P and NP

Lecture 22

Today

Computational Complexity

P, **NP**, **PSPACE**, **EXP**

NP-completeness

Non-deterministic Turing Machines

Resource Bounded Computation

Interested in solving problems using limited time/memory

T-time TM:
On any input of length n, halts within $T(n)$ steps.

Polynomial-Time TM:
T-time TM where T is some polynomial

e.g., $T(n) = 2n + 100$, $T(n) = 5n^2 + 1$, $T(n) = n^{42} + 1$.

S-Space TM:
On any input of length n, uses at most $S(n)$ tape cells.
Polynomial-Space TM: When S is a polynomial

P, PSPACE, EXP

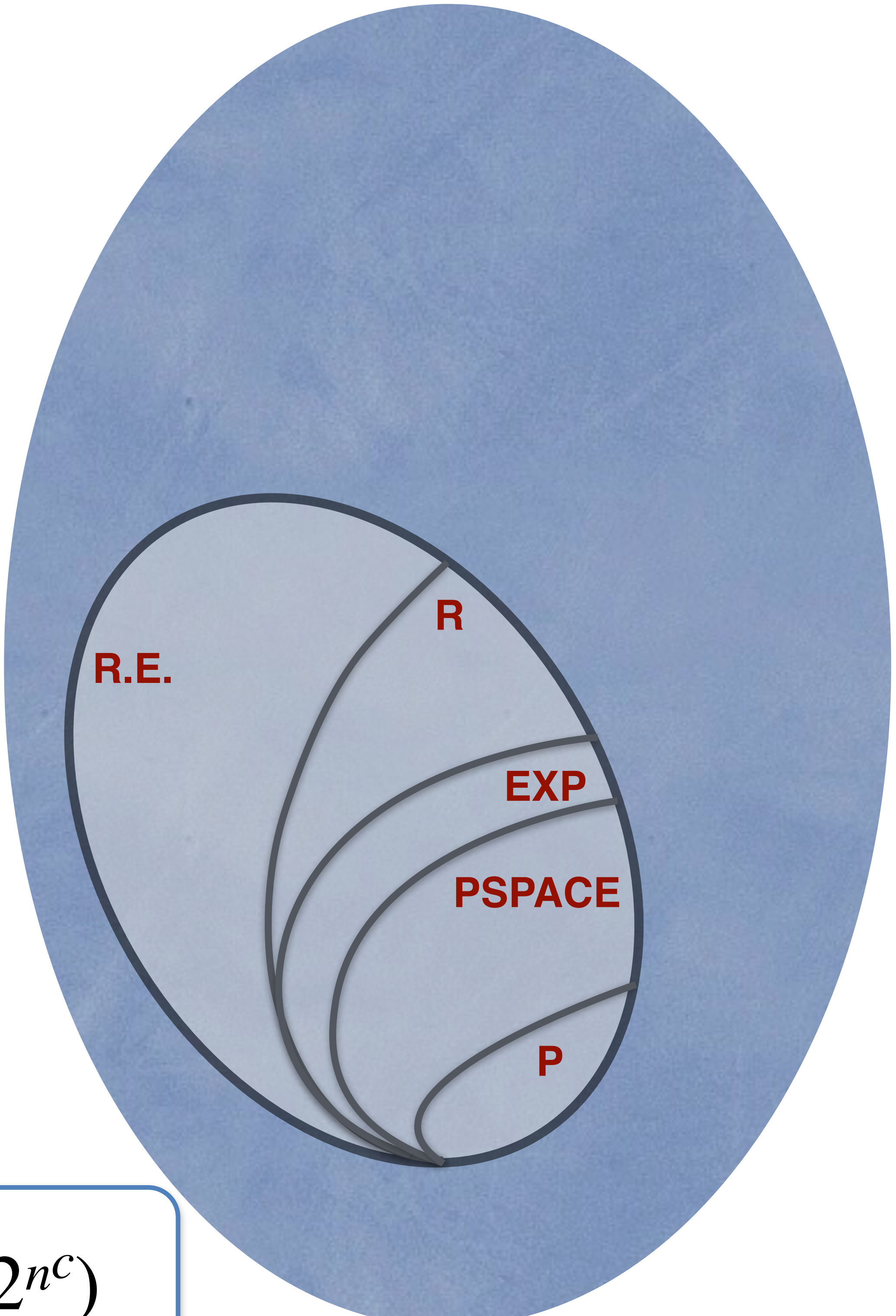
Sub-classes of **R**, the class of all decidable languages

P = class of languages decided by *polynomial-time TMs.*

PSPACE = class of languages decided by *polynomial-space TMs.*

EXP = class of languages decided by *exponential-time* TMs.

