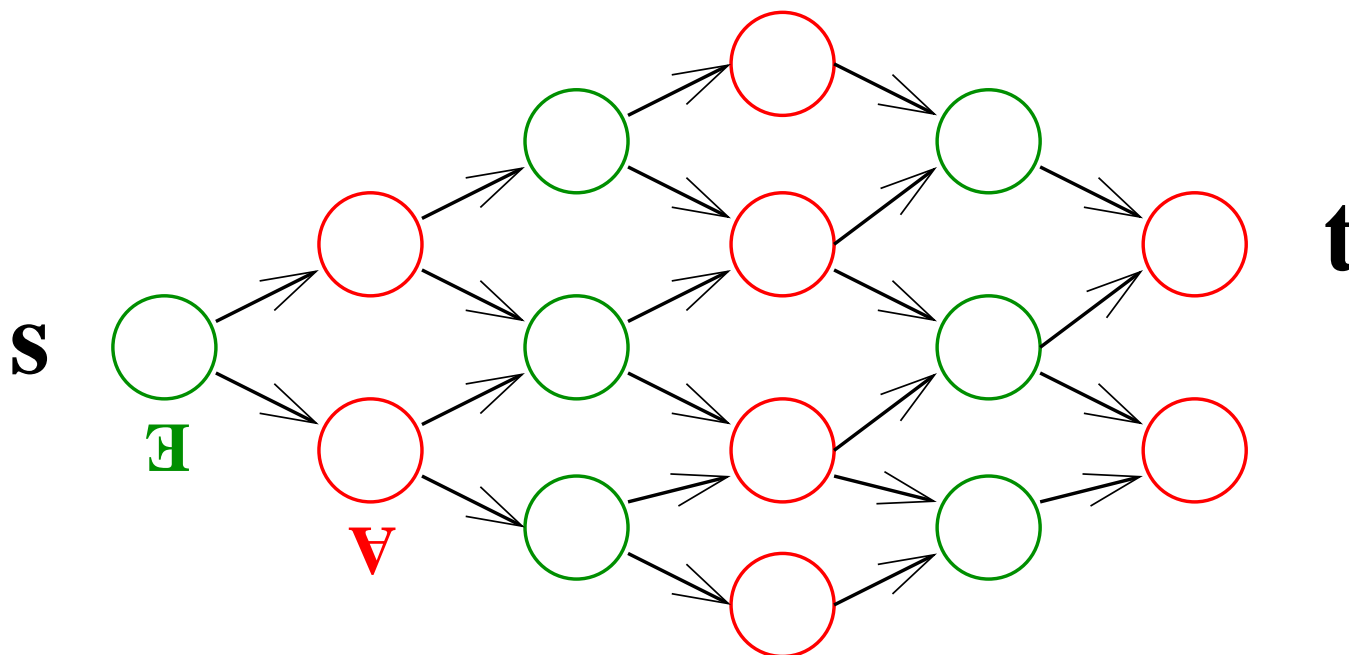
From a practical point of view, the complexity of a problem  $A$  and its complement,  $\overline{A}$  are identical.



