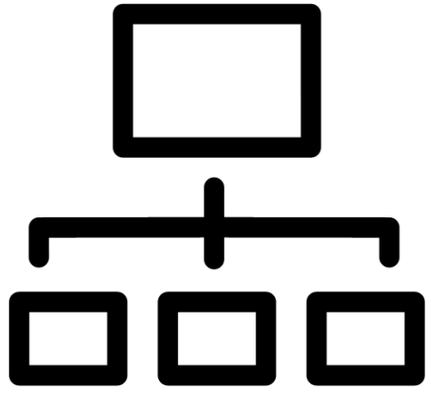
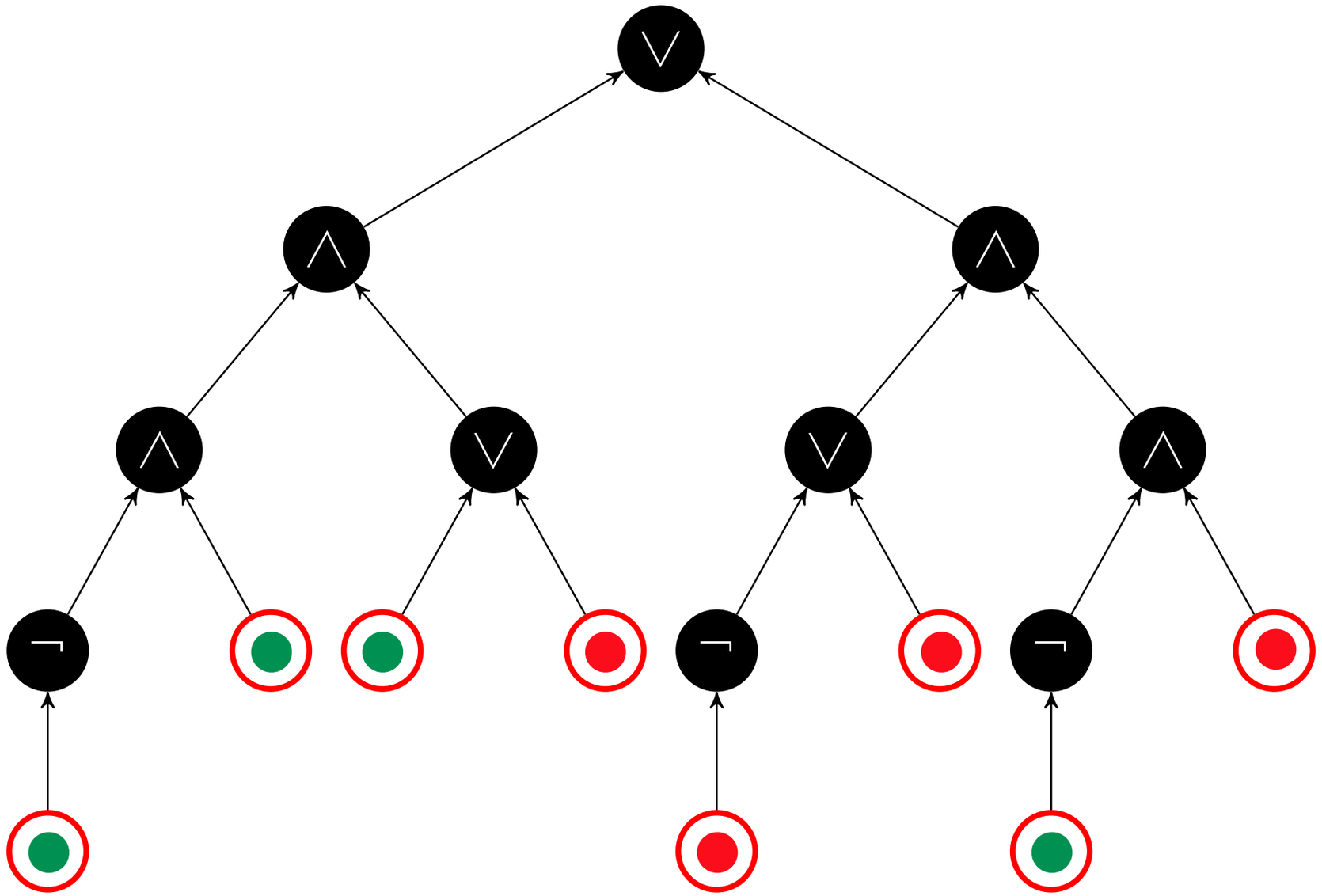
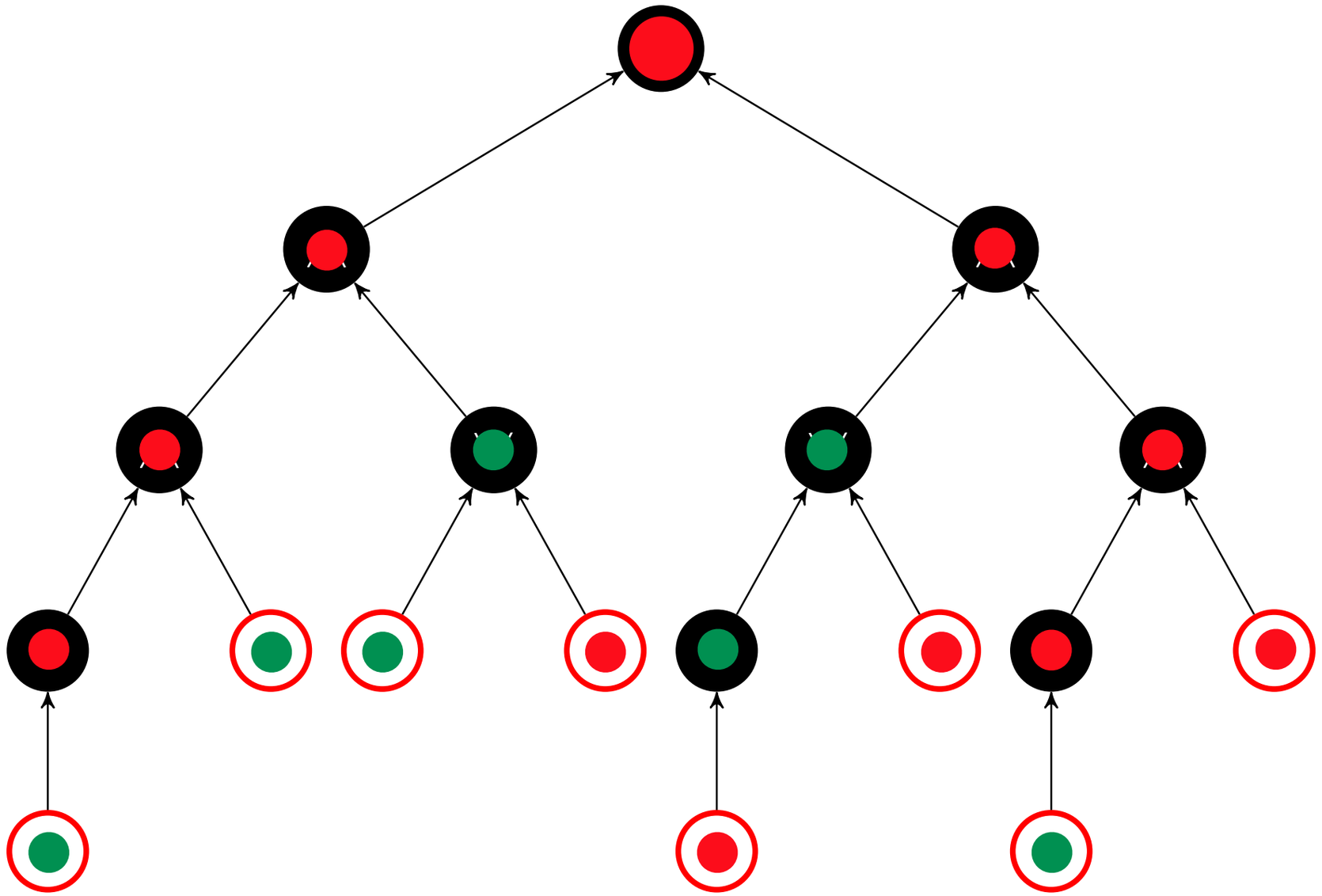
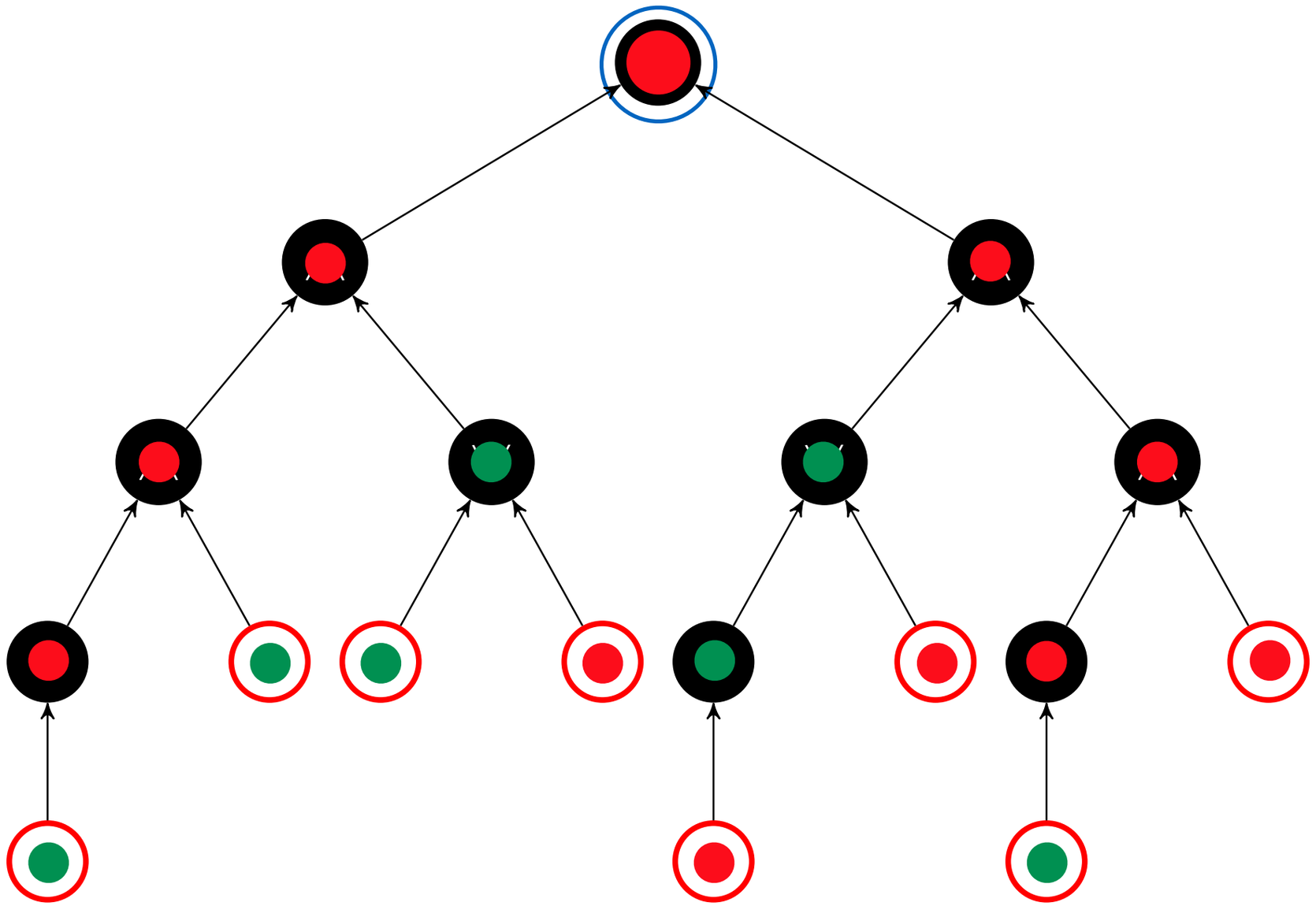
Boolean Circuits

