[Collatz in Dafny]

Hardware & Software Verification

John Wickerson

Lecture 5: SAT and SMT solving

Aside: Quantifiers

$\forall x \in \{3,4,5\}.\ P(x)$	$P(3) \wedge P(4) \wedge P(5)$
$\forall x.\ x \in \{3,4,5\} \Rightarrow P(x)$	$\mathrm{true} \wedge \mathrm{true} \wedge P(3) \wedge P(4) \wedge P(5) \wedge \mathrm{true} \wedge \mathrm{true} \wedge \ldots$
$\forall x.\ x \notin \{3,4,5\} \vee P(x)$	

$\exists x \in \{3,4,5\}.\ P(x)$	$P(3) \vee P(4) \vee P(5)$
$\exists x.\ x \in \{3,4,5\} \wedge P(x)$	$\mathrm{false} \vee \mathrm{false} \vee P(3) \vee P(4) \vee P(5) \vee \mathrm{false} \vee \mathrm{false} \vee \ldots$

Automatic proof

- We often rely on automatic provers:
  - e.g. in Dafny, to show that `invariant P` is preserved,
  - e.g. in Isabelle methods like `by auto`.
- How do these automatic provers work?

SAT queries

- Simple case: proofs about Boolean statements.

SAT queries

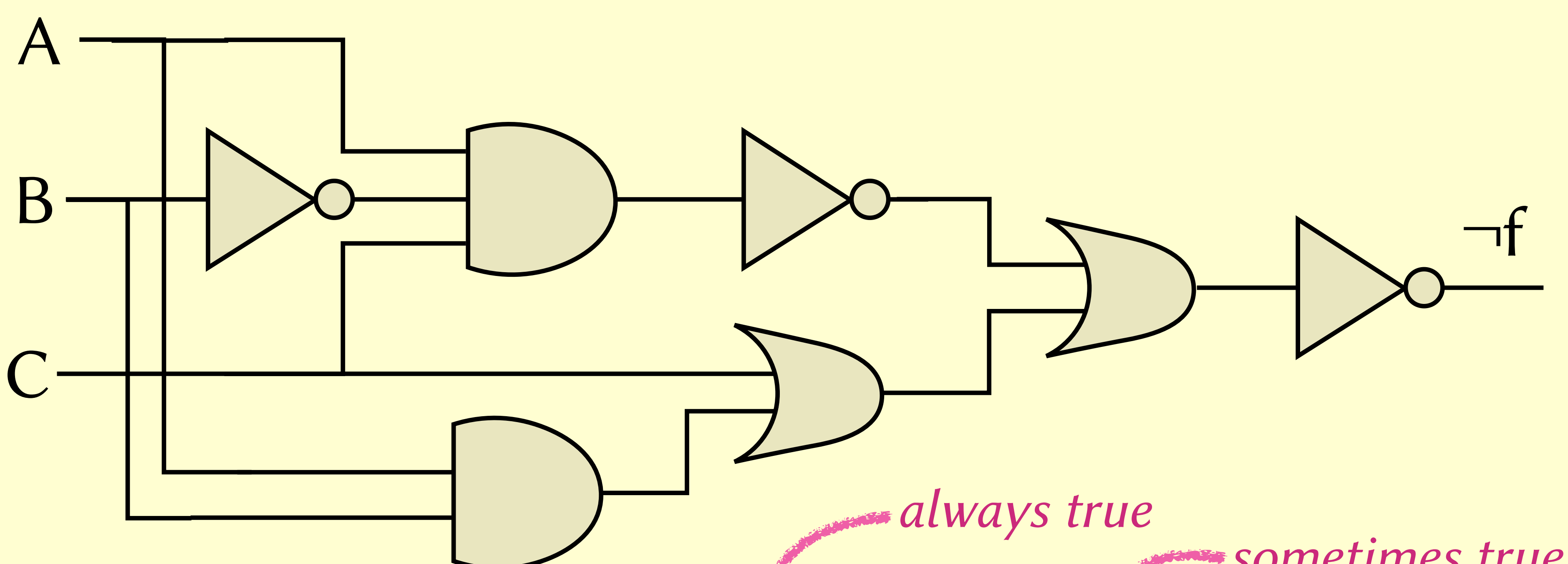
- Simple case: proofs about Boolean statements.
  - $f = ((A \land \neg B \land C) \Longrightarrow (C \lor (B \land A)))$

SAT queries

- Simple case: proofs about Boolean statements.
  - $f = (\neg(A \land \neg B \land C) \lor (C \lor (B \land A)))$

SAT queries

- Simple case: proofs about Boolean statements.
  - $\neg f = \neg(\neg(A \wedge \neg B \wedge C) \vee (C \vee (B \wedge A)))$



A	B	C	¬f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

always true *sometimes true*

A formula can be VALID, SATISFIABLE, UNSATISFIABLE, or INVALID.

always false *sometimes false*

SAT solving

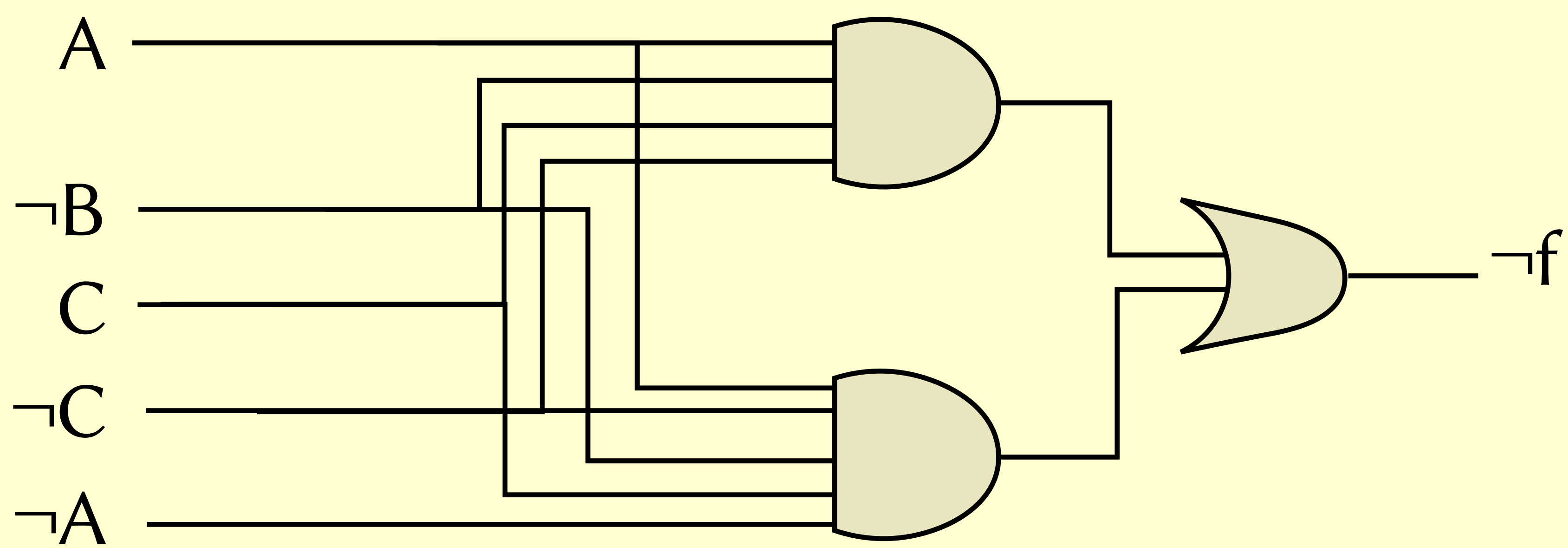
- A simple algorithm:

```
for A in {0, 1}:
  for B in {0, 1}:
    for C in {0, 1}:
      if ¬f(A,B,C) = 1:
        return ("SAT", [A,B,C])
return ("UNSAT")
```

- **Problem:** if the formula has N variables, this algorithm has exponential time-complexity, $O(2^N)$. ☹️

SAT solving

- **Idea:** Use de Morgan's laws to convert formula into *disjunctive normal form.*



- **Hooray:** checking satisfiability becomes trivial!