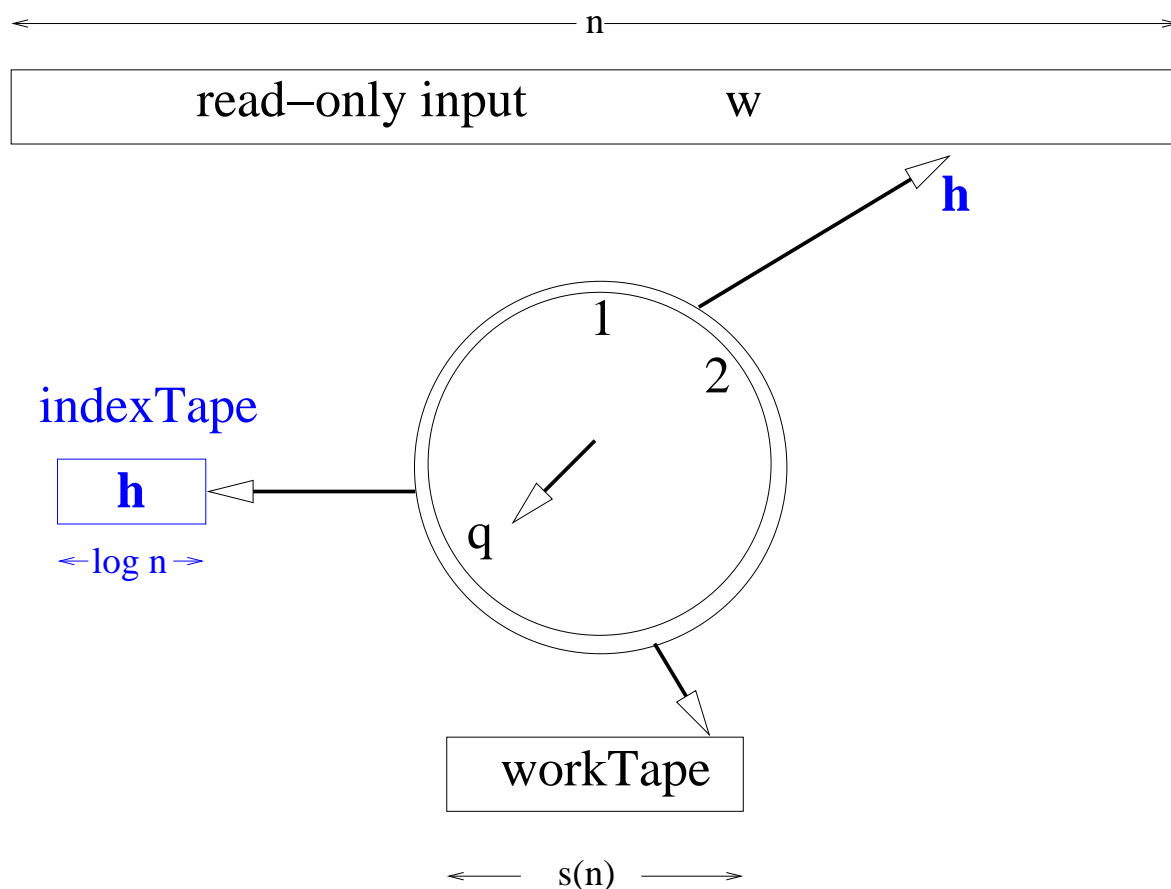
The states of an **alternating Turing machine** is split into:  
**Existential states** ( $\exists$ ) and **Universal states** ( $\forall$ ).

**Def:** An alternating TM in  $ID_0$  **accepts** iff

1.  $ID_0$  is in a final accepting state, or
2.  $ID_0$  is in  $\exists$  state and some next  $ID'$  accepts, or
3.  $ID_0$  is in  $\forall$  state, exists at least one next  $ID$ , and all next  $ID$ 's accept.



From now on assume that our Turing machines have a **random access** read-only input. There is an **index tape** which can be written on and read like other tapes. Whenever the value  $h$ , written in binary, appears on the index tape, the read head will automatically scan bit  $h$  of the input.



**Def:**  $\mathbf{ASPACE}[s(n)]$  and  $\mathbf{ATIME}[t(n)]$  to be the set of problems accepted by alternating TM's using  $O(s(n))$  tape cells,  $O(t(n))$  time, respectively, in any computation path on any input of length  $n$ .

**Alternation Thm:** For  $s(n) \geq \log n$ , and for  $t(n) \geq n$ ,

$$\bigcup_{k=1}^{\infty} \mathbf{ATIME}[(t(n))^k] = \bigcup_{k=1}^{\infty} \mathbf{DSPACE}[(t(n))^k]$$