$O(2^{n^c})$



P as feasible computation

The most standard proxy for "feasible" computation

Caveat: n^{50} is not feasible, even for small values of n.

Why not model say, n^4 as feasible?

Will be model dependent:
depends on 1-tape TM vs. k-tape TM, TM vs. RAM,
size of the tape alphabet etc.

Typically, polynomial overheads when simulating one model in another. Hence **P** is the same class in all such models.

Typically, for *interesting* problems in **P**, reasonably efficient algorithms have been developed.
(But this is provably impossible for all of **P**.)

NP

An important class of languages

Informally: **NP** is the class of languages with an *efficiently verifiable certificate of membership*

e.g., L_{Sudoku} = Set of all generalized ($n^2 \times n^2$) Sudoku puzzles with a solution

Membership certificate: a solution.
Efficiently verifiable
(Linear time to check that all columns, rows and the $n \times n$ cells satisfy the rules in each solution)

NP

Informally: **NP** is the class of languages with an efficiently verifiable *certificate of membership*

Intuitively, for many problems it is *much* easier to verify a solution than to find one (or to find out that one doesn't exist)

Major Open Question:
Prove that this is the case for even one language!

May not have an easy-to-verify certificate of non-membership

NP

Formally:

$L \in$ **NP** iff $\exists\, V \in$ **P** and a polynomial p s.t.

$$L = \{\, x \mid \exists\, w \in \{0,1\}^{p(|x|)} \text{ s.t. } (x,w) \in V \,\}$$

Note: We insist $|w|$ is polynomial in $|x|$, so that the verification can be done in time polynomial in $|x|$:

Suppose V can be decided by a p' time-bounded TM. Then time to verify the certificate:

$$p'(|(x,w)|) = O(p'(|x|+|w|)) = O(\, p'(|x|+p(|x|))\,) \leq p''(|x|)$$

for some polynomial p''

NP: Examples

L in **NP** : there is V in **P** s.t.

$L = \{ x \mid \exists\, w \text{ (short) s.t. } (x,w) \in V \}$

All the languages in **P**

Suppose $L \in$ **P**

Let $V = \{ (x,\varepsilon) \mid x \in L \}$ so that

$L = \{ x \mid \exists w \in \{0,1\}^0 \text{ s.t. } (x,w) \in V \}$

where $V \in$ **P**

P $\subseteq$ **NP**

NP: Examples

L in **NP** : there is V in **P** s.t.
$L = \{ x \mid \exists\, w \text{ (short) s.t. } (x,w) \in V \}$

Checking if there is a structure

$L_{\text{Hamilton}} = \{ G \mid G$ has a Hamiltonian Cycle$\}$

$V_{\text{Hamilton}} = \{ (G,C) \mid C$ is a Hamiltonian Cycle in $G \}$

$L_{\text{Clique}} = \{ (G,t) \mid G$ has a subgraph isomorphic to $K_t \}$

$V_{\text{Clique}} = \{ (G,t,H) \mid H$ is a subgraph of G isomorphic to $K_t \}$

NP: Examples

L in **NP** : there is V in **P** s.t.
$L = \{ x \mid \exists\, w \text{ (short) s.t. } (x,w) \in V \}$

Checking if there is a sufficiently good solution to an *optimization problem*

$L_{\text{TSP}} = \{ (G,t) \mid G$ is a graph with a TSP tour of cost $\leq t \}$

$V_{\text{TSP}} = \{ (G,t,P) \mid P$ is a TSP tour in G with cost $\leq t \}$

Traveling Sales-person Problem

NP: Examples

L in **NP** : there is V in **P** s.t.
$L = \{ x \mid \exists\, w \text{ (short) s.t. } (x,w) \in V \}$

In an axiomatic system, checking if a mathematical theorem has a proof (with at most t characters)

$L_{\text{Prove}} = \{ (\Pi,t) \mid \Pi$ is a statement with a proof of size $\leq t \}$

$V_{\text{Prove}} = \{ (\Pi,t,P) \mid P$ is a proof of Π with size $\leq t \}$

NP: Examples

L in **NP** : there is V in **P** s.t.
$L = \{ x \mid \exists\, w \text{ (short) s.t. } (x,w) \in V \}$

Breaking a Public-Key Encryption Scheme: Recovering the secret-key from a public-key

$L_{\text{PKE-Keys}} = \{ (PK,w) \mid PK$ is a public-key whose secret-key has w as a prefix $\}$

$V_{\text{PKE-Keys}} = \{ (PK,w,SK) \mid$ secret-key SK yields public-key PK and has prefix w $\}$

If **P = NP**, then?