SAT solving

- **Problem:** converting into disjunctive normal form has exponential time-complexity.

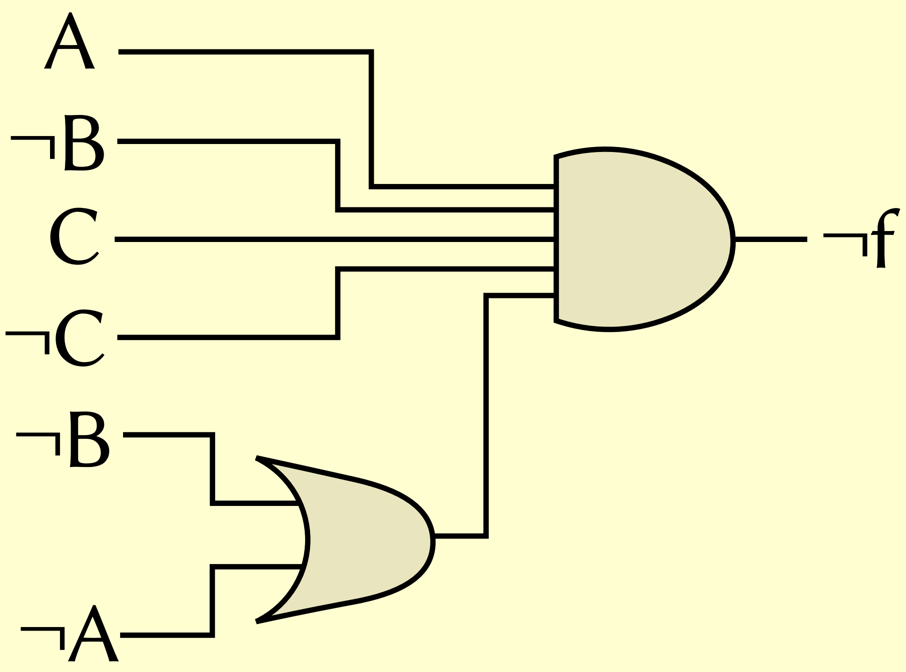
$$A \wedge \neg B \wedge C \wedge D \wedge \neg E \wedge F \wedge (G \vee H)$$

$$\Downarrow$$

$$(A \wedge \neg B \wedge C \wedge D \wedge \neg E \wedge F \wedge G) \vee (A \wedge \neg B \wedge C \wedge D \wedge \neg E \wedge F \wedge H)$$

SAT solving

- **Idea:** Use de Morgan's laws to convert formula into *conjunctive normal form.*



SAT solving

- **Problem:** converting into conjunctive normal form still has exponential time-complexity.

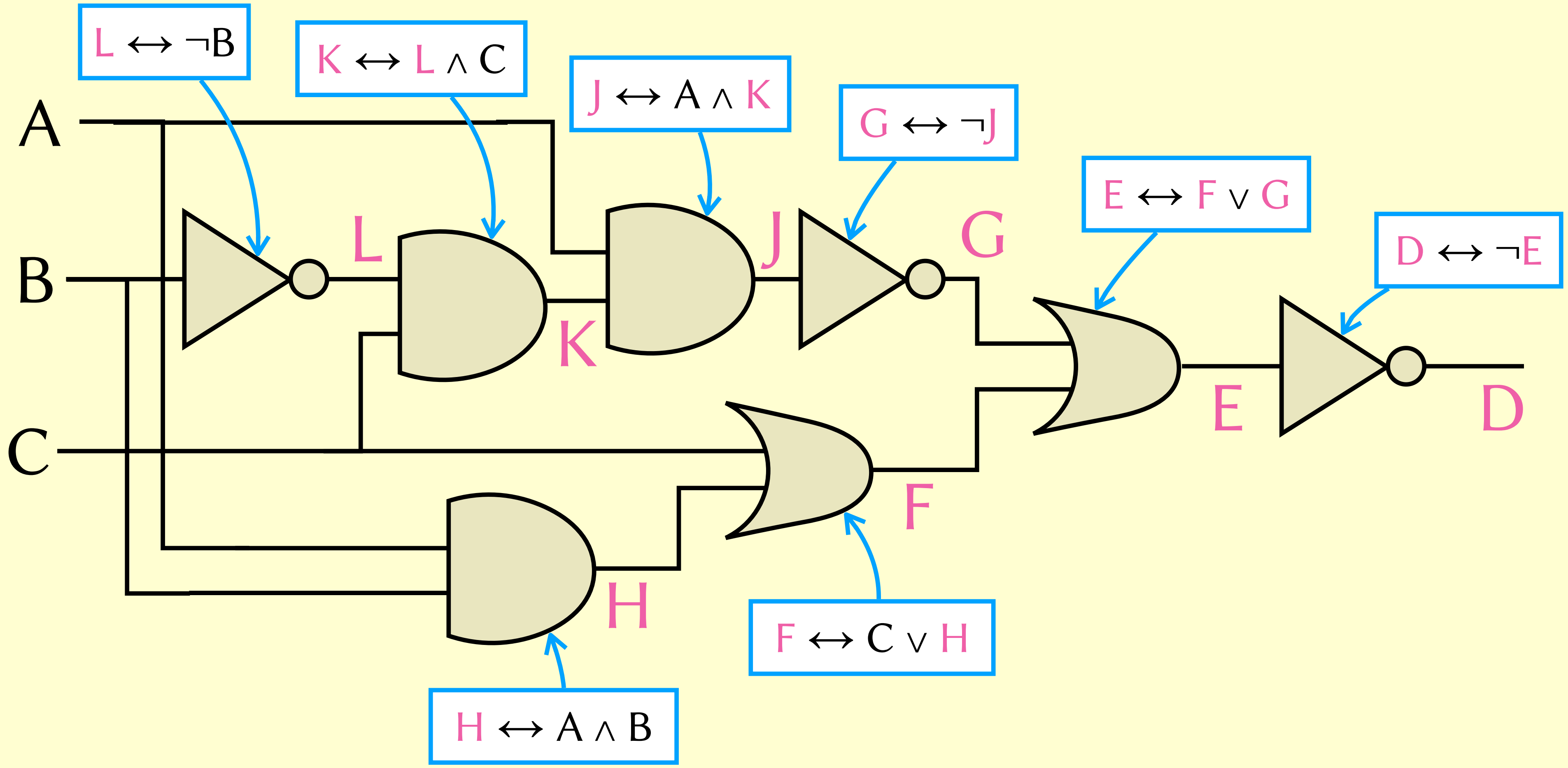
$$A \vee \neg B \vee C \vee D \vee \neg E \vee F \vee (G \wedge H)$$

$$\Downarrow$$

$$(A \vee \neg B \vee C \vee D \vee \neg E \vee F \vee G) \wedge (A \vee \neg B \vee C \vee D \vee \neg E \vee F \vee H)$$