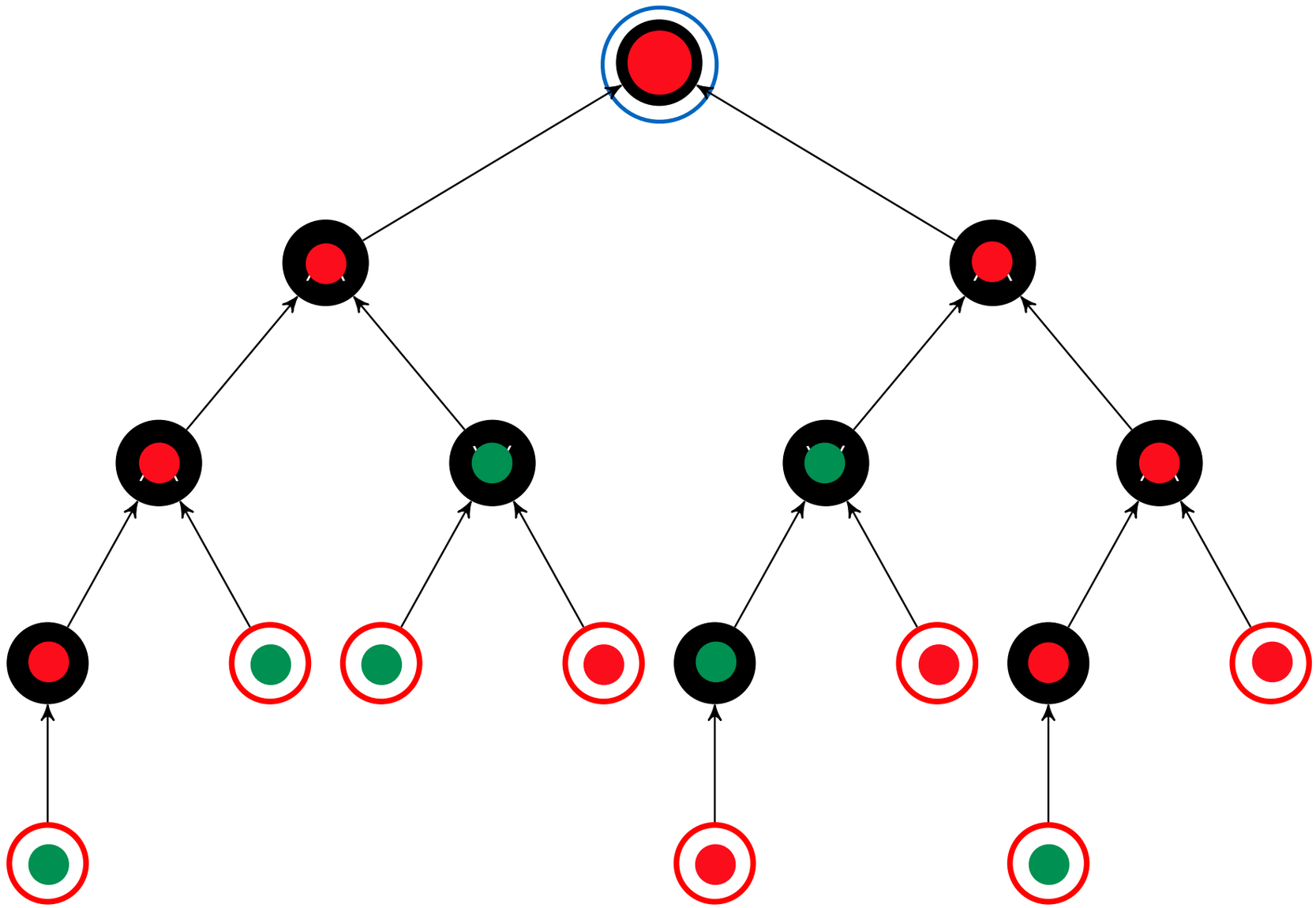




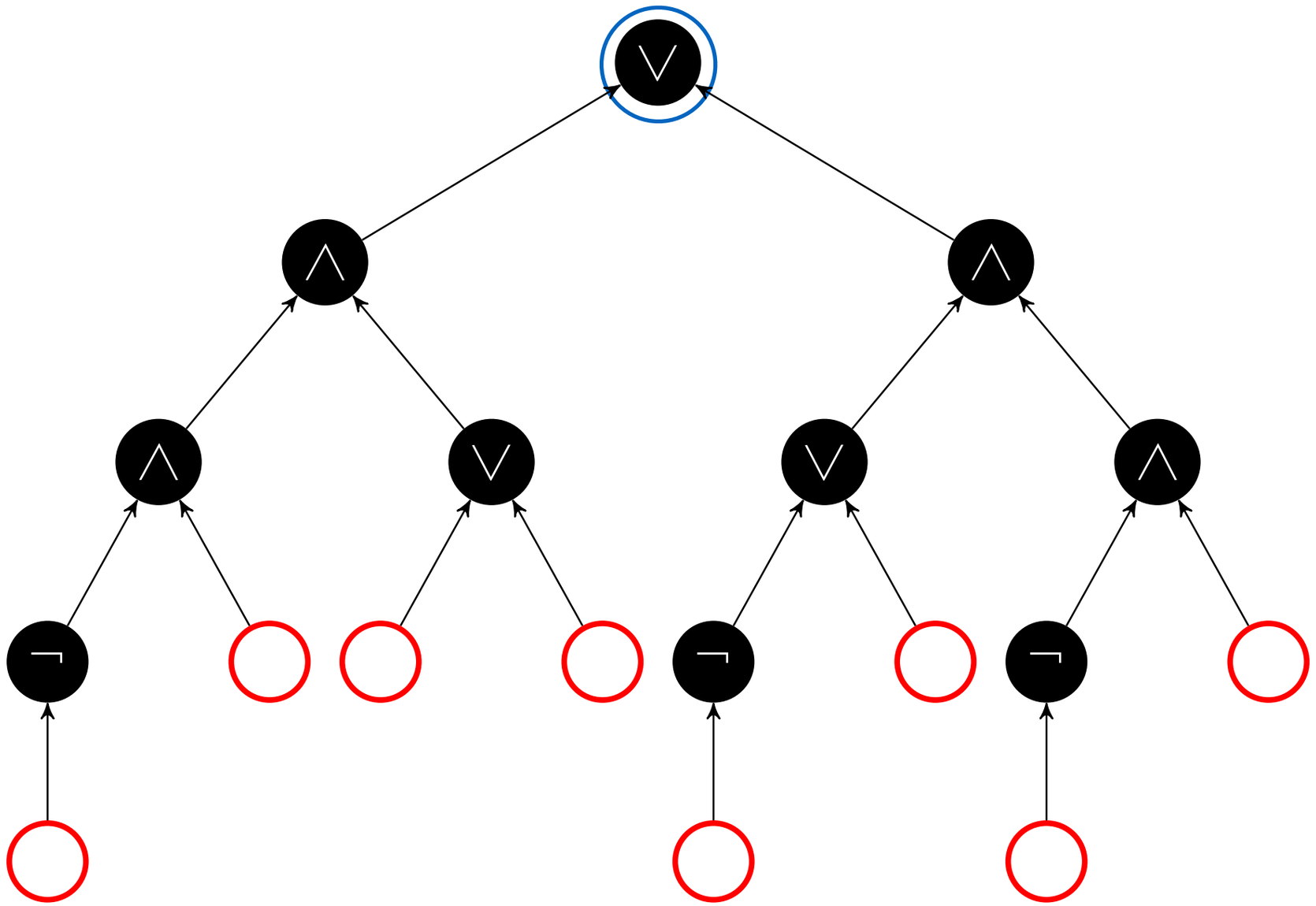




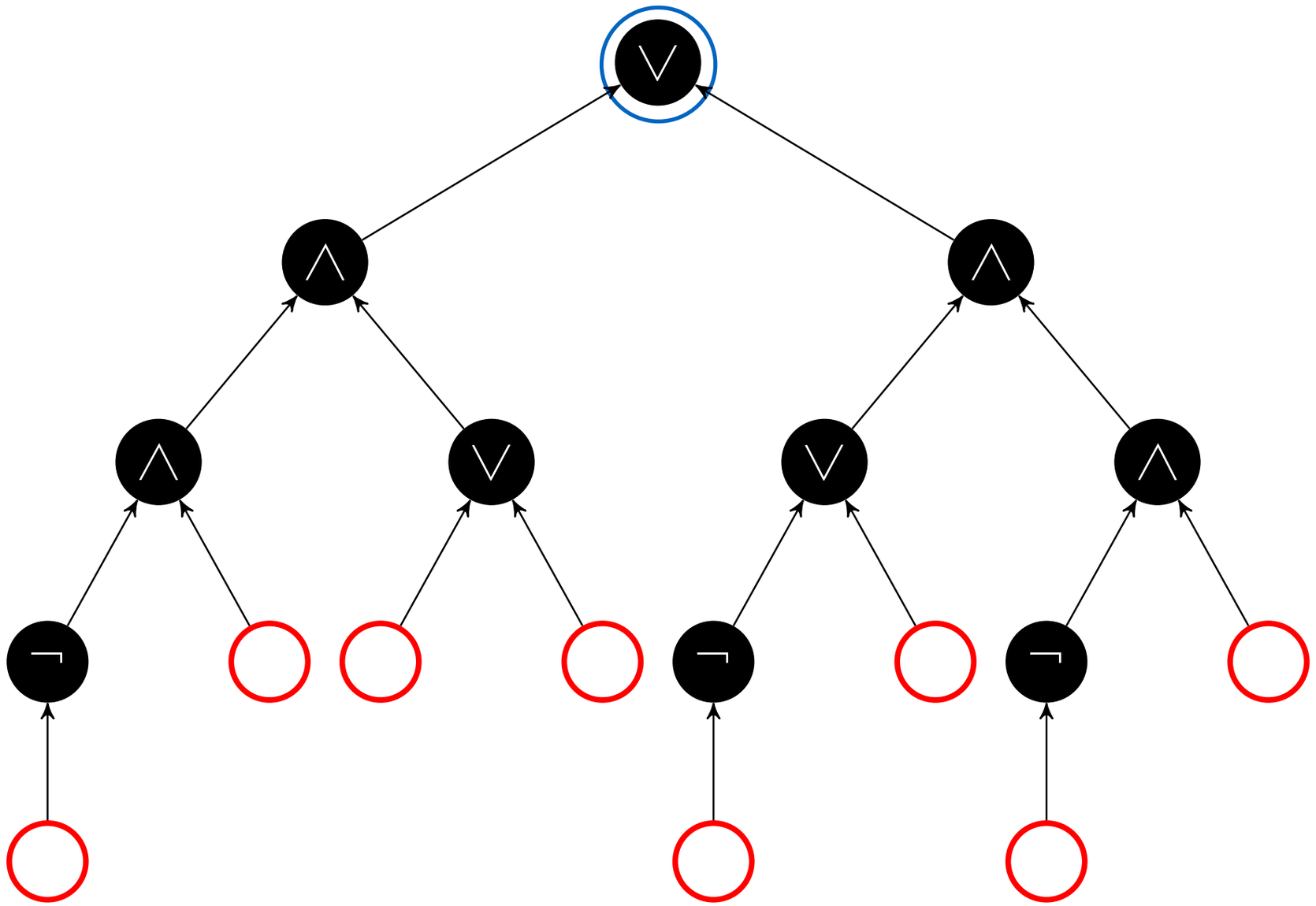
Output Node
(outdegree zero)



Output Node
(outdegree zero)



Output Node
(outdegree zero)



Input Nodes
(indegree zero)

A **n-input, single-output Boolean circuit** is a DAG with **n sources and one sink.**

All remaining vertices are called **gates** and are labeled with the logical operators AND, OR, NOT.