Suppose any $L \in$ **NP** can be decided in time say, quadratic in the time to decide its certificate language V

Can solve large-scale optimization problems (save large amounts of energy, material and other resources)

Prove many outstanding mathematical theorems (if they have proofs short enough for mathematicians to derive manually)

Make Public-Key Cryptography impossible

We believe **P≠NP**, and that these problems don't have polynomial-time algorithms!

Complexity of **NP**

Best known algorithms for many problems in **NP** take exponential time

How hard can problems in **NP** be?
Do they all have at least exponential time algorithms?

Yes!

To check if $x \in L$, can try every possible value of w and check if $(x,w) \in V$

NP ⊆ PSPACE

For any $L \in$ **NP**, a polynomial-space TM M_L.

Run through every possible value of $w \in \{0,1\}^{p(|x|)}$ and call a polynomial-time subroutine M_V to check if $(x,w) \in V$.

Suppose M_V is a p'-time TM. Total space?

M_V is a p'-space TM too.

M_L is a p''-space TM, where
$p''(n) = O(\, p(n) + p'(n+p(n))\,)$

P ⊆ NP ⊆ PSPACE ⊆ EXP

Claim: **PSPACE** ⊆ **EXP**

For $L \in$ **PSPACE**, suppose
a p-space TM M_L with d states and $|\Gamma| = k$

Number of distinct IDs on an input of size n?

$$d \times p(n) \times k^{p(n)} \;\leq\; 2^{p'(n)}$$

If M_L doesn't halt within that many steps,
it must have repeated some ID $\Rightarrow$ in an infinite loop!

An exponential-time TM for L: Simulate M_L for $2^{p'(n)}$ steps.
If M_L has not halted already, halt and reject.

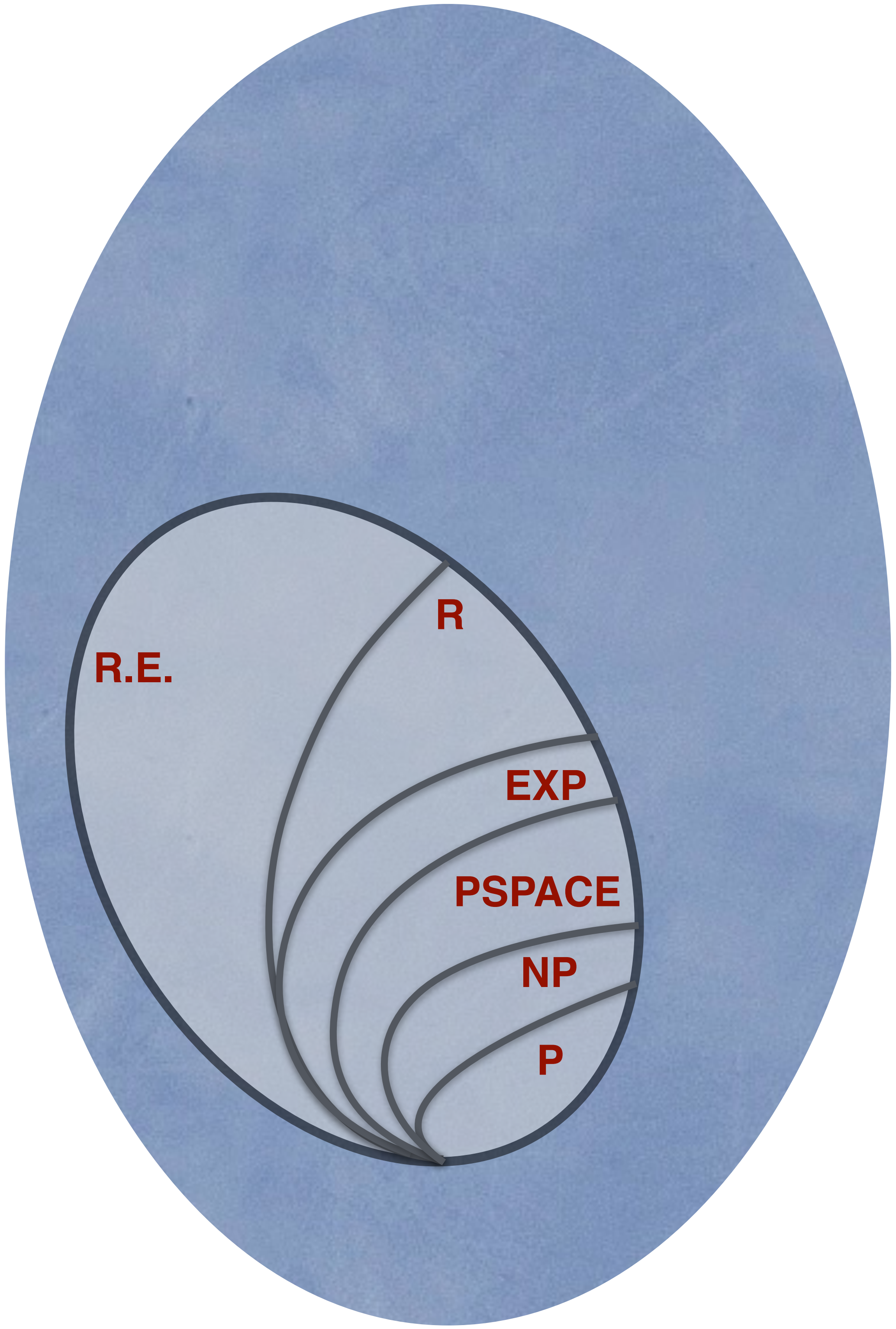
$P \subseteq NP \subseteq PSPACE \subseteq EXP$