SAT solving

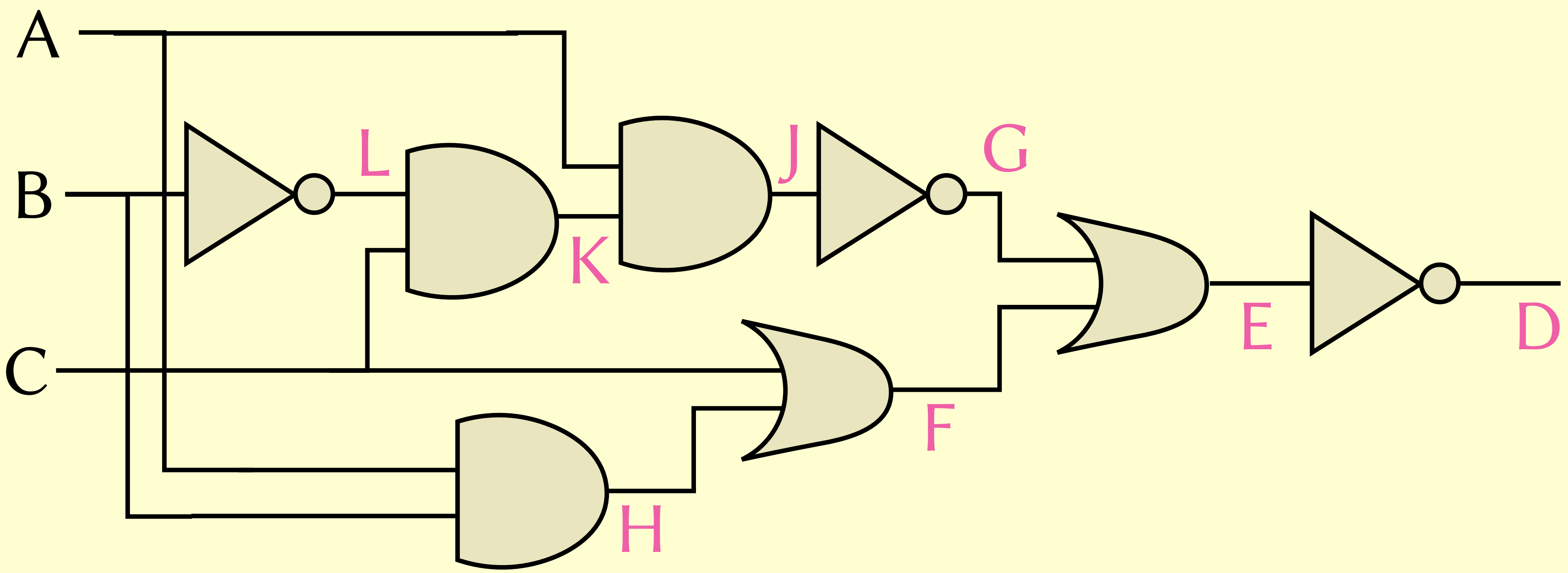
- **Solution:** the Tseitin transform. Make a fresh variable for each wire. Associate each gate with an equality.



Gregory Tseitin
1936–2022

SAT solving

- **Solution:** the Tseitin transform. Make a fresh variable for each wire. Associate each gate with an equality.



$\boxed{L \leftrightarrow \neg B} \wedge$

$\boxed{K \leftrightarrow L \wedge C} \wedge$

$\boxed{J \leftrightarrow A \wedge K} \wedge$

$\boxed{G \leftrightarrow \neg J} \wedge$

$\boxed{E \leftrightarrow F \vee G} \wedge$

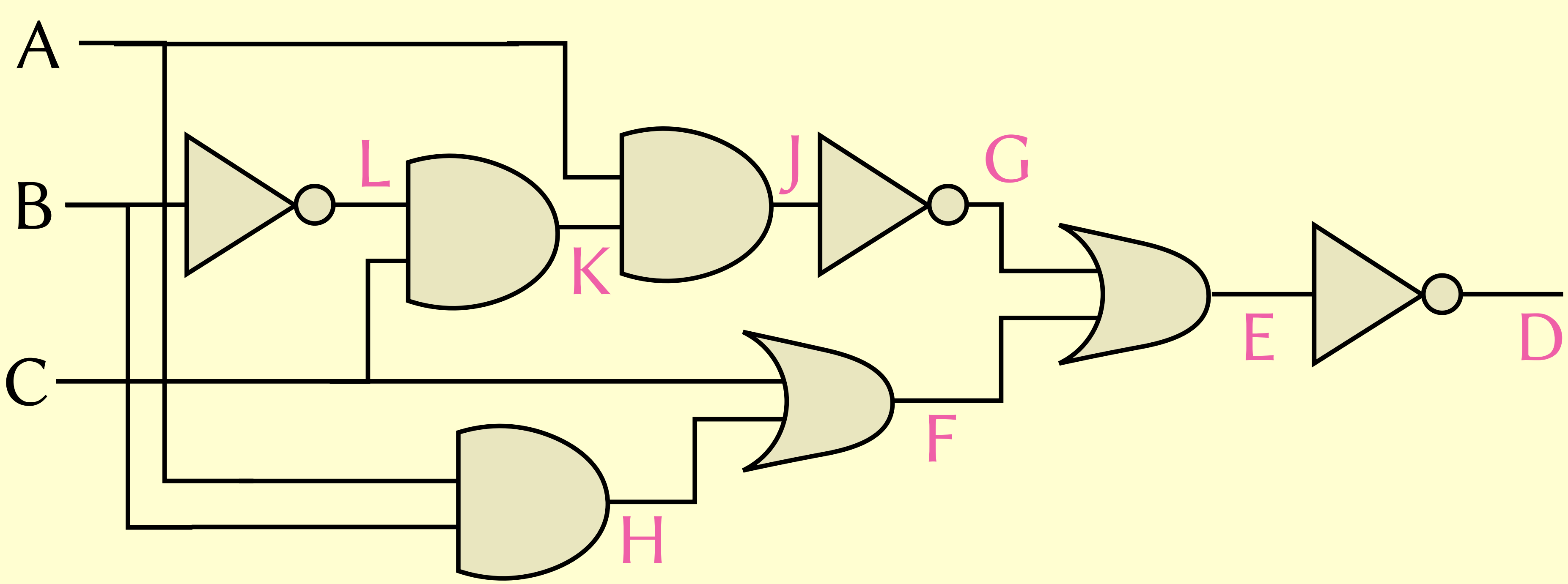
$\boxed{D \leftrightarrow \neg E} \wedge$

$\boxed{F \leftrightarrow C \vee H} \wedge$

$\boxed{H \leftrightarrow A \wedge B} \wedge D$

SAT solving

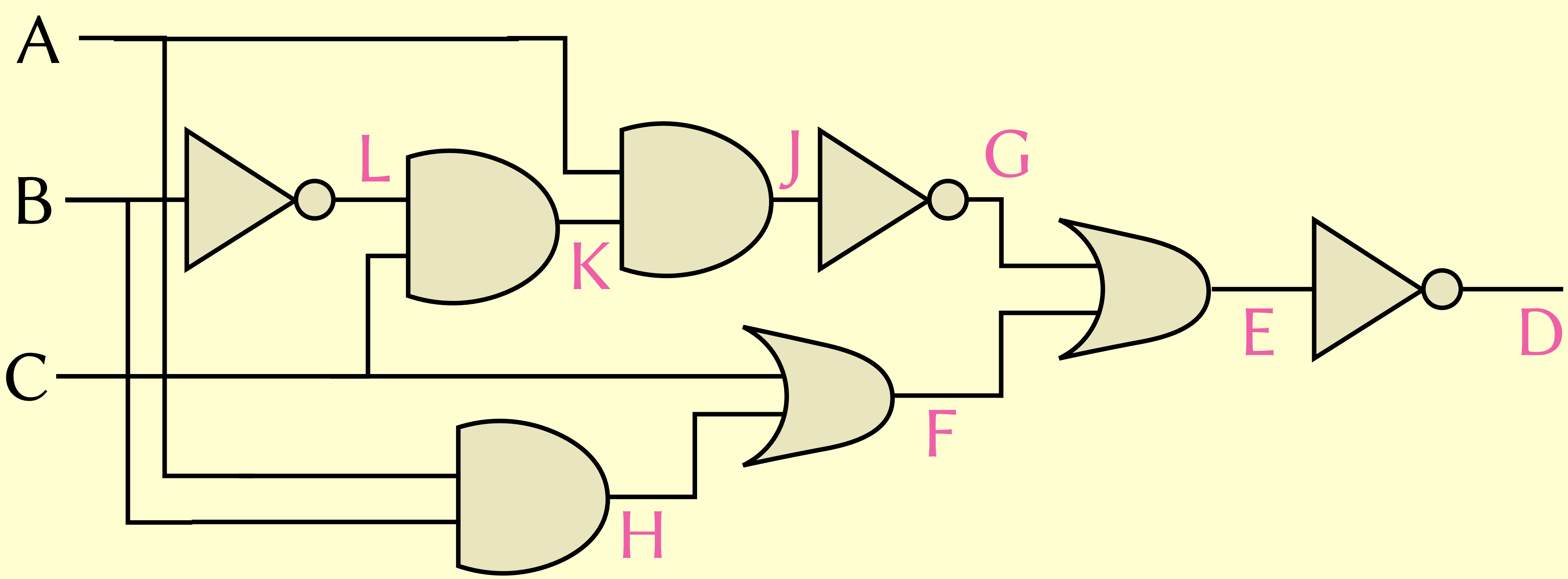
- **Solution:** the Tseitin transform. Make a fresh variable for each wire. Associate each gate with an equality.