$$\mathbf{ASPACE}[s(n)] = \bigcup_{k=1}^{\infty} \mathbf{DTIME}[k^{s(n)}]$$

**Cor:**  $\mathbf{ASPACE}[\log n] = \mathbf{P}$

$$\mathbf{ATIME}[n^{O(1)}] = \mathbf{PSPACE}$$

**Def:** the **monotone, circuit value problem (MCVP)** is the subset of **CVP** in which no negation gates occur.

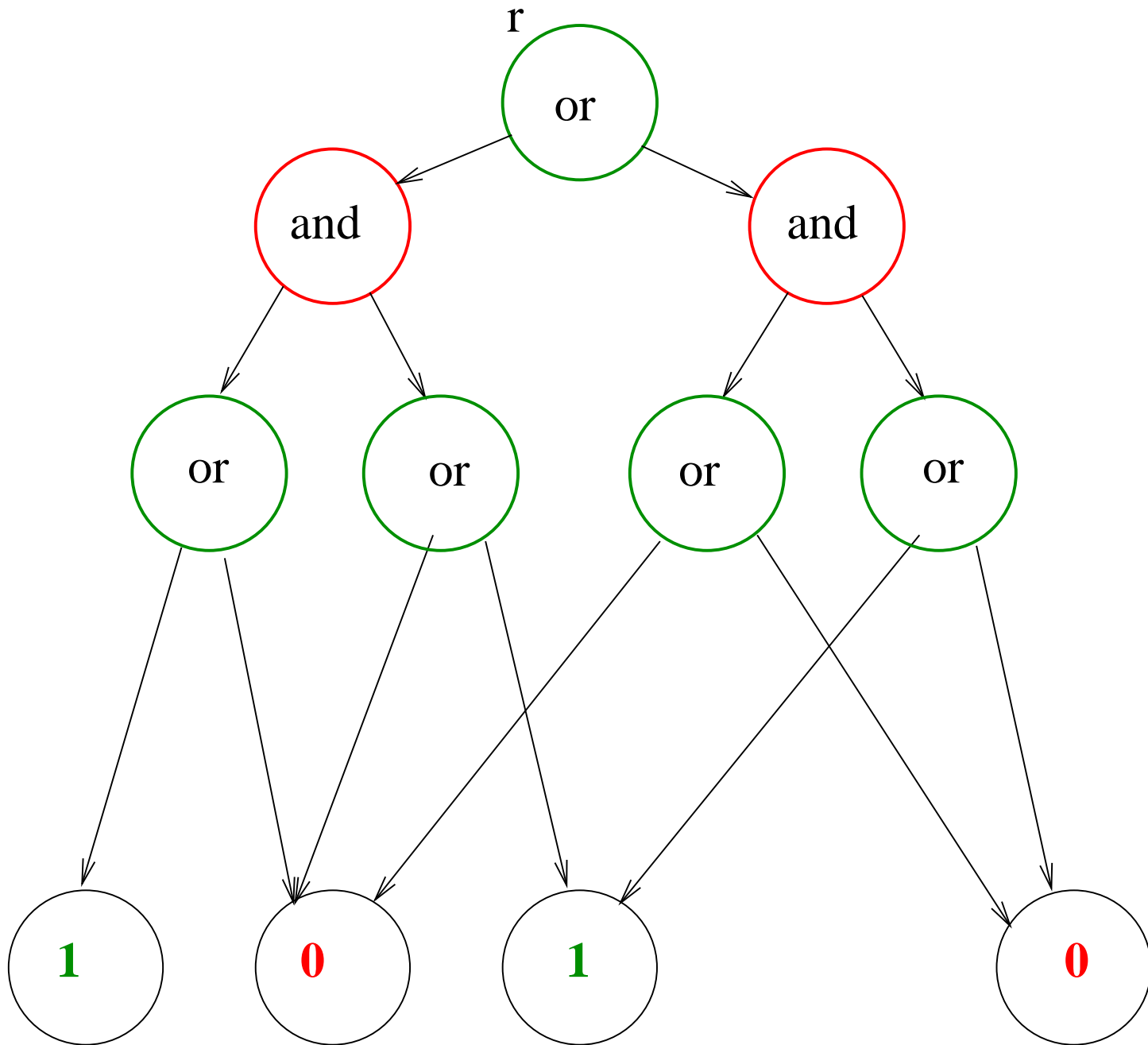
**Prop:** **MCVP** is recognizable in **ASPACE** $[\log n]$ .

**Proof:** Let  $G$  be a monotone boolean circuit. For  $a \in V^G$ , define “**EVAL**( $a$ )”,

1. **if** (InputOn( $a$ )) **then accept**
2. **if** (InputOff( $a$ )) **then reject**
3. **if** ( $G_{\wedge}(a)$ ) **then** **universally** choose child  $b$  of  $a$
4. **if** ( $G_{\vee}(a)$ ) **then** **existentially** choose child  $b$  of  $a$
5. Return(**EVAL**( $b$ ))

$M$  simply calls **EVAL**( $r$ ). **EVAL**( $a$ ) returns “**accept**” iff gate  $a$  evaluates to one.