It is known that **P ≠ EXP**

(Time-Hierarchy Theorem)

Hence, at least one containment in the chain **P** ⊆ **NP** ⊆ **PSPACE** ⊆ **EXP** is strict.

All 3 widely believed to be strict



Polynomial-Time Reduction

Suppose f is a reduction from L_1 to L_2

We say f is a *polynomial-time reduction* if f can be computed by a polynomial-time TM

In that case we write $L_1 \leq_{poly} L_2$

Positive Implication: If $L_1 \leq_{poly} L_2$ and $L_2 \in$ **P** then $L_1 \in$ **P**

Note: $|f(x)| \leq p(|x|)$ for a polynomial p

NP-Completeness

Consider the language

$ACCEPT_{NP} = \{ (z, x, m, 1^t) \mid \exists w \in \{0,1\}^m$ s.t. M_z accepts (x,w) within t steps $\}$

$ACCEPT_{NP} \in$ **NP**

$\forall L \in$ **NP**, $L \leq_{poly} ACCEPT_{NP}$

NP-Completeness

Claim: $ACCEPT_{NP} \in$ **NP**

$$V_{\text{Accept}} = \{ (z, x, m, 1^t, w) \mid w \in \{0,1\}^m \text{ and } M_z \text{ accepts } (x,w) \text{ within } t \text{ steps} \}$$

Claim: $\forall\, L \in$ **NP**, $L \leq_{poly} ACCEPT_{NP}$

Let $V \in \boldsymbol{P}$ and polynomial p be s.t.

$$L = \{ x \mid \exists\, w \in \{0,1\}^{p(|x|)} \text{ s.t. } (x,w) \in V \}$$

Polynomial-time reduction: $f(x) = (z, x, m, 1^t)$

where z s.t. M_z is a p'-time TM for V, $m{=}p(|x|)$, $t{=}p'(|(x,1^m)|)$

NP-Completeness

Consider the language

$$ACCEPT_{NP} = \{ (z, x, m, 1^t) \mid \exists w \in \{0,1\}^m \text{ s.t. } M_z \text{ accepts } (x,w) \text{ within } t \text{ steps } \}$$

$$ACCEPT_{NP} \in \mathbf{NP}$$

$$\forall\ L \in \mathbf{NP},\ L \leq_{poly} ACCEPT_{NP}$$

Implication: $ACCEPT_{NP} \in \mathbf{P} \Leftrightarrow \mathbf{NP} = \mathbf{P}$

$L \leq_{poly} L'$ and $L' \in \mathbf{P}$
$\Rightarrow L \in \mathbf{P}$

NP-Completeness

A language A is said to be **NP**-complete if

$A \in$ **NP**

$\forall L \in$ **NP**, $L \leq_{poly} A$

Any NP-complete language is one of the hardest **NP** languages: if it has a $T(n)$-time algorithm, no **NP** language needs more than $p(n) + T(p(n))$ time for some polynomial p (that depends on the language)