AND/OR gates have fan-in two and NOT gates have fan-in one.

(inputs) (output)

A **n-input, single-output Boolean circuit** is a DAG with **n sources and one sink.**

All remaining vertices are called **gates** and are labeled with the logical operators AND, OR, NOT.

AND/OR gates have fan-in two and NOT gates have fan-in one.

(inputs) (output)

A **n-input, single-output Boolean circuit** is a DAG with **n sources and one sink.**

All remaining vertices are called **gates** and are labeled with the logical operators AND, OR, NOT.

AND/OR gates have fan-in two and NOT gates have fan-in one.

Size of the circuit C, denoted by |C| = number of nodes in C

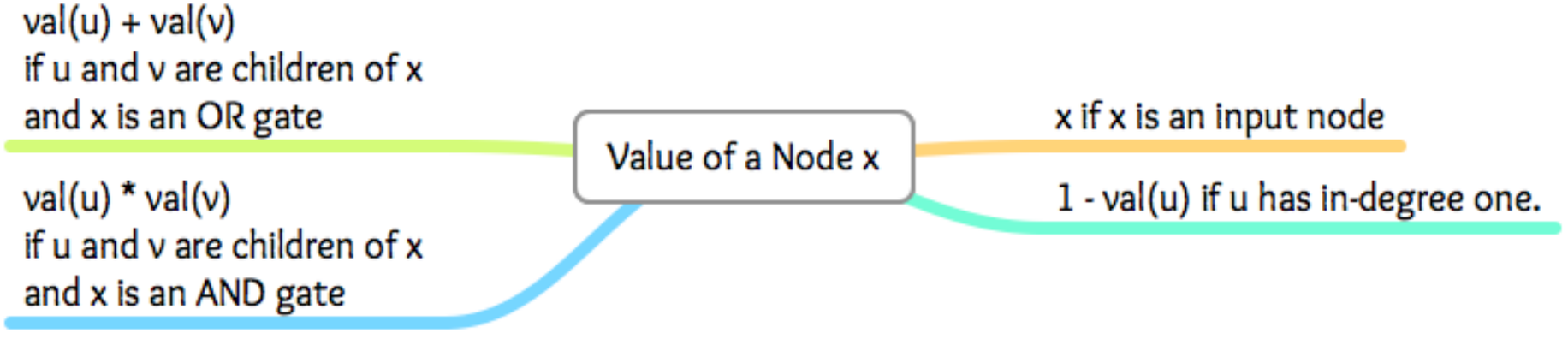
(inputs) (output)

A **n-input, single-output Boolean circuit** is a DAG with **n sources and one sink.**

All remaining vertices are called **gates** and are labeled with the logical operators AND, OR, NOT.

AND/OR gates have fan-in two and NOT gates have fan-in one.

Size of the circuit C, denoted by |C| = number of nodes in C



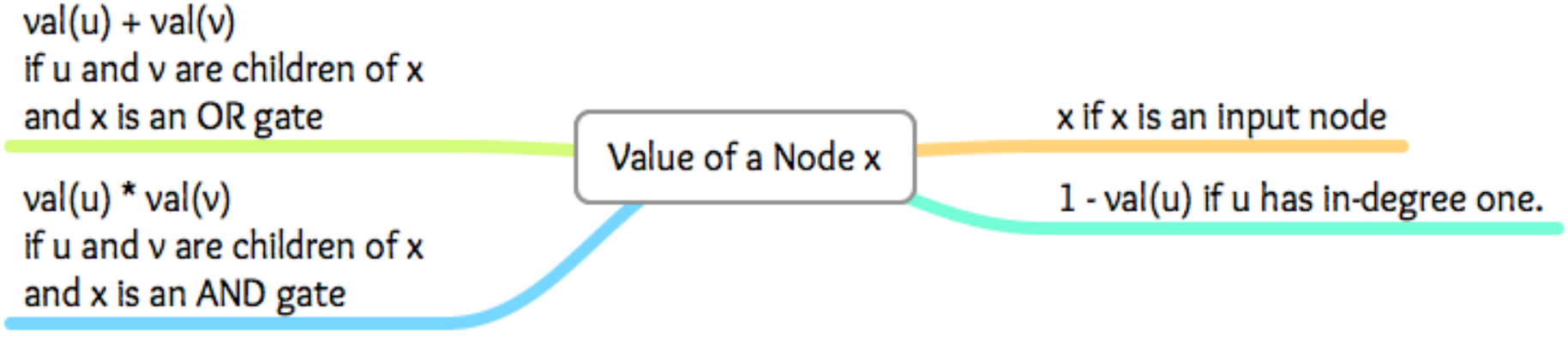
(inputs) (output)

A **n-input, single-output Boolean circuit** is a DAG with **n sources and one sink.**

All remaining vertices are called **gates** and are labeled with the logical operators AND, OR, NOT.

AND/OR gates have fan-in two and NOT gates have fan-in one.

Size of the circuit C, denoted by |C| = number of nodes in C



Value of the circuit C = value[**output node**]