Space used for naming vertices  $a, b$ :  $O(\log n)$ . □



**Def:** The **quantified satisfiability problem (QSAT)** is the set of true formulas of the following form:

$$\Psi = Q_1 x_1 Q_2 x_2 \cdots Q_r x_r (\varphi)$$

For any boolean formula  $\varphi$  on variables  $\bar{x}$ ,

$$\varphi \in \text{SAT} \iff \exists \bar{x} (\varphi) \in \text{QSAT}$$

$$\varphi \notin \text{SAT} \iff \forall \bar{x} (\neg \varphi) \in \text{QSAT}$$

Thus QSAT logically contains SAT and  $\overline{\text{SAT}}$ .

**Prop:** QSAT is recognizable in  $\mathbf{ATIME}[n]$ .

**Proof:** Construct ATM,  $A$ , on input,  $\Phi \equiv$

$$\begin{array}{ccccccc} \exists x_1 & \forall x_2 & \cdots & \exists x_{2k-1} & \forall x_{2k} & \bigwedge_{i=1}^r & \bigvee_{j=1}^s & \ell_{ij} \\ & & & & & i & j & \ell_{ij}(b_1, \dots, b_{2k}) \end{array}$$

**Quantifiers:**

- in  $\exists$  state,  $A$  writes a bit  $b_1$  for  $x_1$ ,
- in  $\forall$  state,  $A$  writes a bit  $b_2$  for  $x_2$ , and so on.

**Boolean operators:**

- in  $\forall$  state,  $A$  chooses  $i$ ,
- in  $\exists$  state,  $A$  chooses  $j$

**Final state:** accept iff  $\ell_{ij}(b_1, \dots, b_{2k})$  is true.

$A$  accepts  $\Phi \iff \Phi$  is true.



**Thm:** For any  $s(n) \geq \log n$ ,

$$\mathbf{NSPACE}[s(n)] \subseteq \mathbf{ATIME}[s(n)^2] \subseteq \mathbf{DSPACE}[s(n)^2]$$

**Proof:**  $\mathbf{NSPACE}[s(n)] \subseteq \mathbf{ATIME}[s(n)^2]$ :

Let  $N$  be an  $\mathbf{NSPACE}[s(n)]$  Turing machine.

Let  $w$  be an input to  $N$ ,  $n = |w|$ .

$$w \in \mathcal{L}(N) \iff \text{CompGraph}(N, w) \in \mathbf{REACH}$$

$w \in \mathcal{L}(N) \iff \text{CompGraph}(N, w) \in \text{REACH}$

$P(d, x, y) \equiv$  “ In  $\text{CompGraph}(N, w)$ ,  $\text{dist}(x, y) \leq 2^d$  ”

$P(d, x, y) \equiv \exists z (P(d - 1, x, z) \wedge P(d - 1, z, y))$

1. **Existentially:** choose middle ID  $z$ .
2. **Universally:**  $(x, y) := (x, z) \ \& \ (z, y)$
3. Return( $P(d - 1, x, y)$ )