If any **NP**-complete language is in **P**, then **P** = **NP**

NP-Completeness

$ACCEPT_{NP}$ is an **NP**-complete language

Next time: Several *natural* problems are **NP**-complete languages

More than 50 years of effort into finding efficient algorithms for many of these problems

Now widely believed that *such algorithms do not exist*

Non-Deterministic TM

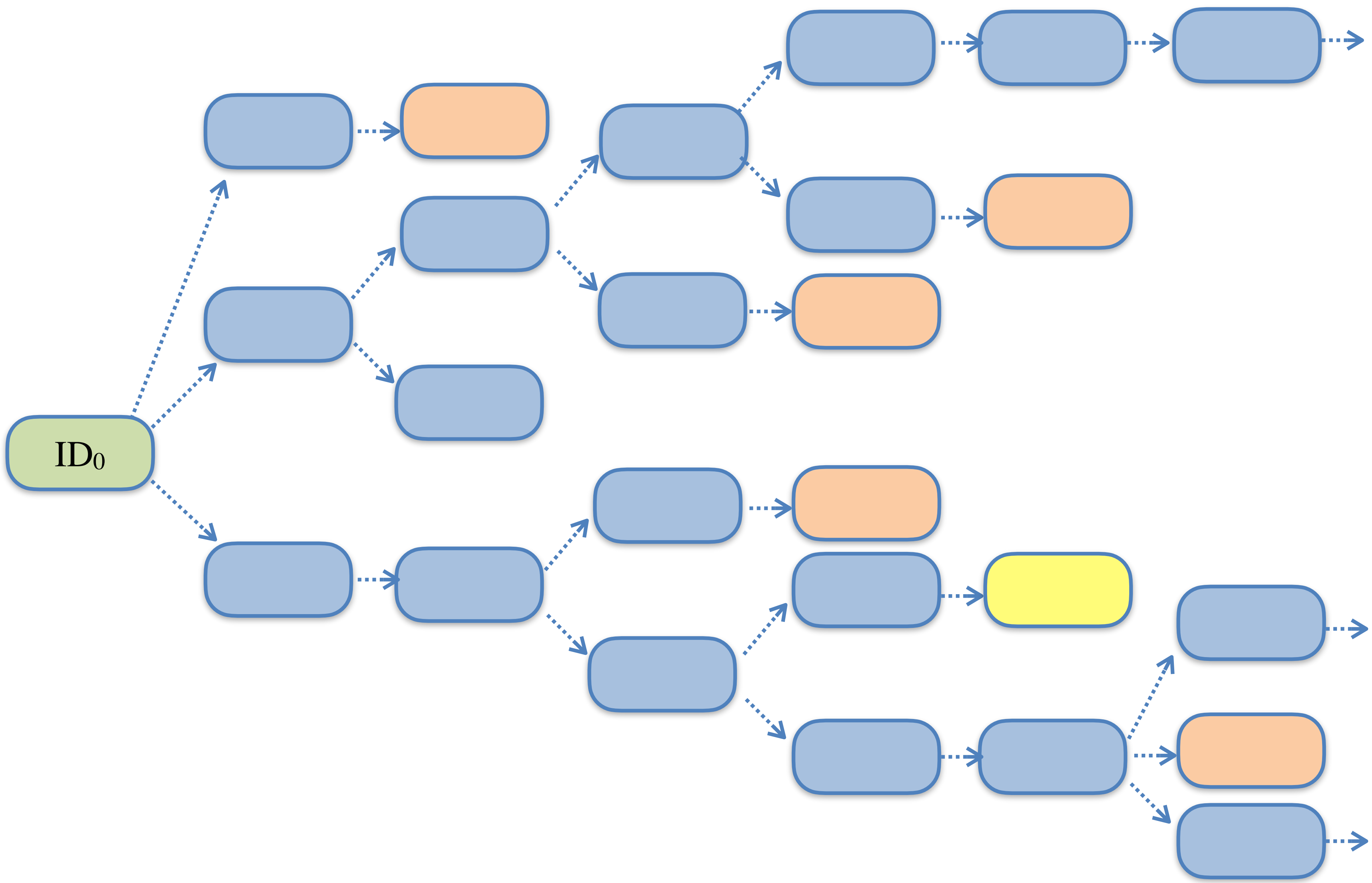
Recall that in a TM the finite control is implemented as (essentially) a DFA