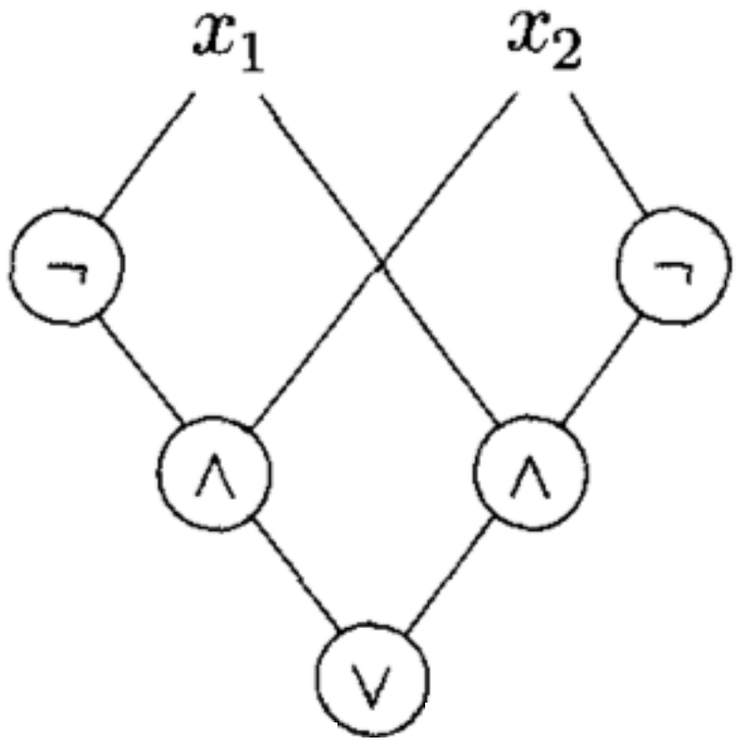
Parity

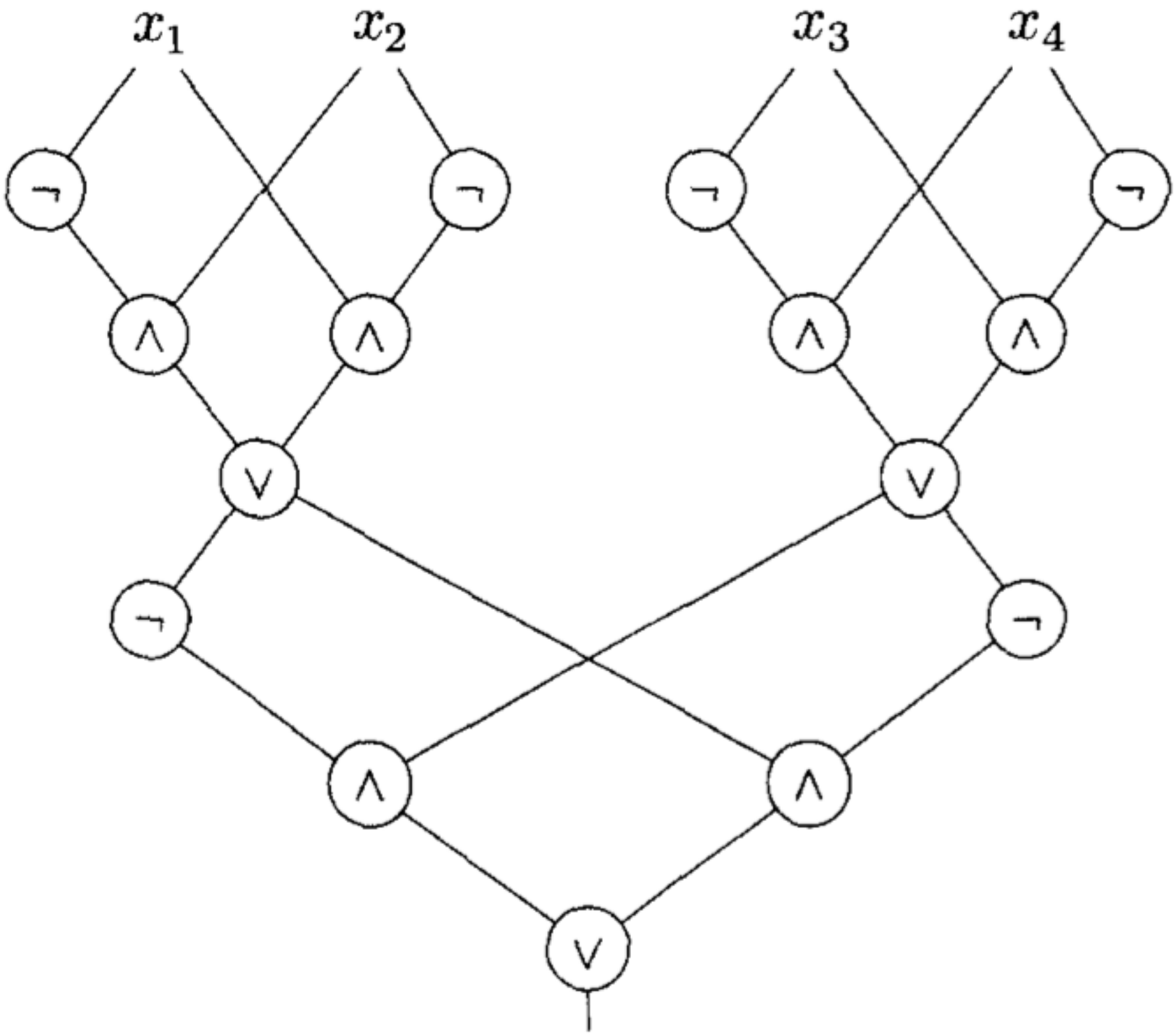
Parity

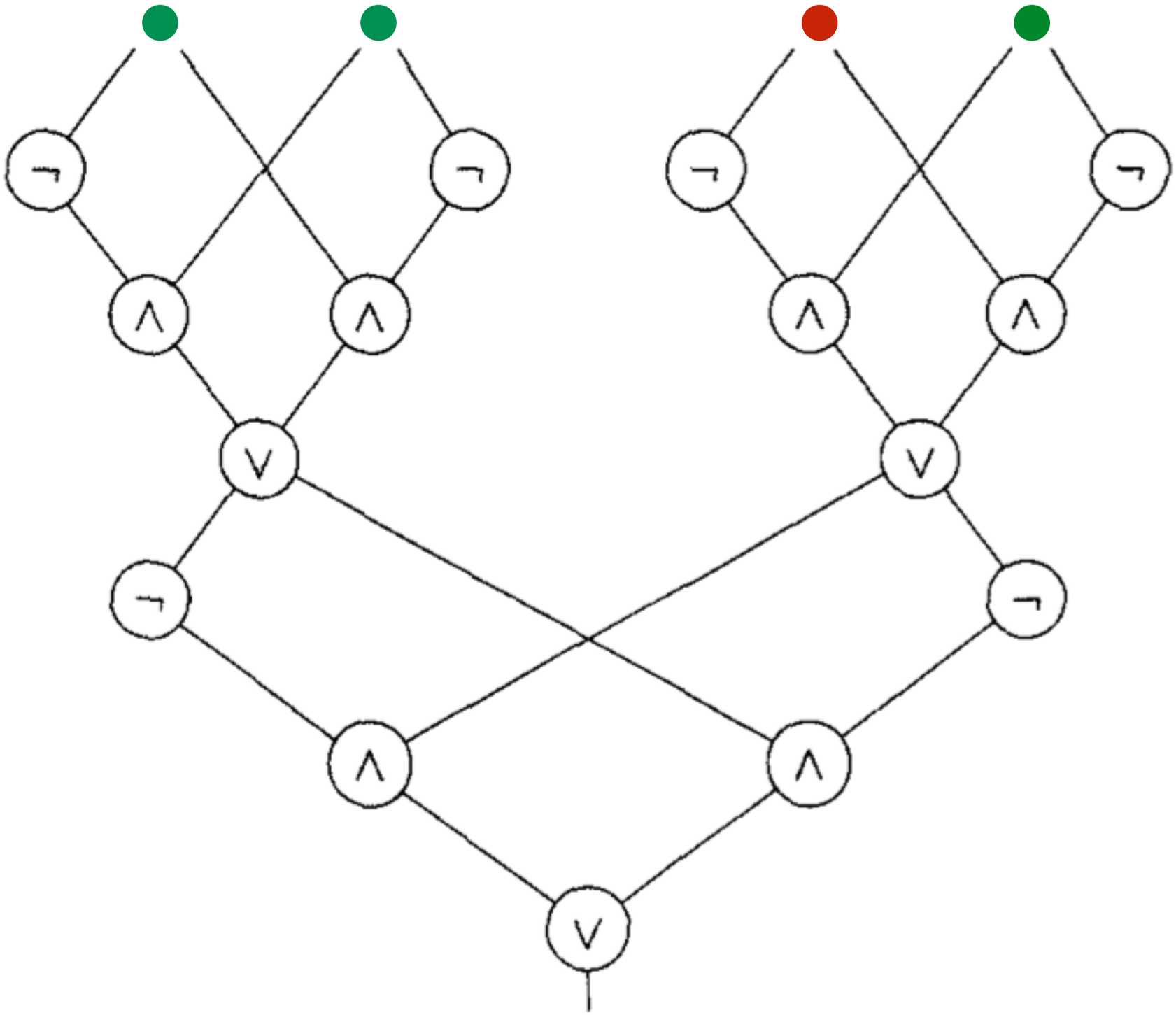
Output **ONE** if, and only if, there are an **odd** number of inputs that are **1.**

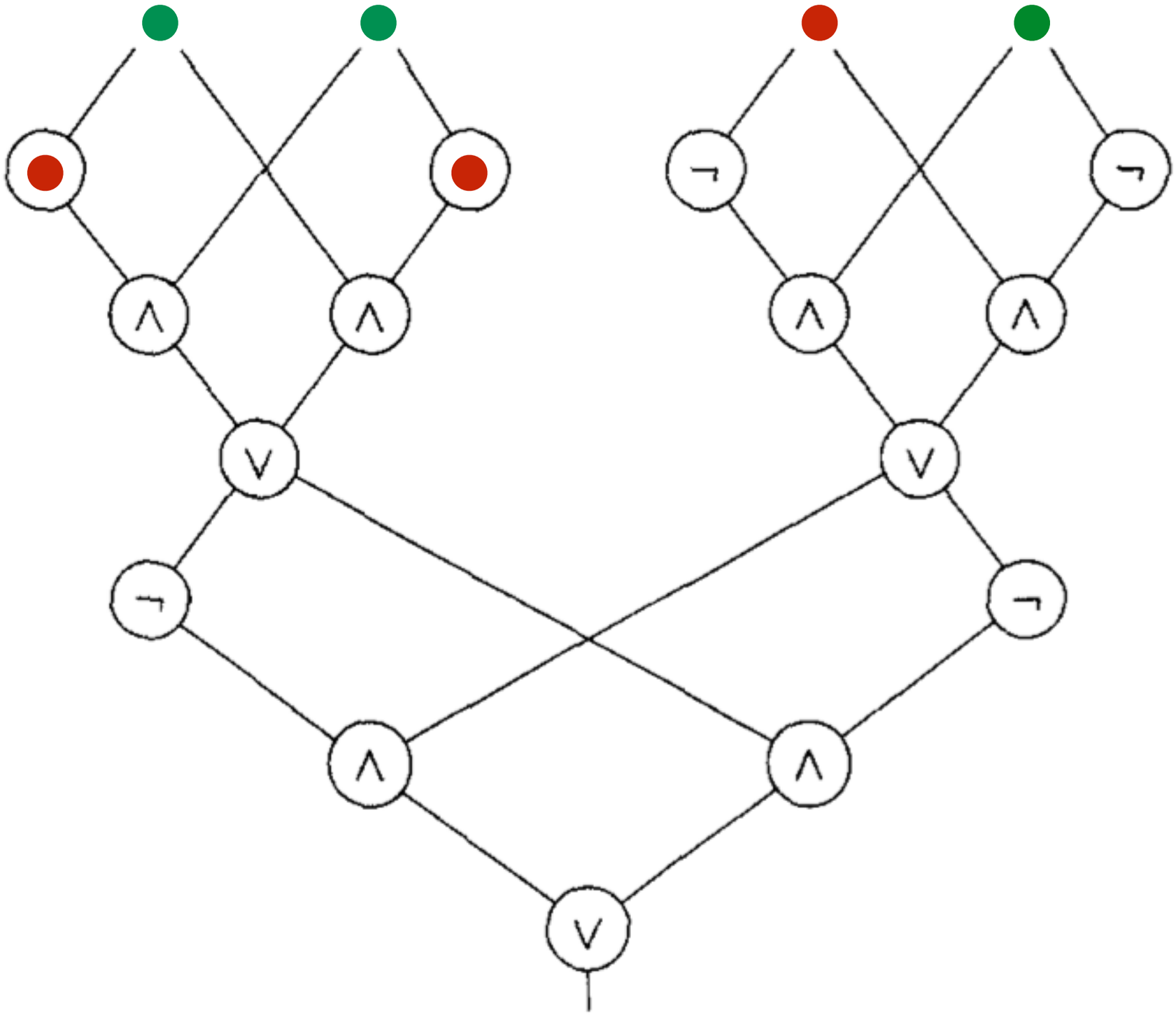
Parity

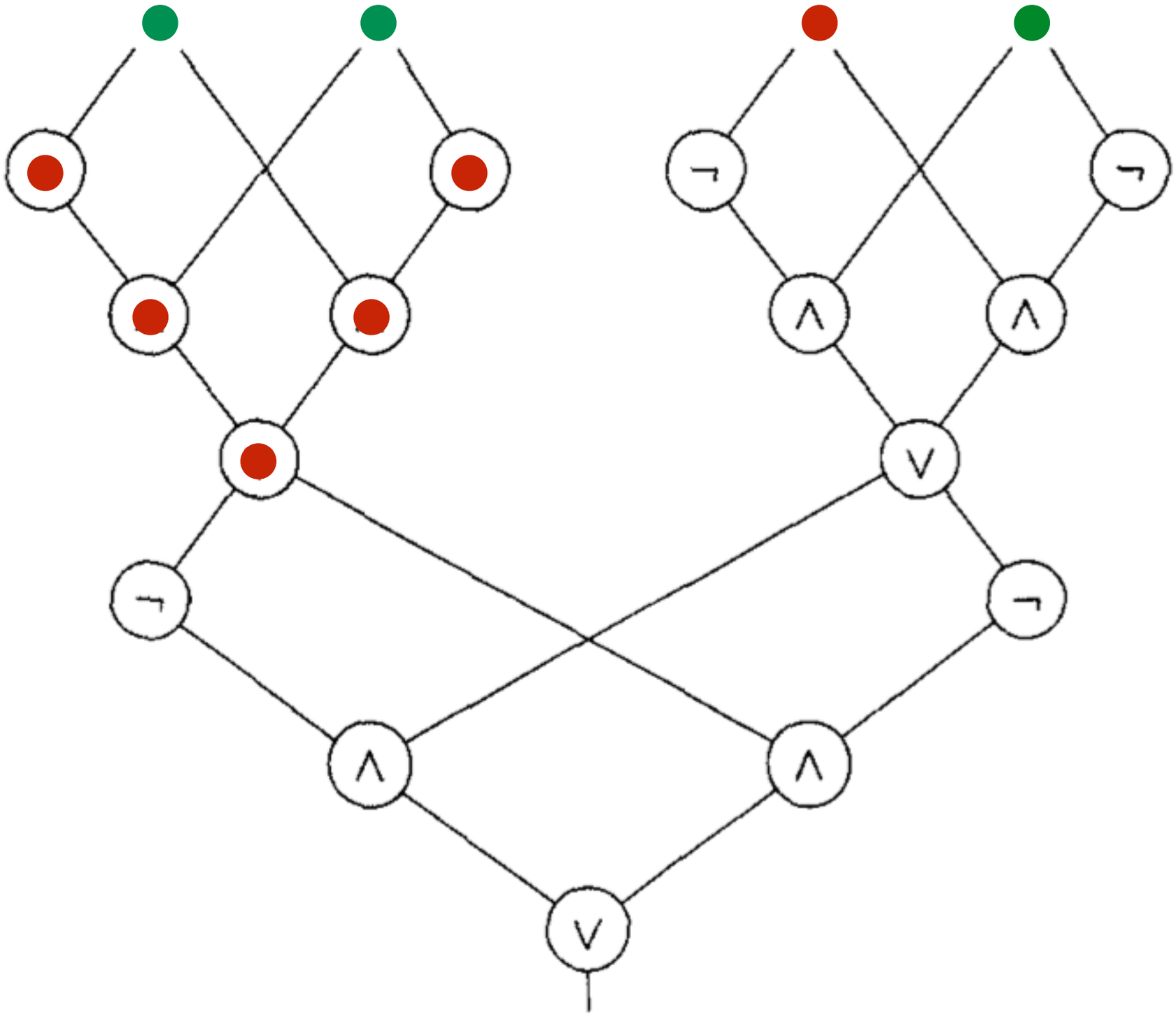
Output **ONE** if, and only if, there are an **odd** number of inputs that are **1.**