$$T(d) = O(s(n)) + T(d - 1) = O(d \cdot s(n))$$

$$d = O(s(n))$$

$$T(d) = O((s(n))^2)$$

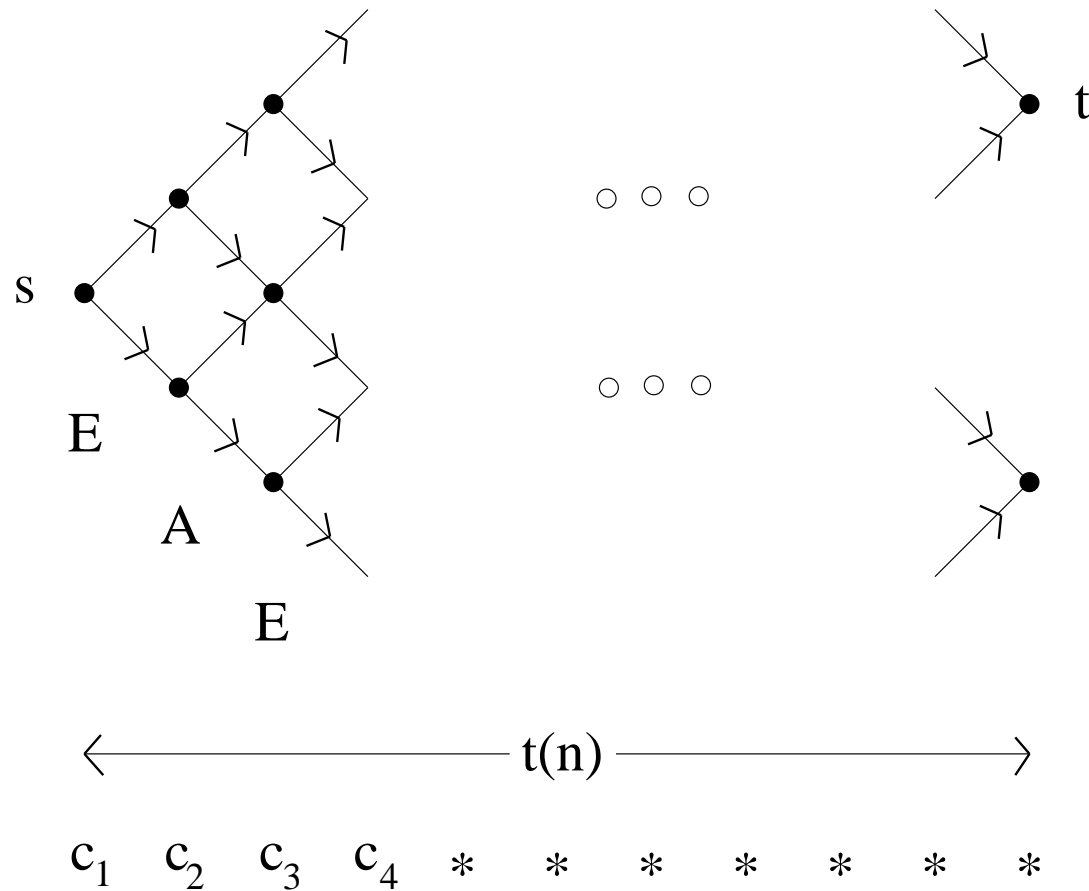
$$\mathbf{ATIME}[t(n)] \subseteq \mathbf{DSPACE}[t(n)]$$

Let  $A$  be an  $\mathbf{ATIME}[t(n)]$  machine, input  $w$ ,  $n = |w|$ .

$\text{CompGraph}(A, w)$  has depth  $c(t(n))$  and size  $2^{c(t(n))}$ , for some constant  $c$ .