Non-Deterministic TM (NTM): Allow the finite control to be an NFA

$$\delta : Q \times \Gamma \to \mathcal{P}(\, Q \times \Gamma \times \{\, \mathrm{L}, \mathrm{R} \,\}\,)$$

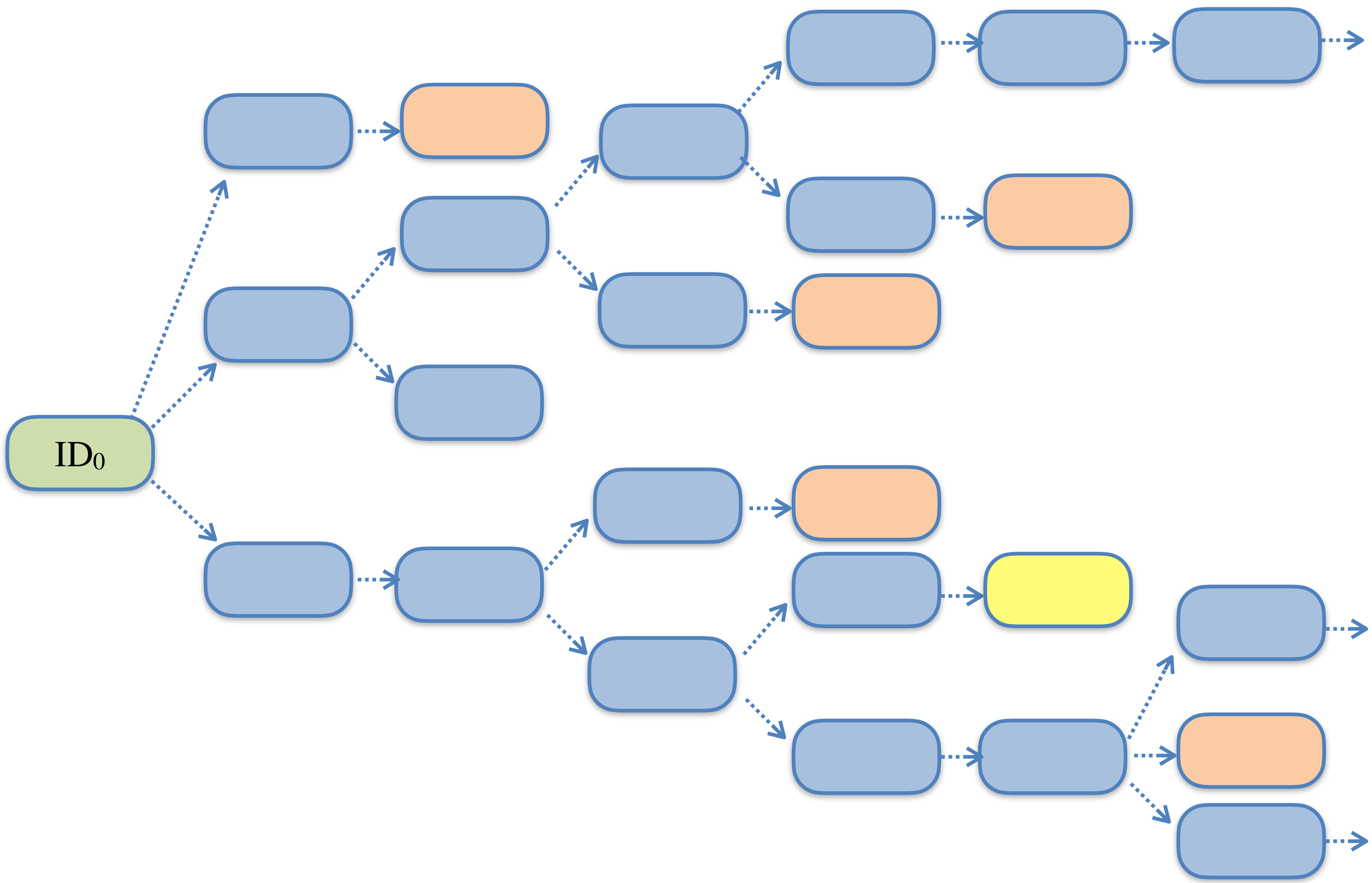
From an ID the TM can move to 0 or more IDs by following each possible transition in the set returned by δ