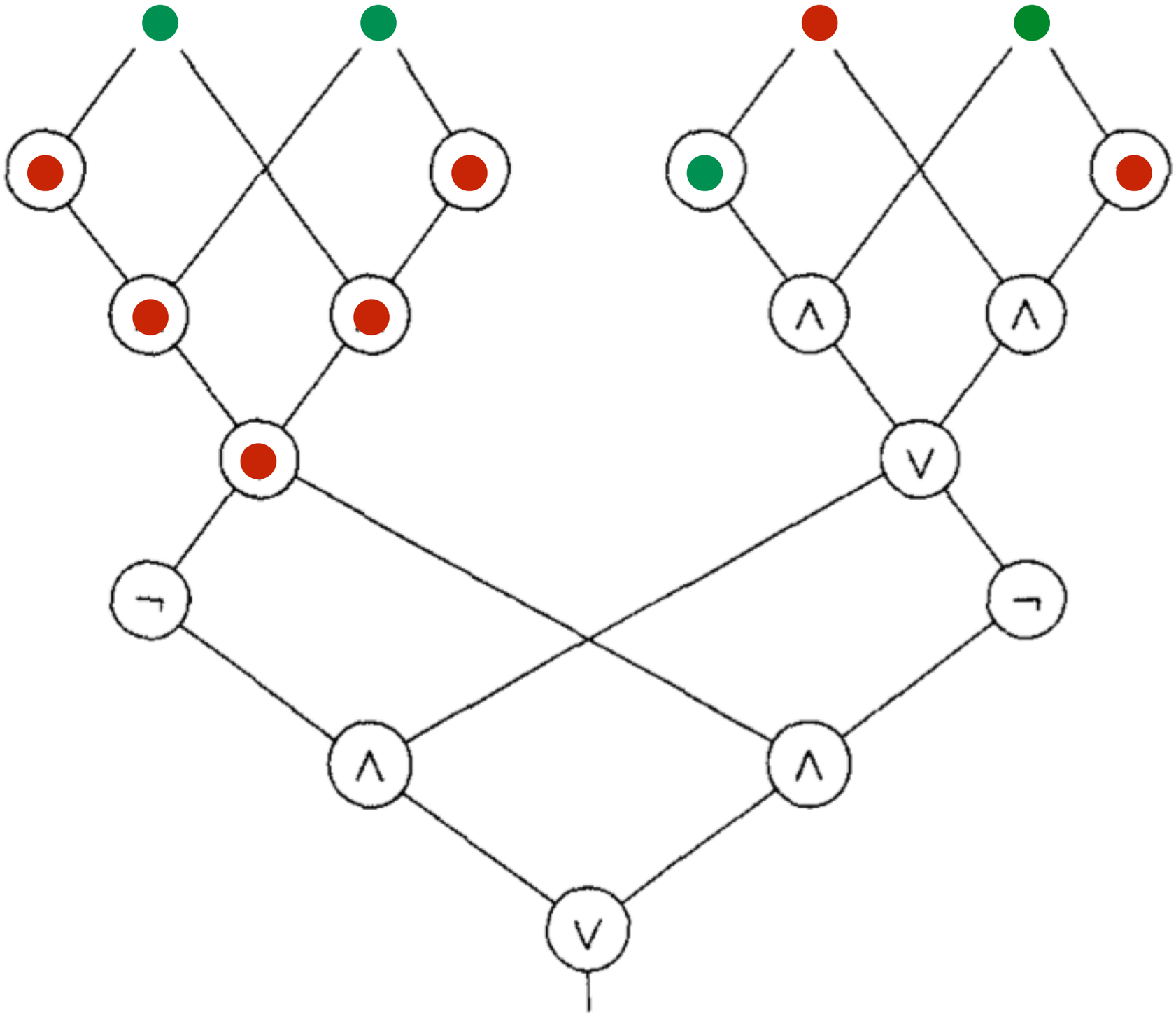


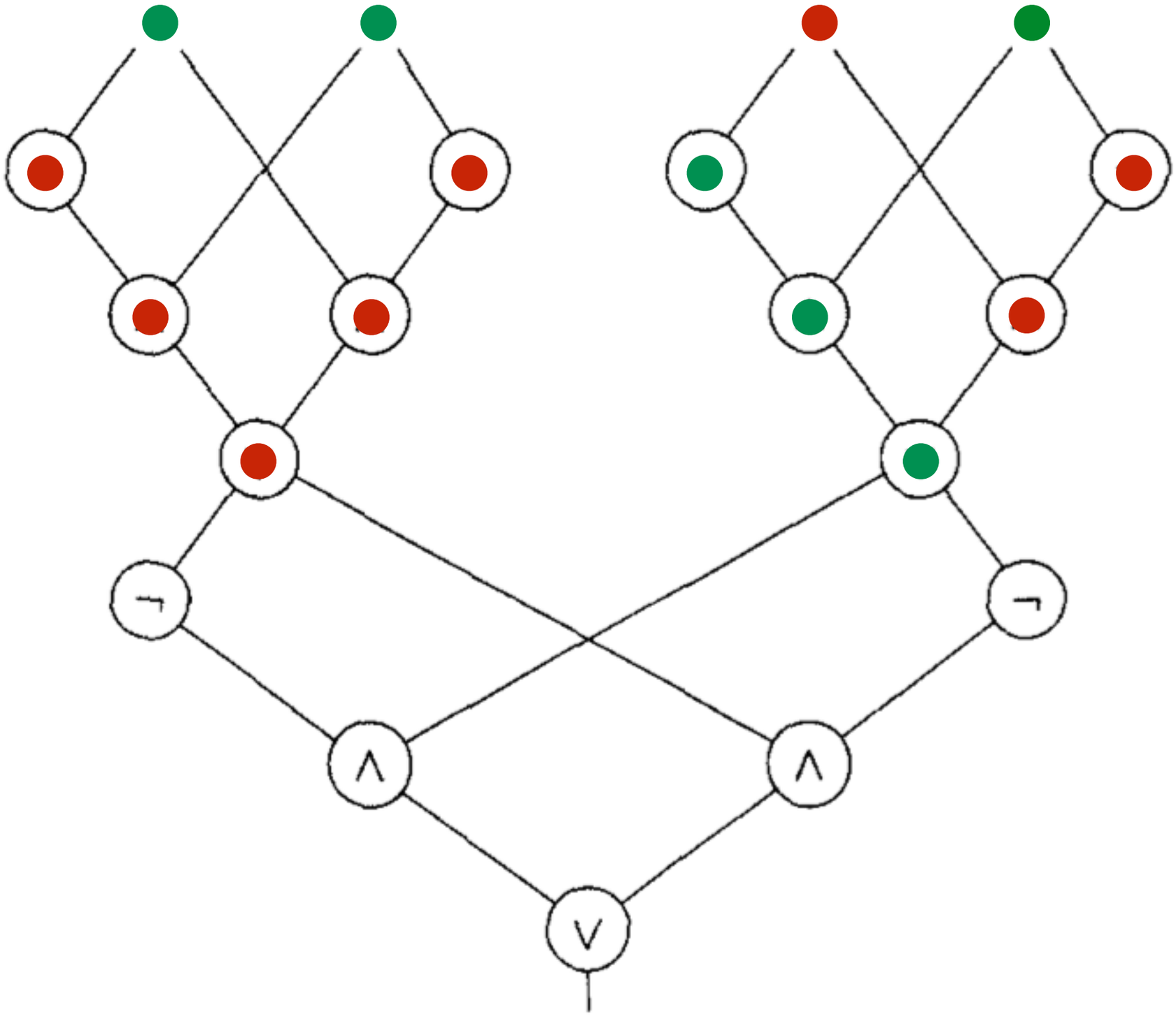

x_1
x_2
x_3
x_4
¬
¬
¬
¬
∧
∧
∧
∧
∨
∨
¬
¬
∧
∧
∨

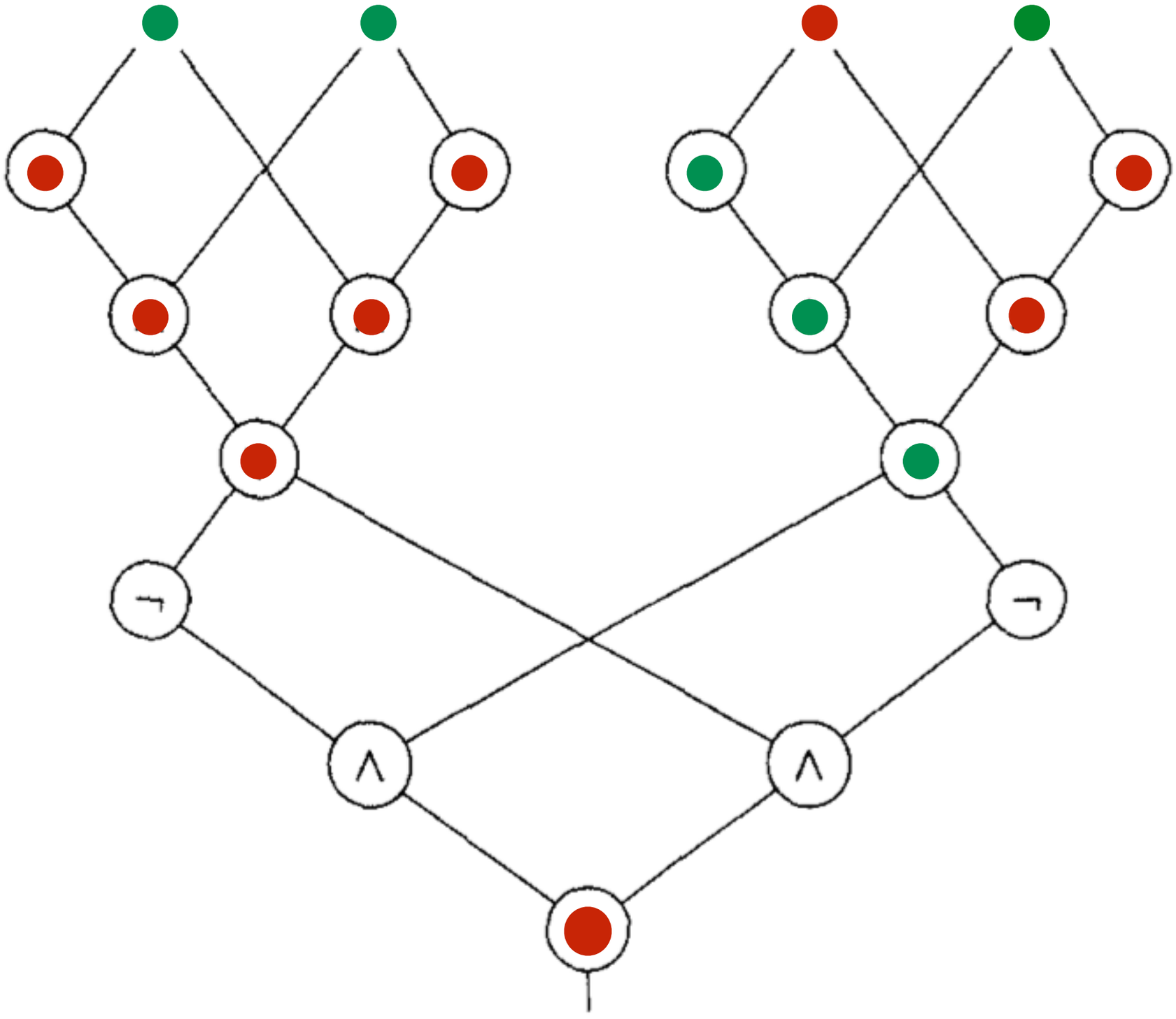




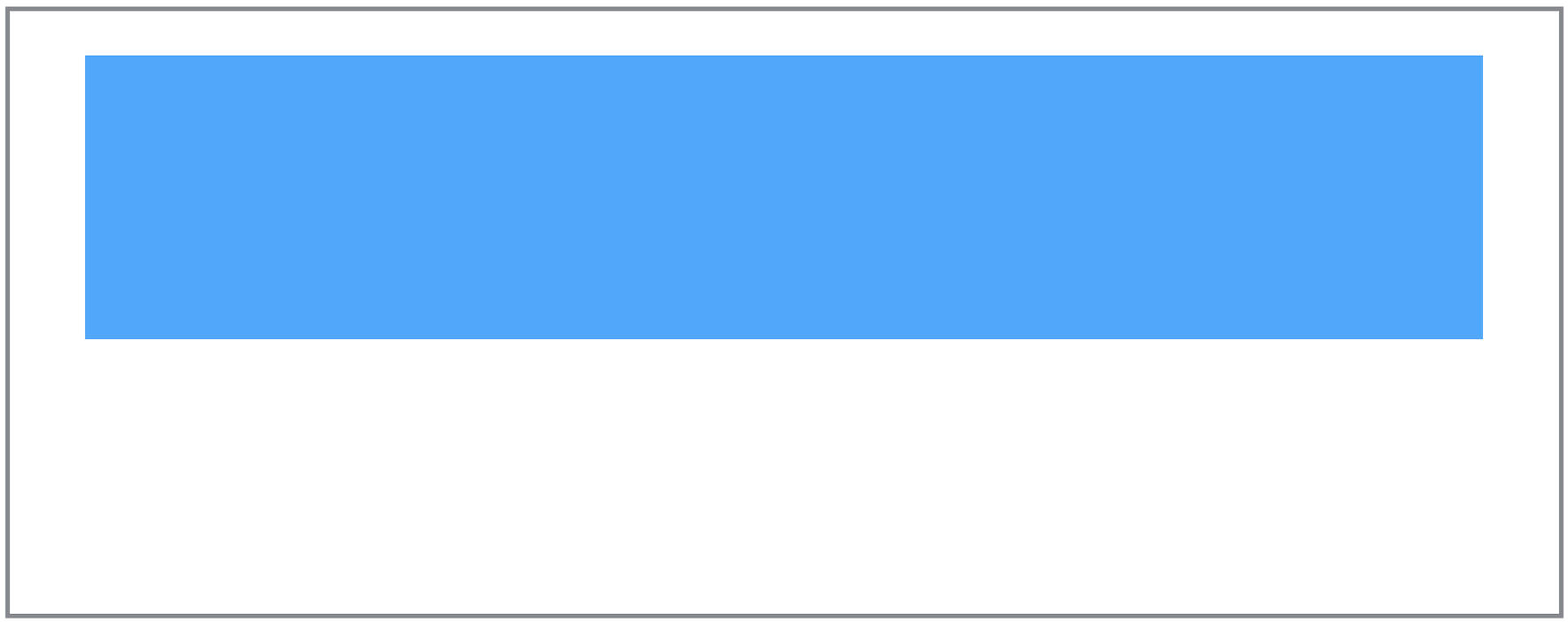