$$\boxed{L \leftrightarrow \neg B} \wedge \boxed{K \leftrightarrow L \wedge C} \wedge \boxed{J \leftrightarrow A \wedge K} \wedge \boxed{G \leftrightarrow \neg J} \wedge \boxed{E \leftrightarrow F \vee G} \wedge \boxed{D \leftrightarrow \neg E} \wedge \boxed{F \leftrightarrow C \vee H} \wedge \boxed{H \rightarrow A \wedge B} \wedge \boxed{A \wedge B \rightarrow H} \wedge D$$

SAT solving

- **Solution:** the Tseitin transform. Make a fresh variable for each wire. Associate each gate with an equality.



$\boxed{L \leftrightarrow \neg B} \wedge$

$\boxed{K \leftrightarrow L \wedge C} \wedge$

$\boxed{J \leftrightarrow A \wedge K} \wedge$

$\boxed{G \leftrightarrow \neg J} \wedge$

$\boxed{E \leftrightarrow F \vee G} \wedge$

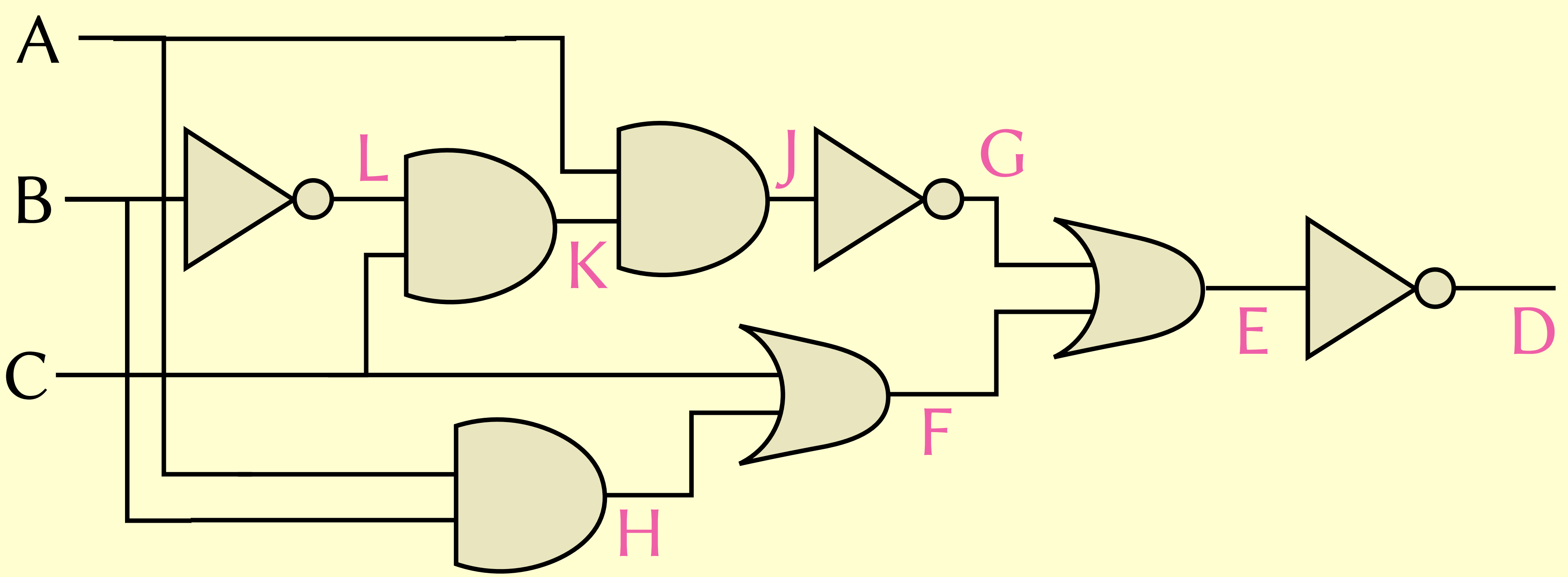
$\boxed{D \leftrightarrow \neg E} \wedge$

$\boxed{F \leftrightarrow C \vee H} \wedge$

$\boxed{\neg H \vee (A \wedge B)} \wedge \boxed{\neg(A \wedge B) \vee H} \wedge D$

SAT solving

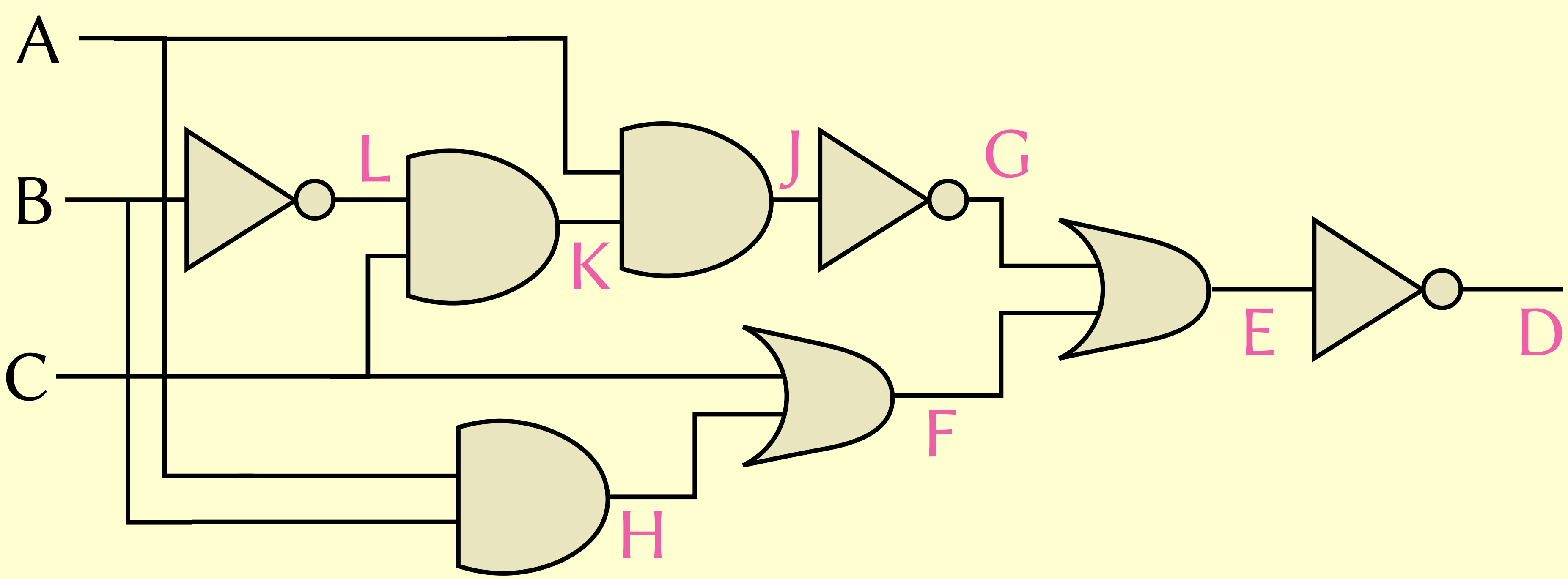
- **Solution:** the Tseitin transform. Make a fresh variable for each wire. Associate each gate with an equality.



$$\boxed{L \leftrightarrow \neg B} \wedge$$
$$\boxed{K \leftrightarrow L \wedge C} \wedge$$
$$\boxed{J \leftrightarrow A \wedge K} \wedge$$
$$\boxed{G \leftrightarrow \neg J} \wedge$$
$$\boxed{E \leftrightarrow F \vee G} \wedge$$
$$\boxed{D \leftrightarrow \neg E} \wedge$$
$$\boxed{F \leftrightarrow C \vee H} \wedge$$
$$\boxed{\neg H \vee A} \wedge \boxed{\neg H \vee B} \wedge \boxed{\neg(A \wedge B) \vee H} \wedge D$$

SAT solving

- **Solution:** the Tseitin transform. Make a fresh variable for each wire. Associate each gate with an equality.