Non-Deterministic TM



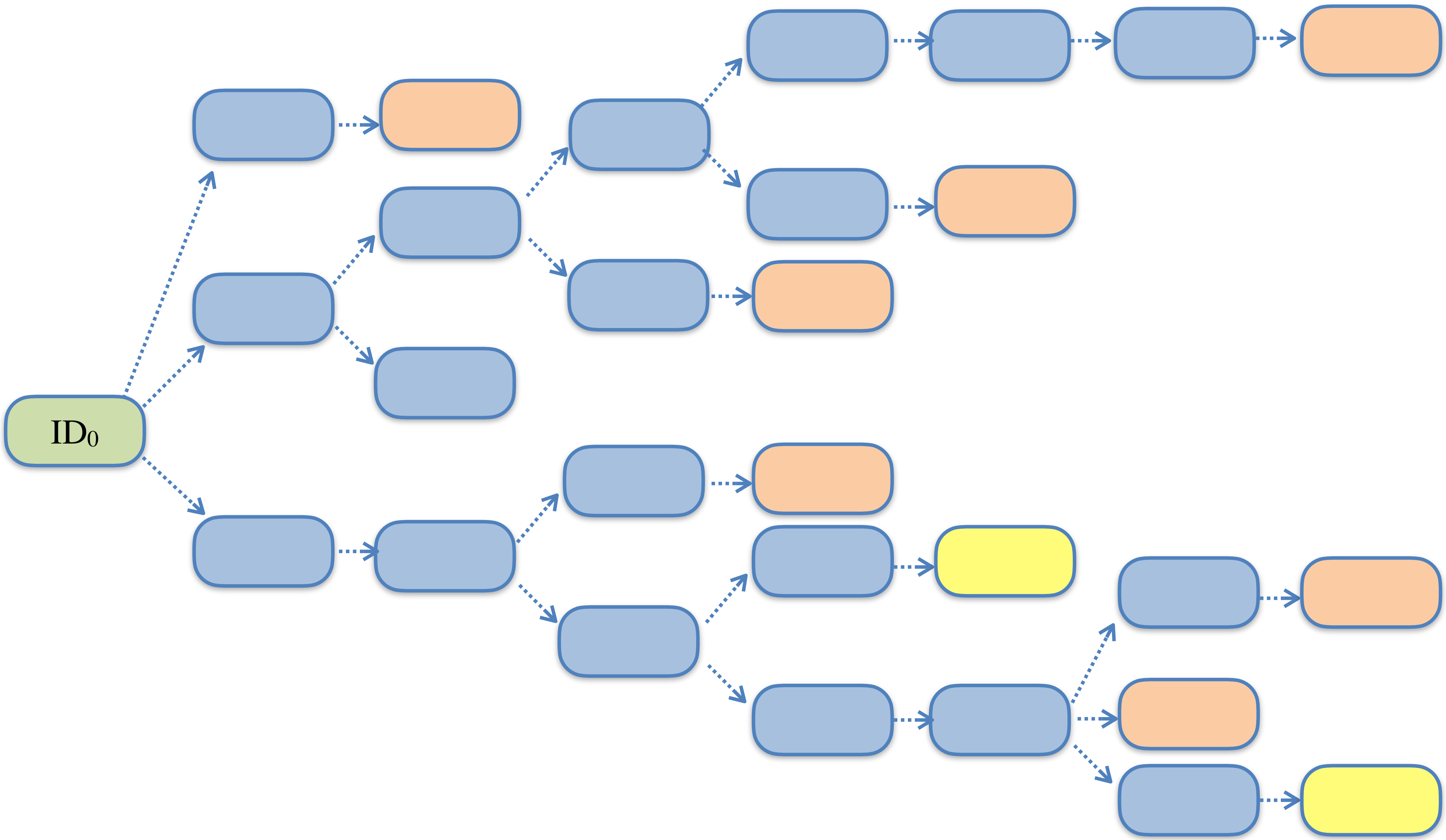
As in the case of NFAs, we say an NTM accepts a string if there exists some execution path starting from the initial ID that accepts (even if some others reject)