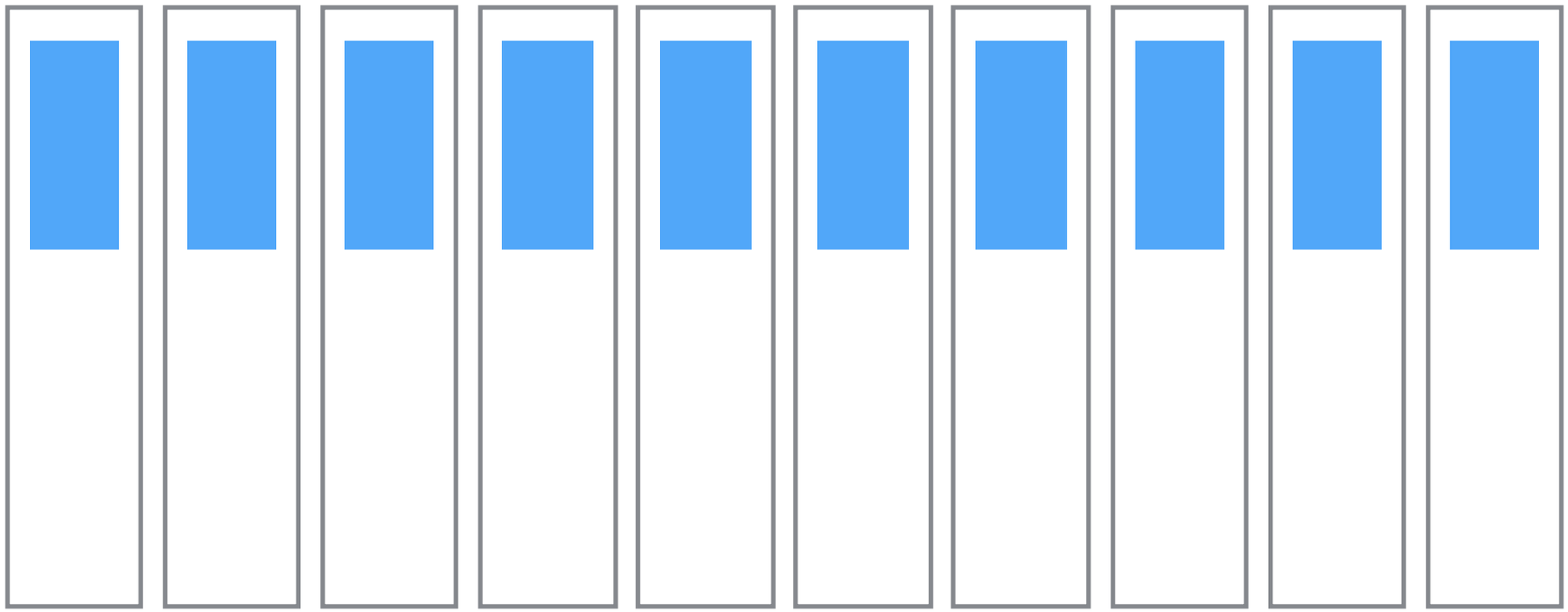




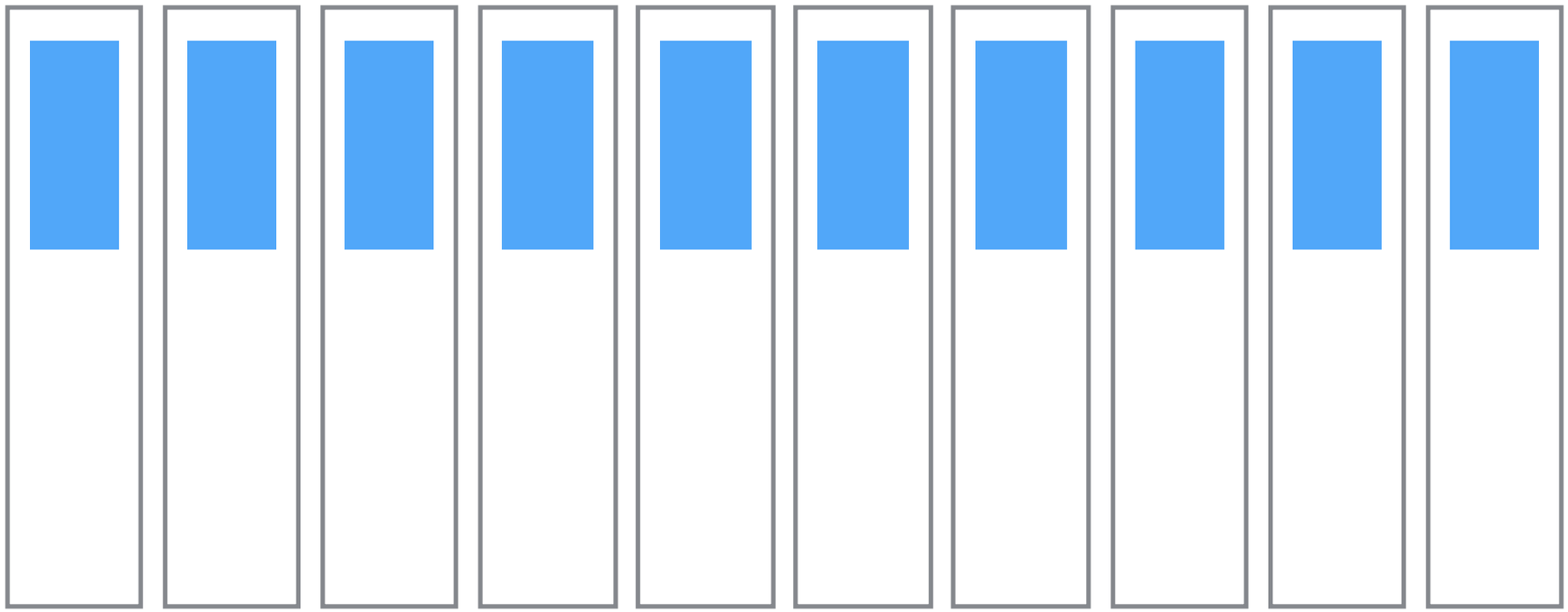
When does a circuit **accept** a language L?



When does a circuit **accept** a language L?