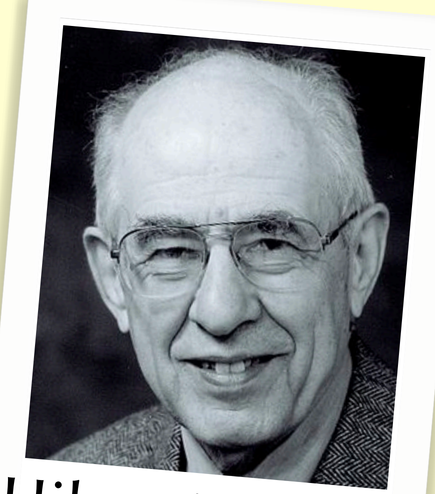


$$\boxed{L \leftrightarrow \neg B} \;\wedge$$
$$\boxed{K \leftrightarrow L \wedge C} \;\wedge$$
$$\boxed{J \leftrightarrow A \wedge K} \;\wedge$$
$$\boxed{G \leftrightarrow \neg J} \;\wedge$$
$$\boxed{E \leftrightarrow F \vee G} \;\wedge$$
$$\boxed{D \leftrightarrow \neg E} \;\wedge$$
$$\boxed{F \leftrightarrow C \vee H} \;\wedge$$
$$\boxed{\neg H \vee A} \;\wedge\; \boxed{\neg H \vee B} \;\wedge\; \boxed{\neg A \vee \neg B \vee H} \;\wedge\; D$$

SAT solving

- **Problem:** the satisfiability problem for CNF is difficult (unlike for DNF).
- **Solution:** Davis and Putnam to the rescue!

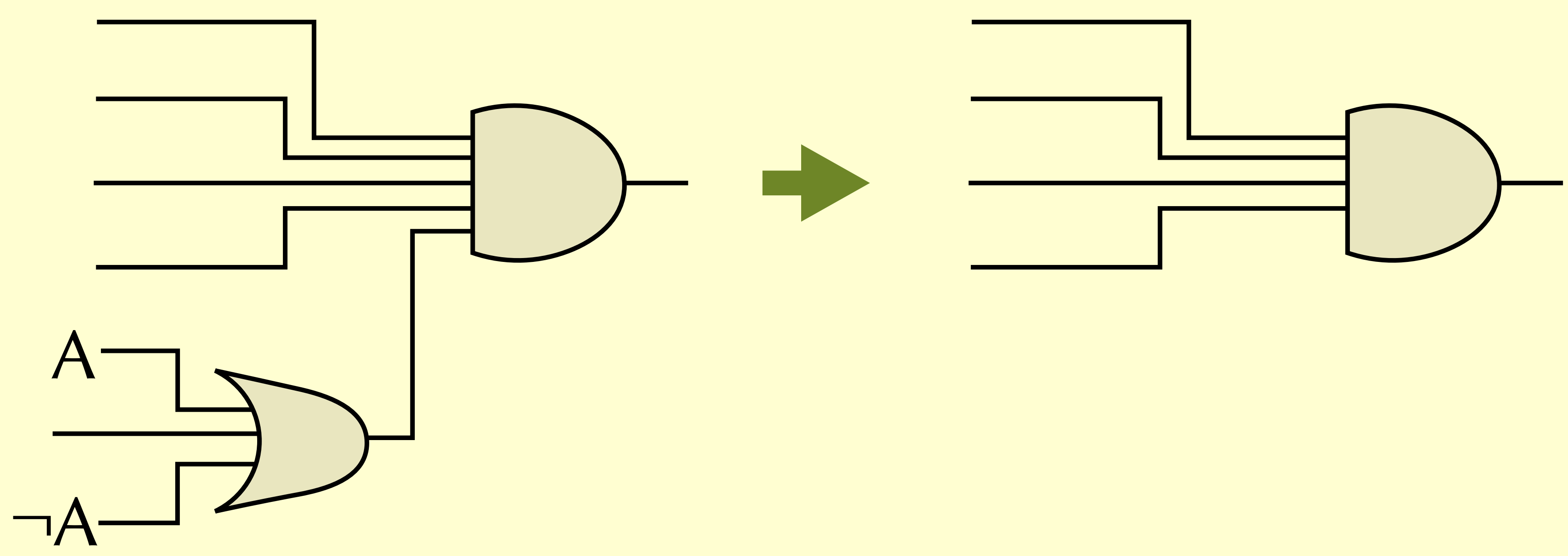
Martin Davis
1928–2023



Hilary Putnam
1926–2016

The DP method

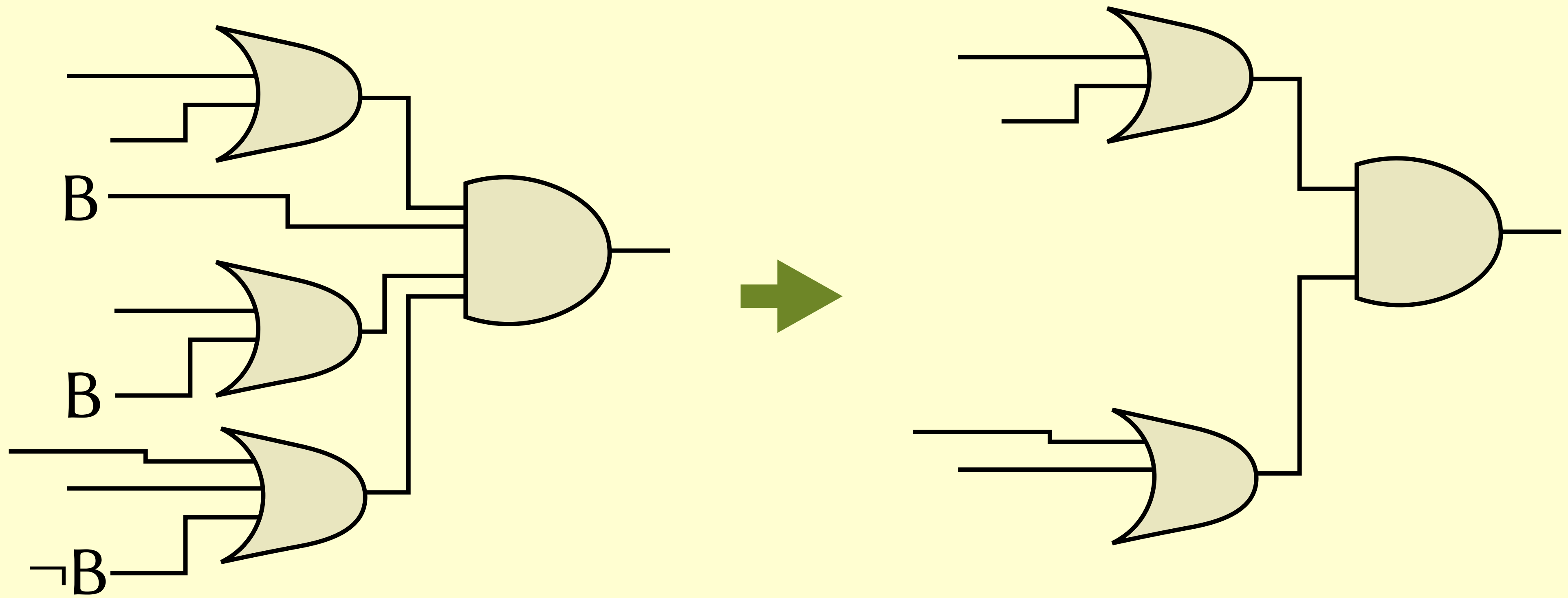
1. If an OR-gate takes both L and ¬L, delete it.