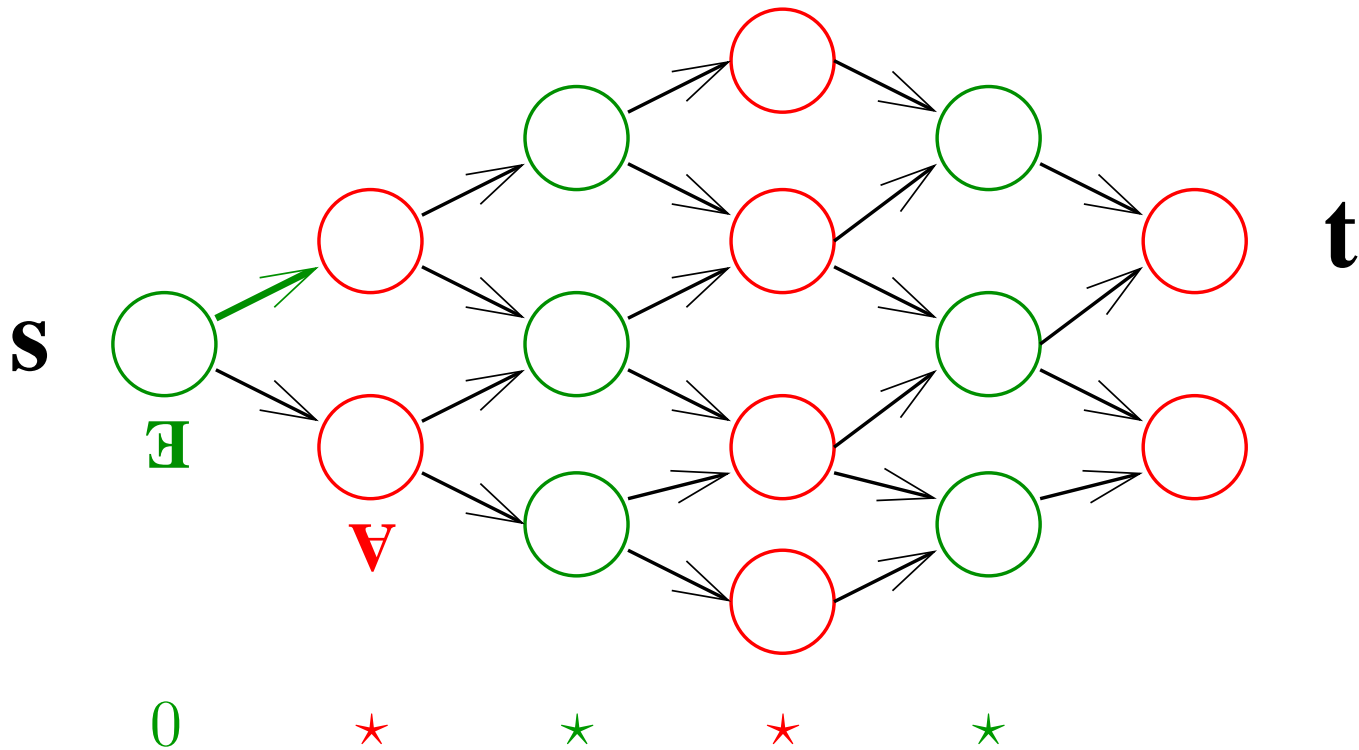
Search this and/or graph systematically using  $c(t(n))$  extra bits of space.

$$\mathbf{ATIME}[t(n)] \subseteq \mathbf{DSPACE}[t(n)]$$



Evaluate computation graph of  $\text{ATIME}[t(n)]$  machine using  $t(n)$  space to cycle through all possible computations of  $A$  on input  $w$ .

**Example:**  $\text{ATIME}[t(n)] \subseteq \text{DSPACE}[t(n)]$



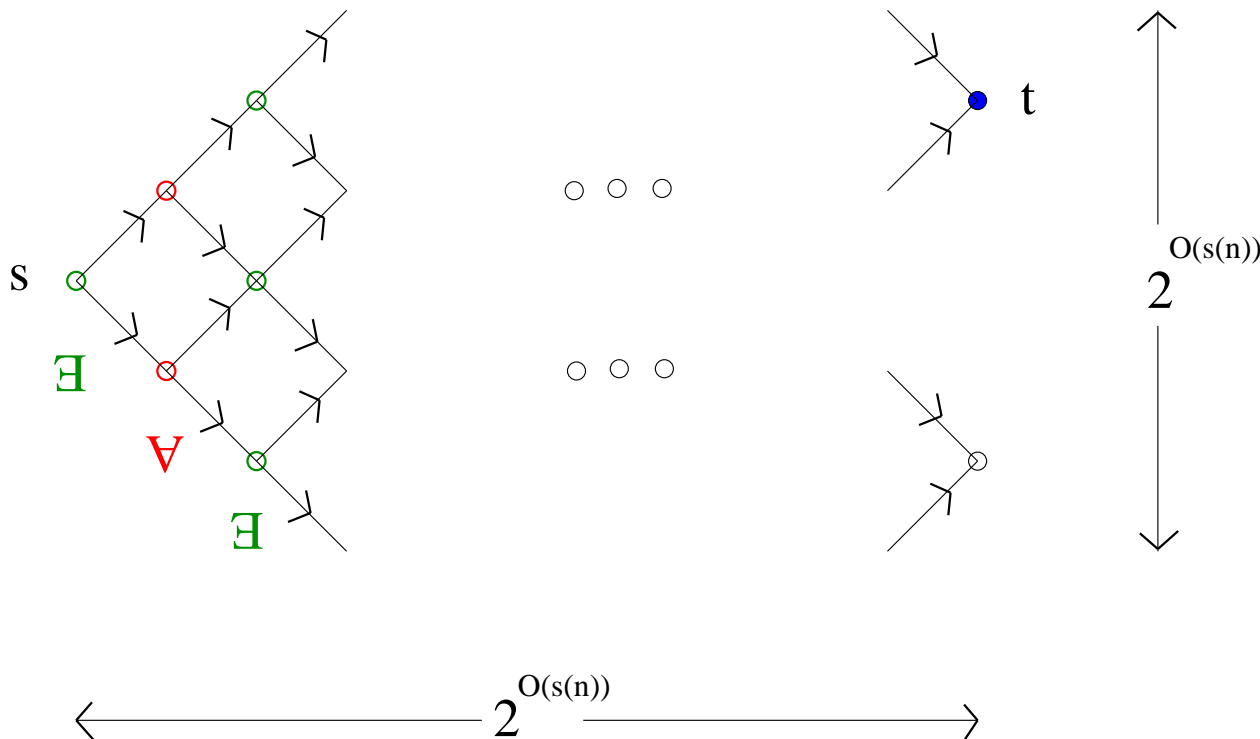
$$\text{Thm: } \mathbf{ASPACE}[s(n)] = \mathbf{DTIME}[2^{O(s(n))}]$$