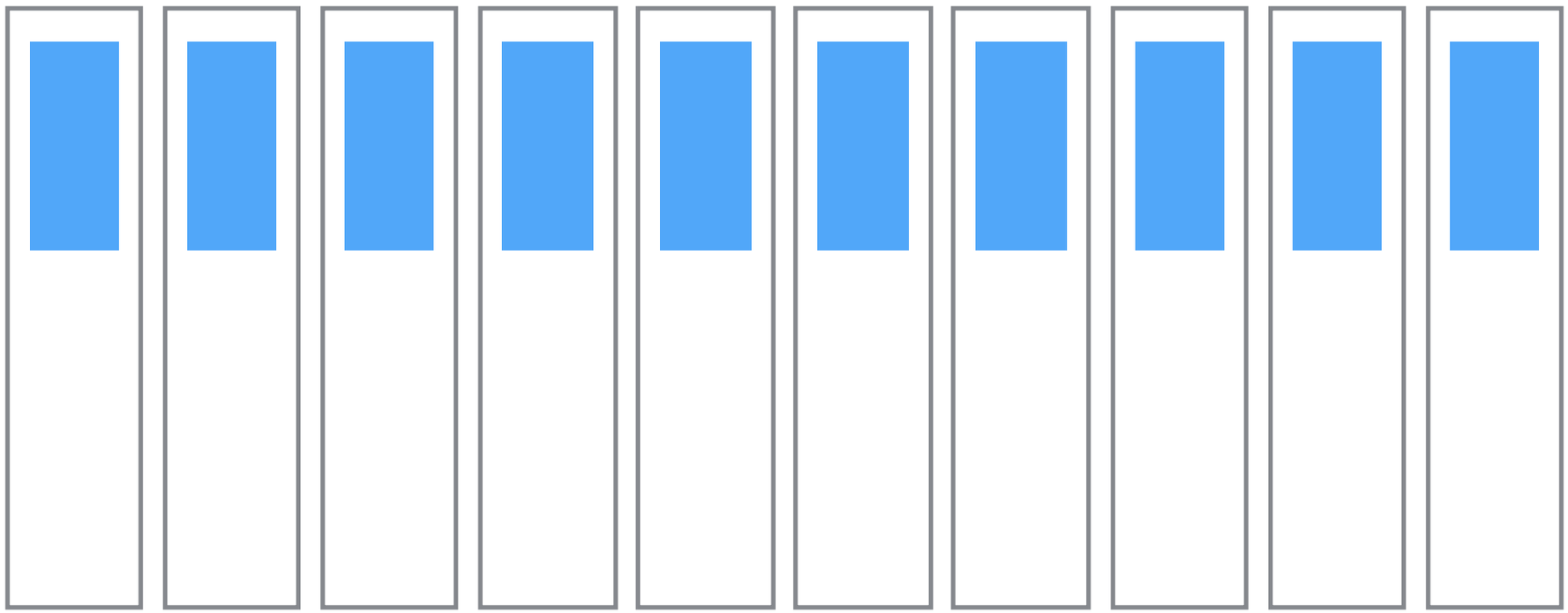


When does a circuit **accept** a language L?

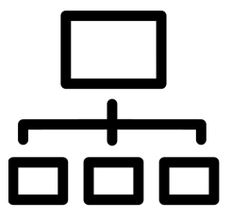


Strings
of length k

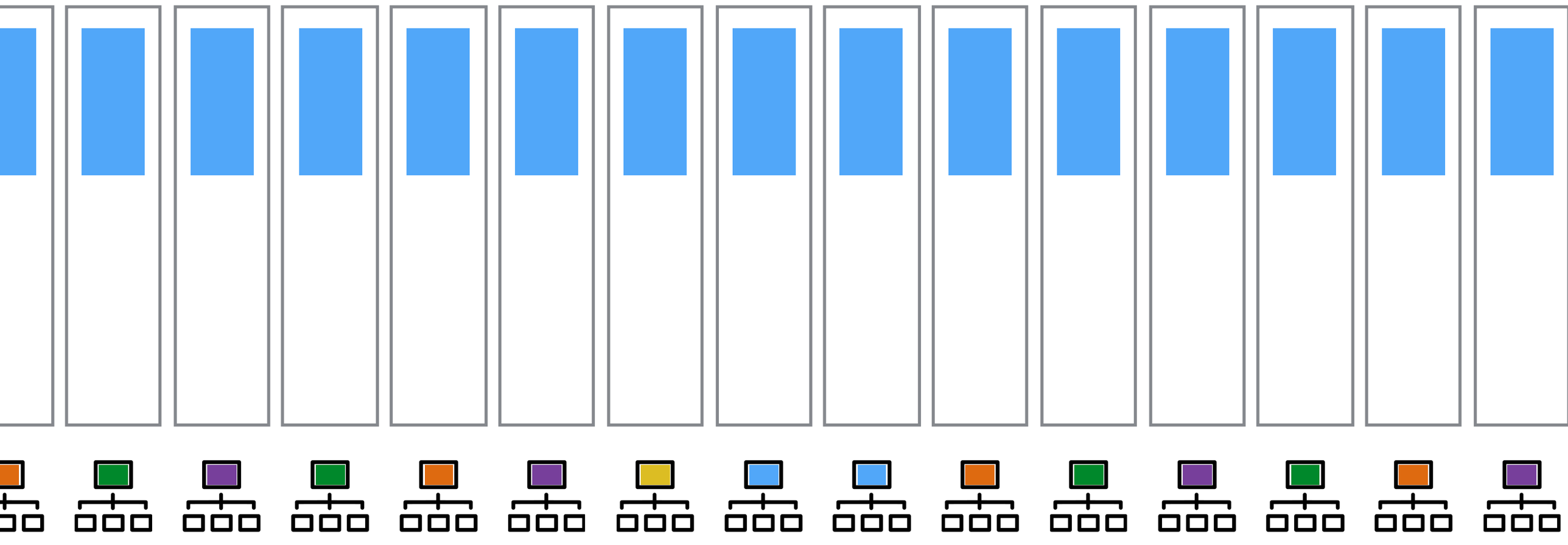
When does a circuit **accept** a language L?



Strings
of length k



A **family** of circuits **accepts a language** L if,
for *every* n, there is a circuit C_n
that outputs 1 precisely on inputs that belong to L.



Let L be any language.
Is there a circuit that accepts L?

Let L be any language.
Is there a circuit that accepts L?

Possibly of exponential size?

0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1
0 1 0 0 1 0 0 1 0
0 0 1 0 0 1 0 0 1
1 0 0 1 0 0 1 0 0
1 0 1 1 0 1 1 0 1
1 1 0 1 1 0 1 1 0
1 0 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1
0 1 0 0 1 0 0 1 0
0 0 1 0 0 1 0 0 1
1 0 0 1 0 0 1 0 0
1 0 1 1 0 1 1 0 1
1 1 0 1 1 0 1 1 0
1 0 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 1

Some of the strings in, say, L^c.

So, given enough resources, circuits
can figure out *everything*!

So, given enough resources, circuits
can figure out everything!

Let's try restricting their resources.

So, given enough resources, circuits can figure out everything!

Let's try restricting their resources.

$P_{/poly}$: The class of languages accepted by **polynomial-sized** circuit families.

So, given enough resources, circuits can figure out everything!

Let's try restricting their resources.

$P_{/poly}$: The class of languages accepted by **polynomial-sized** circuit families.

Eg. The language of **"all ones"** can be decided by a linear-sized circuit.

In fact, *any* **unary language**
can be decided by a linear-sized circuit.

In fact, *any* **unary language**
can be decided by a linear-sized circuit.

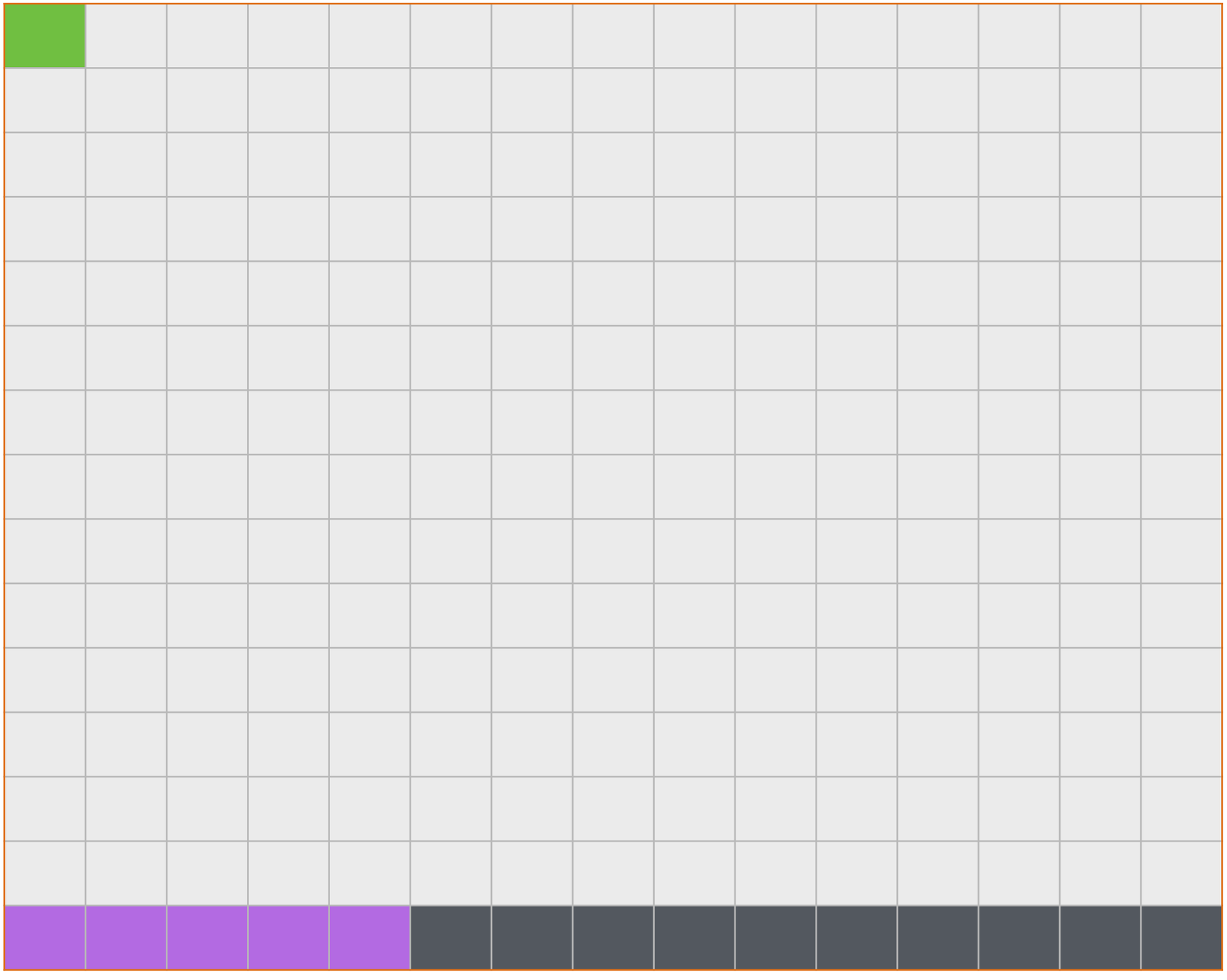
Think: How powerful does this make the class $\mathbf{P}_{/\mathbf{poly}}$?

In fact, *any* **unary language**
can be decided by a linear-sized circuit.