The DP method

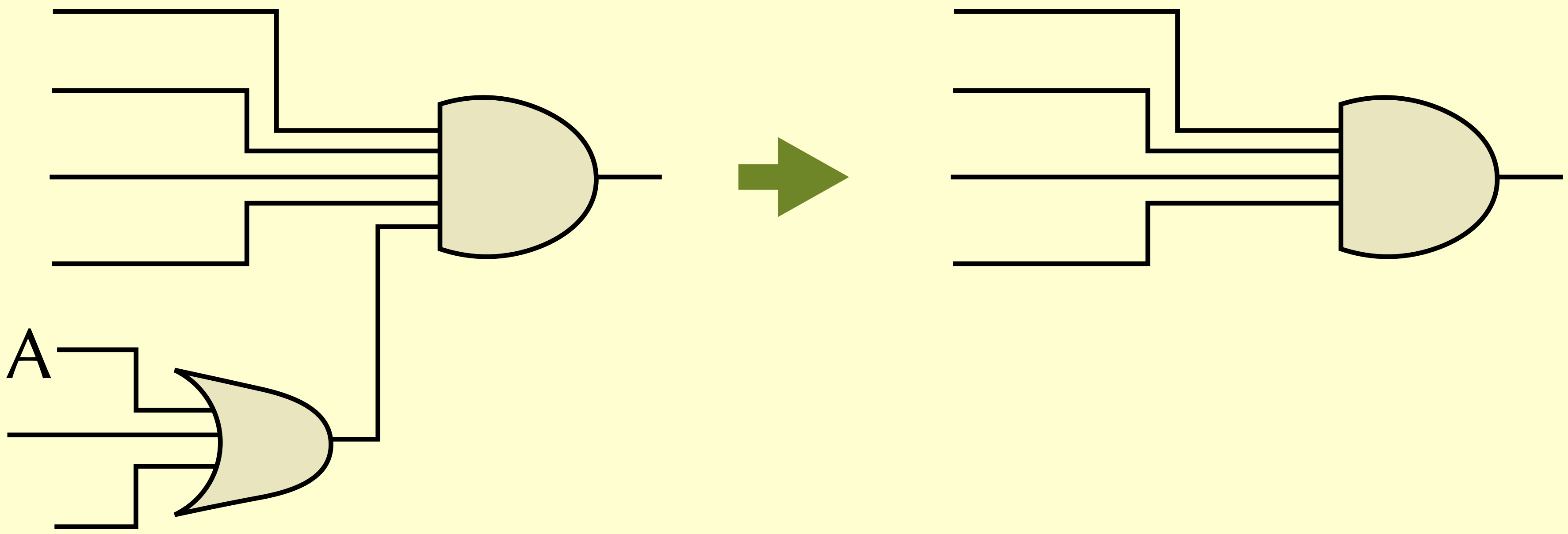
1. If an OR-gate takes both L and ¬L, delete it.

2. If L is connected directly to the AND-gate, delete it, delete all OR-gates that take L, and delete any connections to ¬L.
(The solution, if it exists, will surely involve setting L=1.)



The DP method

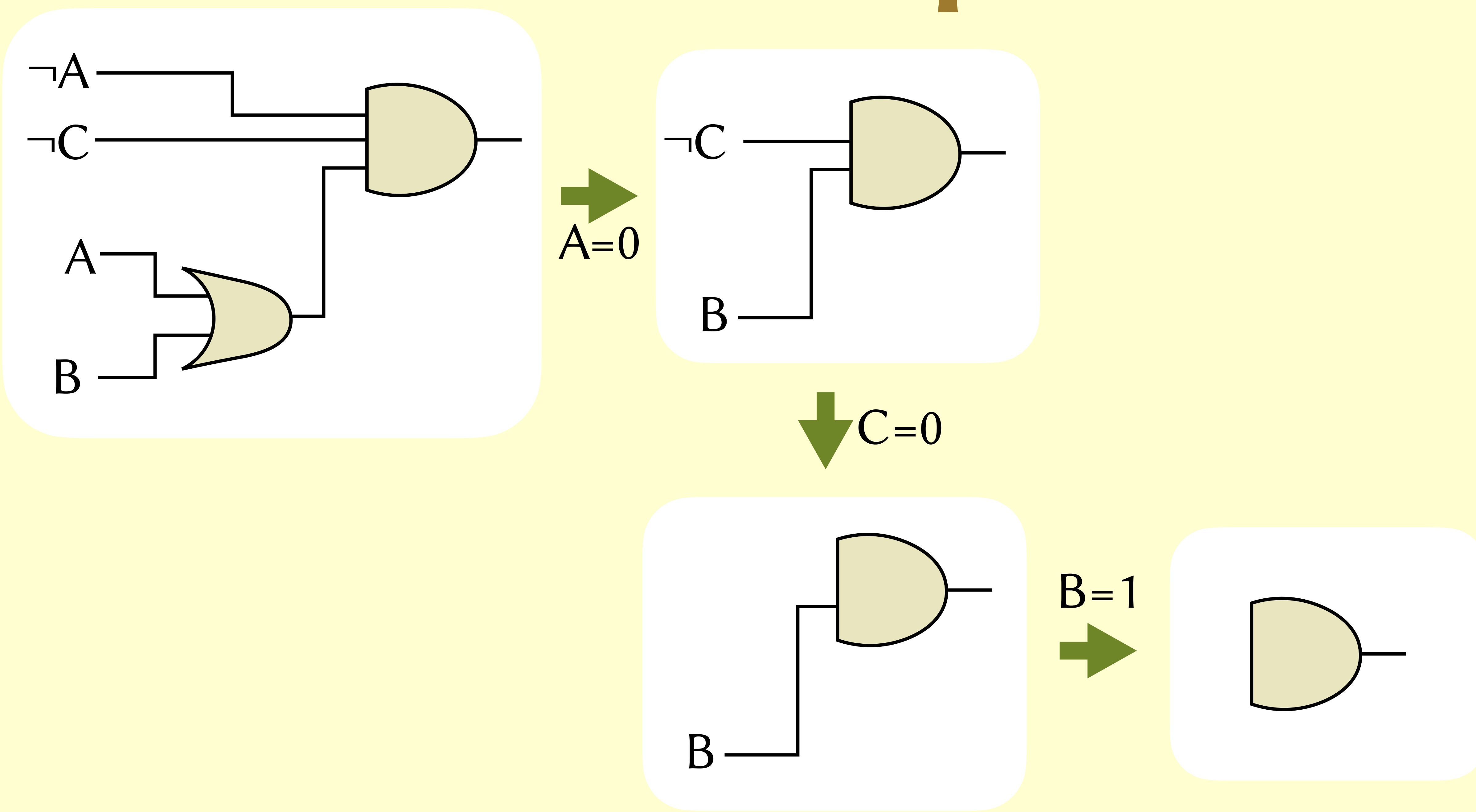
1. If an OR-gate takes both L and ¬L, delete it.
2. If L is connected directly to the AND-gate, delete it, delete all OR-gates that take L, and delete any connections to ¬L.
   (The solution, if it exists, will surely involve setting L=1.)
3. If L is unused, delete all OR-gates that take ¬L.
   (The solution, if it exists, will surely involve setting L=0.)