Non-Deterministic TM



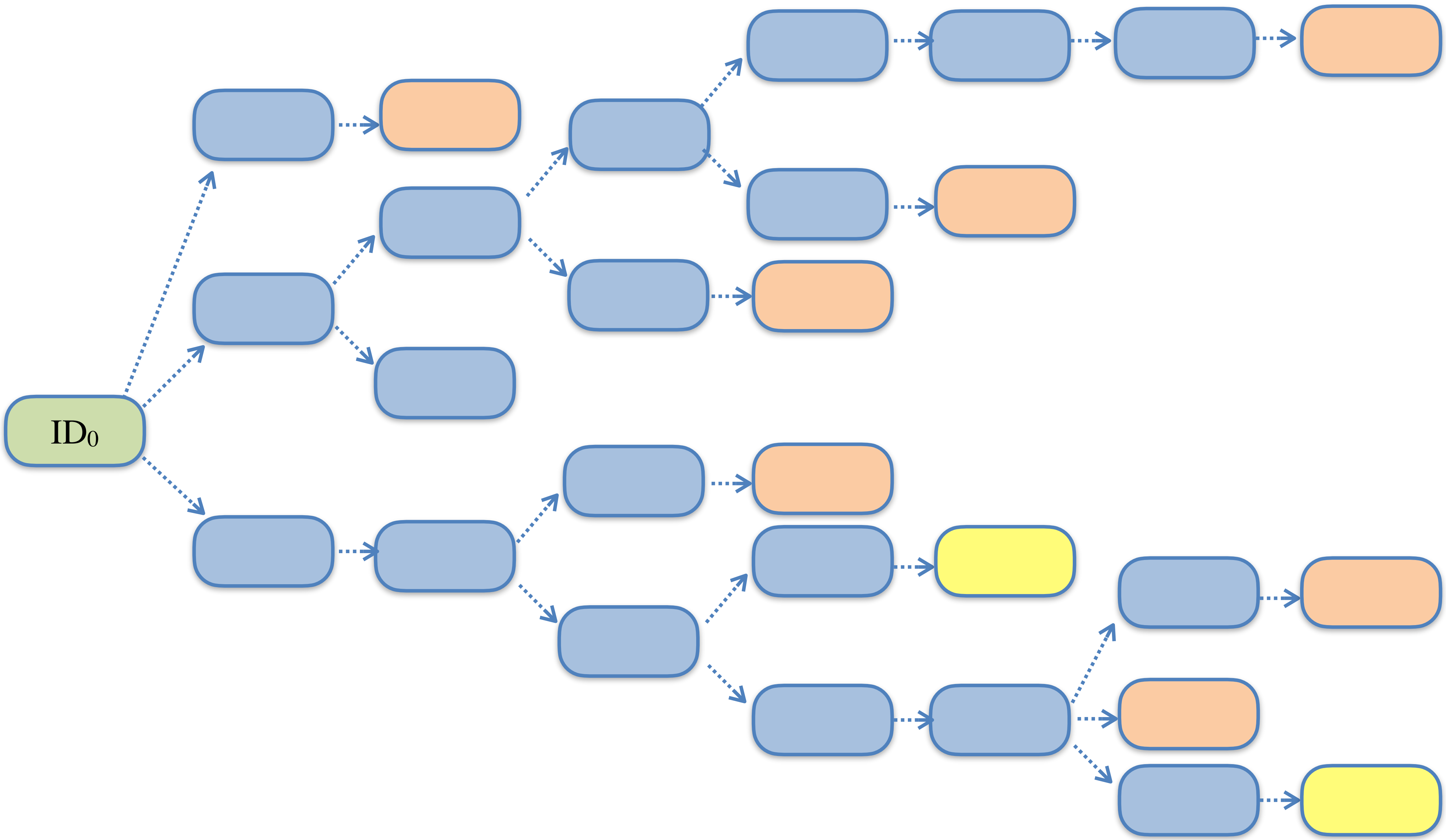
A normal (deterministic) TM can simulate an NTM execution by doing a breadth-first search on the above (implicit) graph

Polynomial-Time NTM



There is a polynomial p s.t., on any input x, every execution thread should finish within $p(|x|)$ steps

Polynomial-Time NTM



Any path in the execution tree can be specified by the sequence of non-deterministic choices: a k-ary string of length $p(n)$ (=depth), where k is max $|\delta(q,a)|$

NP and NTM

$L \in \mathbf{NP} \Leftrightarrow \exists$ a polynomial-time NTM M s.t. $L(M)=L$

$\Rightarrow$: Suppose L has certificate language $V \in \mathbf{P}$.
NTM M behaves as follows:

- write down a "certificate" w of the appropriate length, writing 0 or 1 non-deterministically at each step.
- deterministically check if $(x,w) \in V$, and accept if so.

M accepts x iff $\exists$ w (of the correct length) s.t. $(x,w) \in V$.

$\Leftarrow$: Define V s.t. $(x,w) \in V$ iff when M is run with start ID for input x, using w as the string of non-deterministic choices, it accepts.