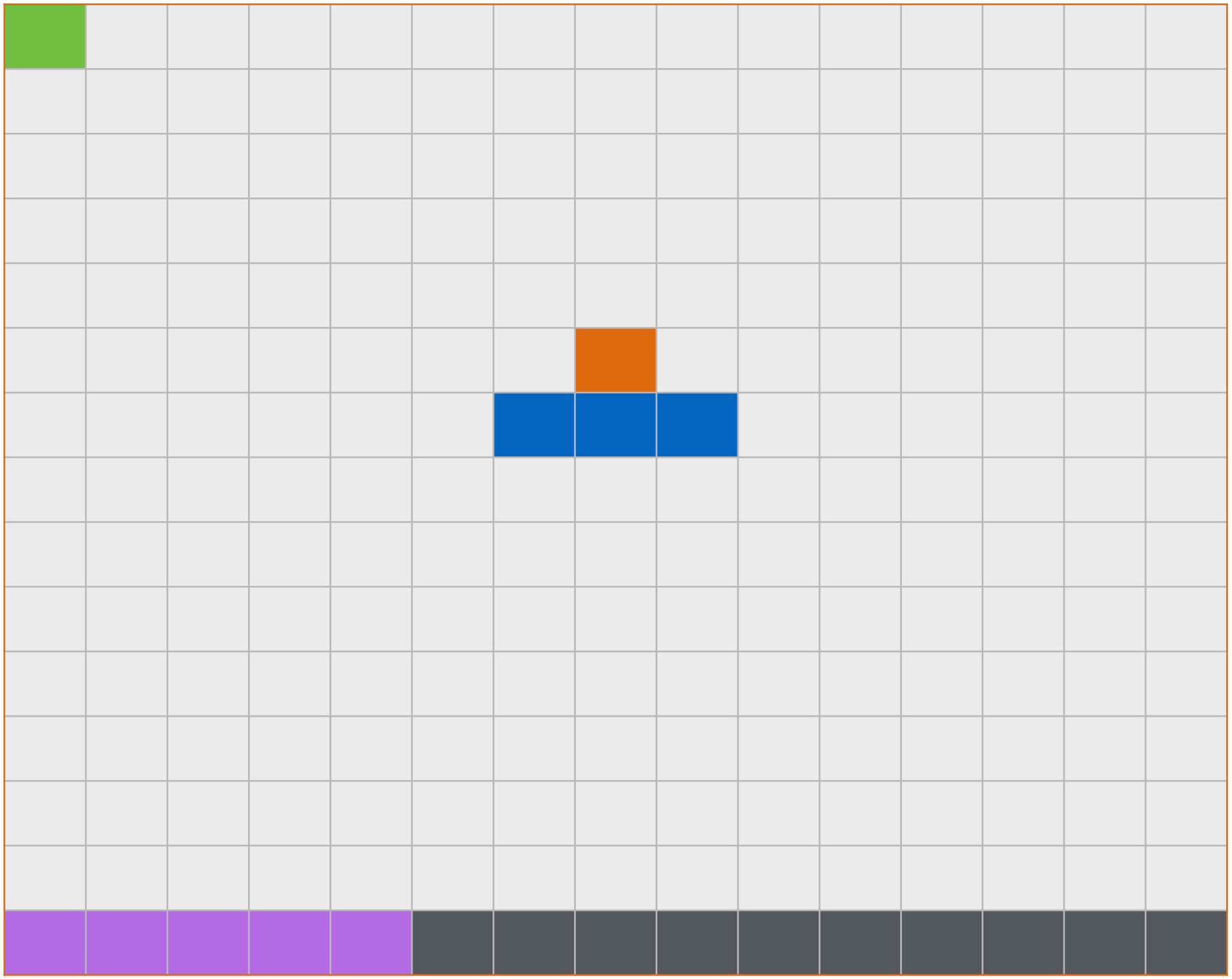
Think: How powerful does this make the class $\mathbf{P}_{/poly}$?

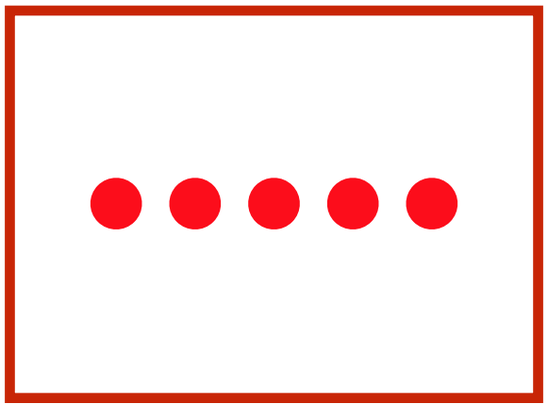
Hint: Can you think of an unary language that is undecidable?

P is contained in $\mathbf{P}_{/\mathbf{poly}}$.



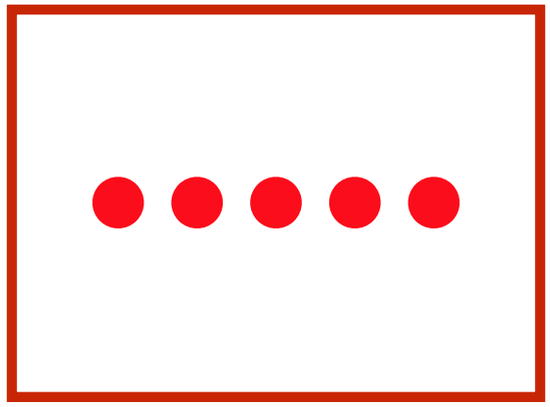
P is contained in $\mathbf{P}_{/\mathbf{poly}}$.





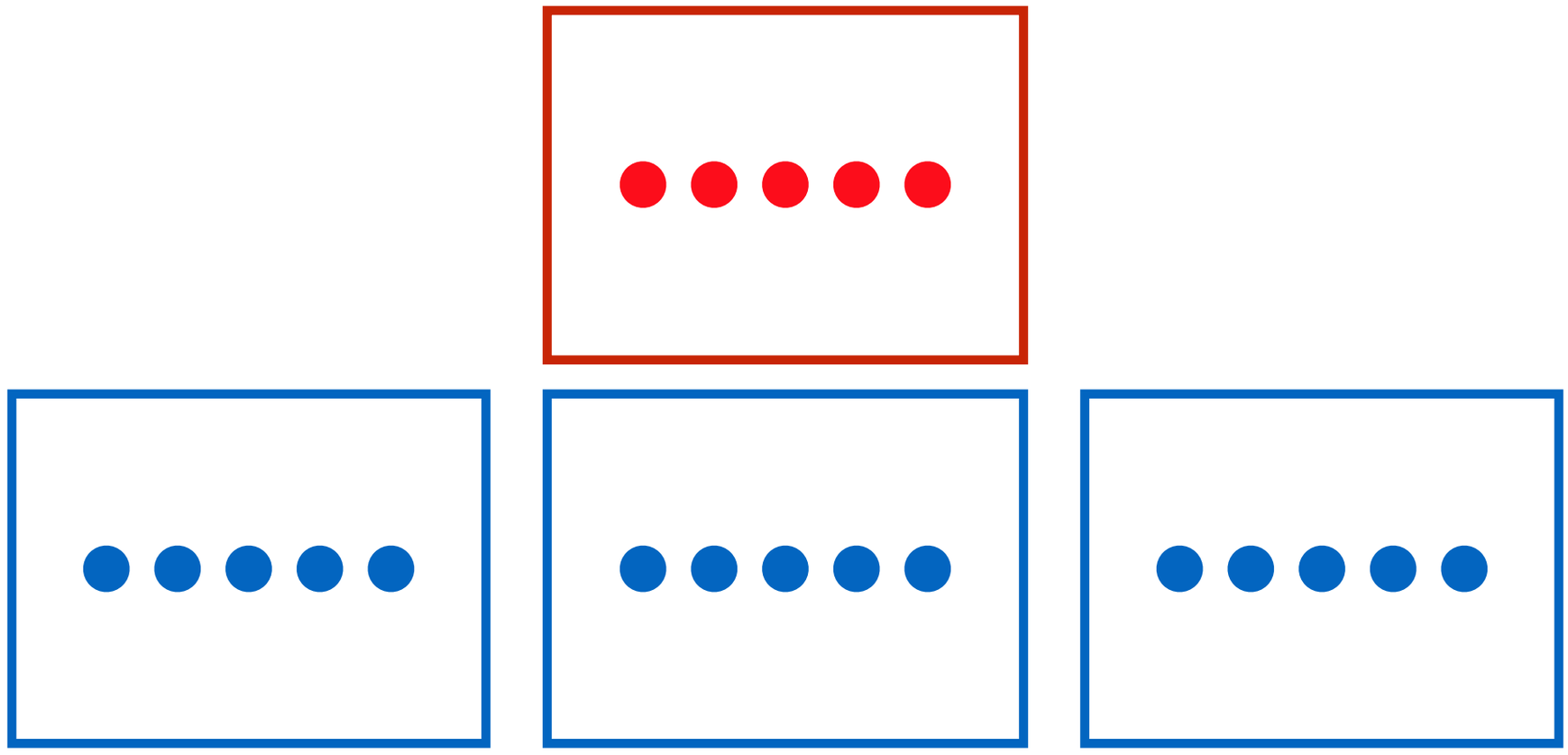
s_k - this entry has symbol k and does **not** have the head over it.

h_k - this entry has symbol k and the tape head **is** pointing here.



s_k - this entry has symbol k and does **not** have the head over it.

h_k - this entry has symbol k and the tape head **is** pointing here.



Turing Machines **with advice**

Turing Machines **with advice**

Turing Machines **with advice**

Turing Machines **with advice**

Turing Machines **with advice**

Turing Machines **with advice**

Turing Machines **with advice**

Turing Machines **with advice**

free advice!

$$\mathrm{DTIME}(T(n))/a(n)$$

DTIME(T(n))/a(n)

The class of languages decidable by
time-T(n) TMs with a(n) bits of advice,
contains every L such that
there exists a sequence$\{\alpha_n\}$ of strings
with $\alpha_n \in \{0,1\}^{a(n)}$ and a TM M satisfying:

$$M(x, \alpha_n) = 1 \Leftrightarrow x \in L$$

for every $x \in \{0, 1\}^n$, where on input
(x,α_n) the machine M runs for at most O(T(n)) steps.

In fact, *any* **unary language**
can be decided by a linear-sized circuit.

Think: How powerful does this make the class $\mathbf{P}_{/poly}$?

Hint: Can you think of an unary language that is undecidable?

In fact, *any* **unary language**
can be decided by a TM with very little advice.

Think: How powerful does this make the class $\mathbf{P}_{/poly}$?

Hint: Can you think of an unary language that is undecidable?

In fact, *any* **unary language**
can be decided by a TM with very little advice.

Polynomial time TMs with polynomial advice decide $\mathbf{P}_{/\mathbf{poly}}$.

$$\mathrm{P}_{/\mathrm{poly}} = \cup_{\mathrm{c},\mathrm{d}} \mathrm{DTIME}(\mathrm{n}^{\mathrm{c}})/\mathrm{n}^{\mathrm{d}}$$

One direction is easy - which one?

Summary

Circuits are a non-uniform model of computation.

They are akin to Turing Machines with advice.

$\mathbf{P}_{/poly}$ contains P, but also contain undecidable languages!