The DP method

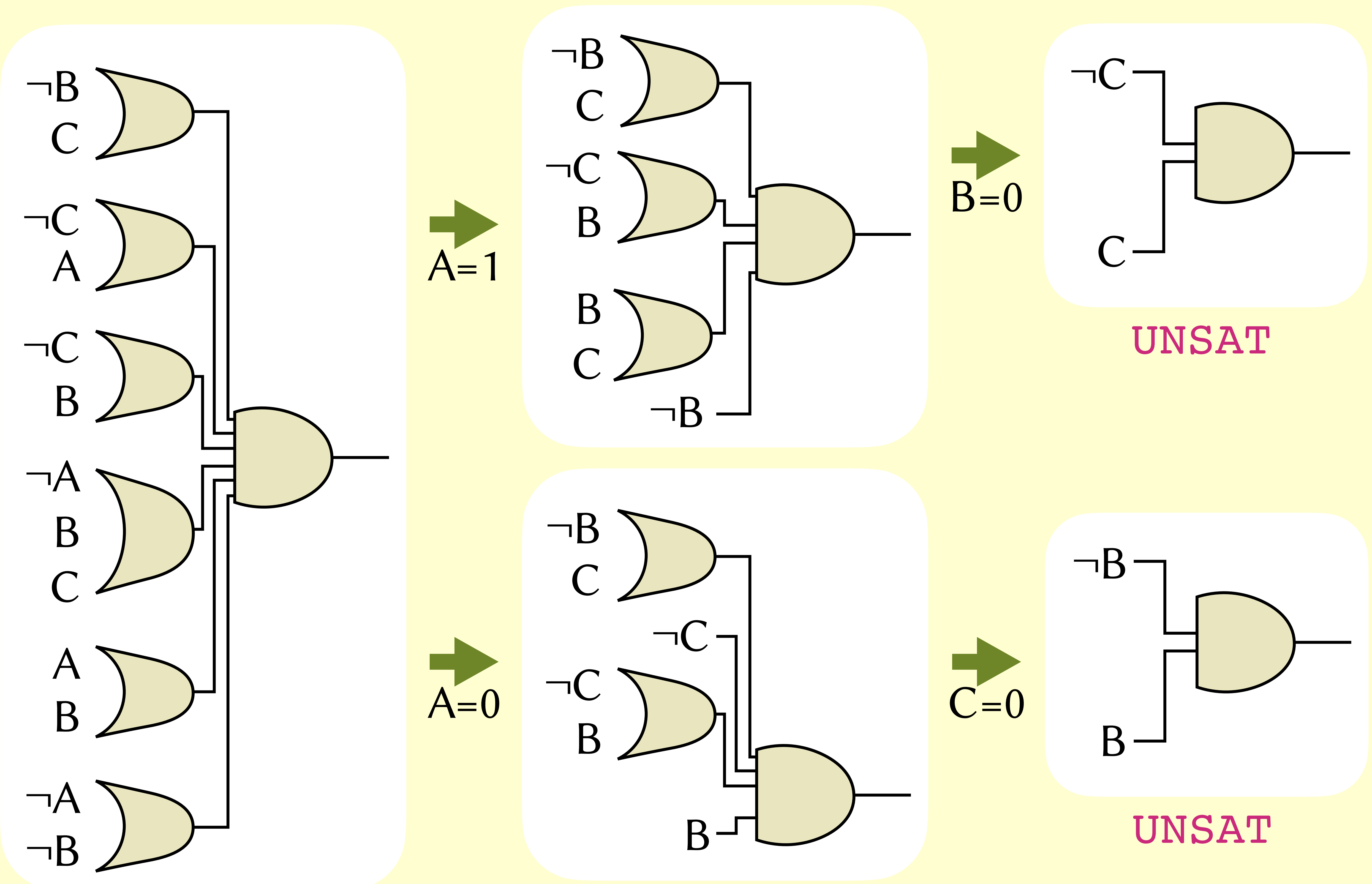
1. If an OR-gate takes both L and ¬L, delete it.
2. If L is connected directly to the AND-gate, delete it, delete all OR-gates that take L, and delete any connections to ¬L.
   (The solution, if it exists, will surely involve setting L=1.)
3. If L is unused, delete all OR-gates that take ¬L.
   (The solution, if it exists, will surely involve setting L=0.)
4. If any OR-gate has no inputs, the formula is false.
5. If the AND-gate has no inputs, the formula is true.
6. Pick a literal L and repeat the above for the cases L=0 and L=1.

DP example 1



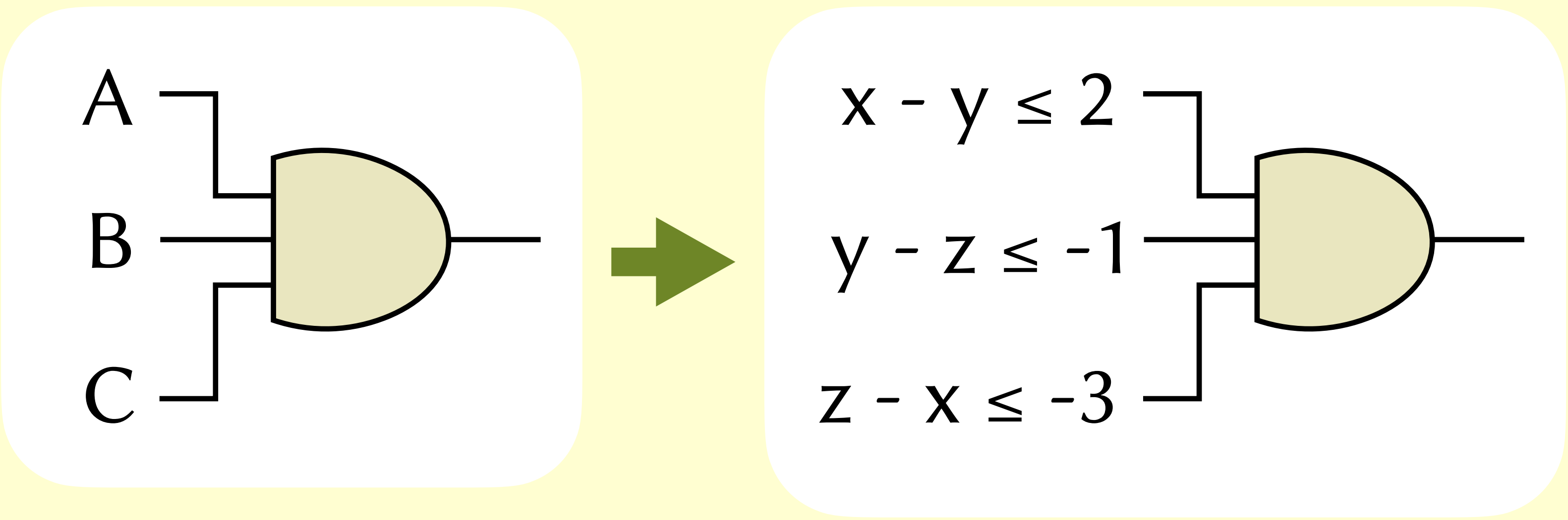
SAT, A=0, B=1, C=0

DP example 2



Towards SMT solving

- We can now prove basic Boolean formulas. But what about proving something like $A \times (B + C) = A \times B + A \times C$?
- If these are 32-bit integers, we could make this a SAT problem by treating each variable as 32 Boolean variables and encoding the rules of Boolean arithmetic.
- Or we can move up to SMT: *satisfiability modulo theories.*



Some theories

- **Equality and uninterpreted functions**, which knows that x=y and y=z implies x=z, and that x=y implies f(x)=f(y).
- **Difference logic**, where statements take the form $x - y \leq c$.
- **Presburger arithmetic**, which allows statements about naturals containing +, 0, 1, and =. For instance, n is a McNugget number if $\exists x\ y\ z.\ n = 6x + 9y + 20z$.

Mojżesz Presburger
1904–c.1943

Some theories

- **Equality and uninterpreted functions**, which knows that x=y and y=z implies x=z, and that x=y implies f(x)=f(y).