**Proof:**  $\mathbf{ASPACE}[s(n)] \subseteq \mathbf{DTIME}[2^{O(s(n))}]$ :

Let  $A$  be an  $\mathbf{ASPACE}[s(n)]$  machine,  $w$  an input,  $n = |w|$ .

$\text{CompGraph}(A(w))$  has size  $\leq 2^{O(s(n))}$

Marking algorithm evaluates this in  $\mathbf{DTIME}[2^{O(s(n))}]$ .



# **DTIME** $[2^{O(s(n))}] \subseteq \mathbf{ASPACE}[s(n)]$ :

Let  $M$  be **DTIME** $[2^{k(s(n))}]$  TM,  $w$  an input,  $n = |w|$ .

alternating procedure  $C(t, p, a)$  accepts iff contents of cell  $p$  at time  $t$  in  $M$ 's computation on input  $w$  is symbol  $a$ .

$C(t + 1, p, b)$  holds iff the three symbols  $a_{-1}, a_0, a_1$  in tape positions  $p - 1, p, p + 1$  lead to a "b" in position  $p$  in one step of  $M$ 's computation.

$$C(t + 1, p, b) \equiv \bigvee_{(a_{-1}, a_0, a_1) \xrightarrow{M} b} \bigwedge_{i \in \{-1, 0, 1\}} C(t, p + i, a_i)$$

Space needed is  $O(\log 2^{k(s(n))}) = O(s(n))$ .

Note that  $M$  accepts  $w$  iff  $C(2^{k(s(n))}, 1, \langle q_f, 1 \rangle)$

	Space						
	0	1	$\bar{s}$	$n - 1$	$n$	$\dots$	$2^{ks(n)}$
0	$\langle q_0, w_0 \rangle$	$w_1$	$\dots$	$w_{n-1}$	$\sqcup$	$\dots$	$\sqcup$
1	$w_0$	$\langle q_1, w_1 \rangle$	$\dots$	$w_{n-1}$	$\sqcup$	$\dots$	$\sqcup$
<b>Time</b>	$\vdots$	$\vdots$	$\vdots$			$\vdots$	
$\bar{t}$				$a_{-1}$	$a_0$	$a_1$	
$\bar{t} + 1$				$b$			
	$\vdots$	$\vdots$	$\vdots$			$\vdots$	
$2^{ks(n)}$	$\langle q_f, 1 \rangle$	$\sqcup$	$\dots$	$\sqcup$	$\sqcup$	$\dots$	$\sqcup$

$$C(t + 1, p, b) \equiv \bigvee_{(a_{-1}, a_0, a_1) \xrightarrow{M} b} \bigwedge_{i \in \{-1, 0, 1\}} C(t, p + i, a_i)$$

This completes the proof of the Alternation Thm. □