- **Difference logic**, where statements take the form x - y ≤ c.

- **Presburger arithmetic**, which allows statements about naturals containing +, 0, 1, and =. For instance, n is a McNugget number if ∃x y z. n = 6x + 9y + 20z.

- **Non-linear arithmetic**, which allows queries like:

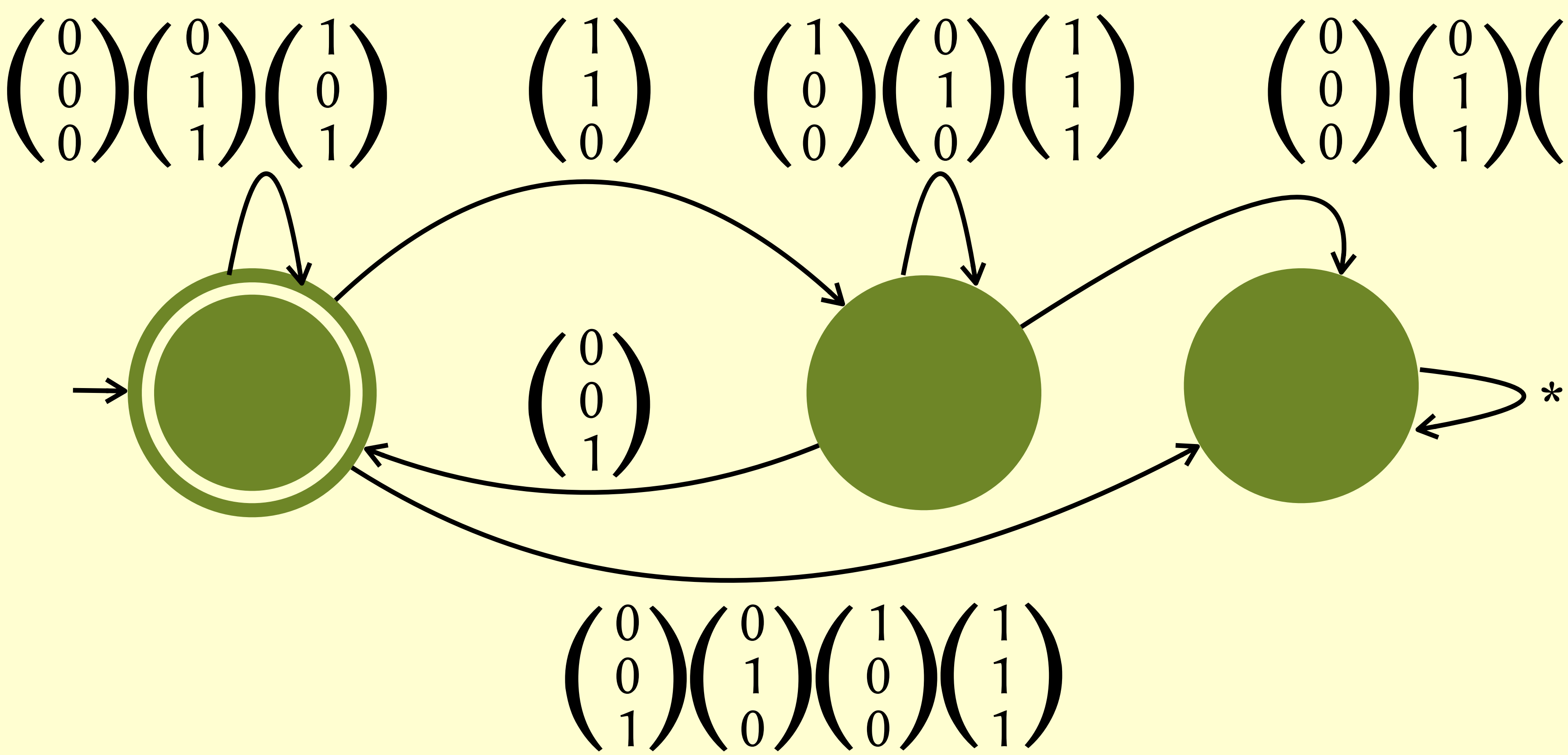
$$(\sin(x)^3 = \cos(\log(y) \cdot x) \vee b \vee -x^2 \geq 2.3y) \wedge \left(\neg b \vee y < -34.4 \vee \exp(x) > \frac{y}{x}\right)$$

- **Theory of arrays**, **theory of bit-vectors**, etc.

Decidability of Presburger

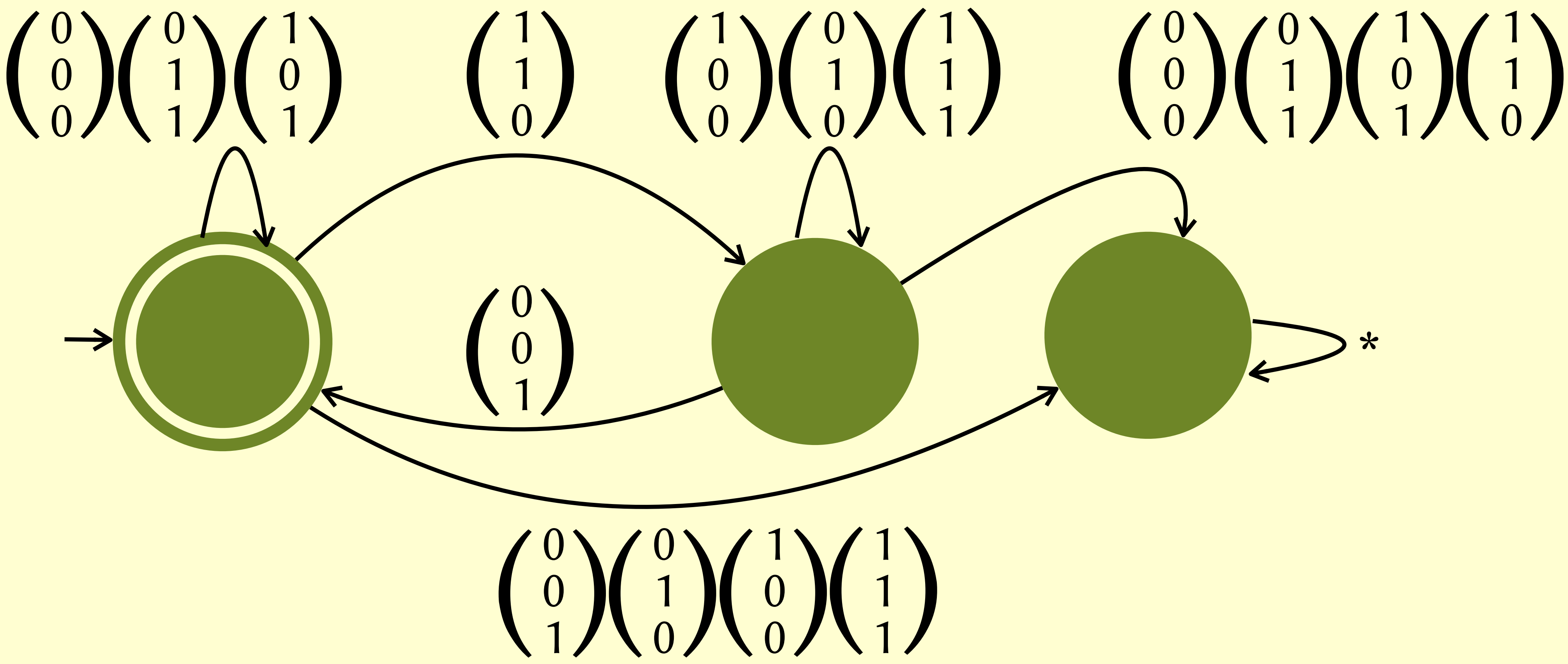
$x + y = z$

	1	2	4	8	16	32	64
x =	0	1	0	0	1	0	0
y =	0	1	0	1	0	1	0
z =	0	0	1	1	1	1	0

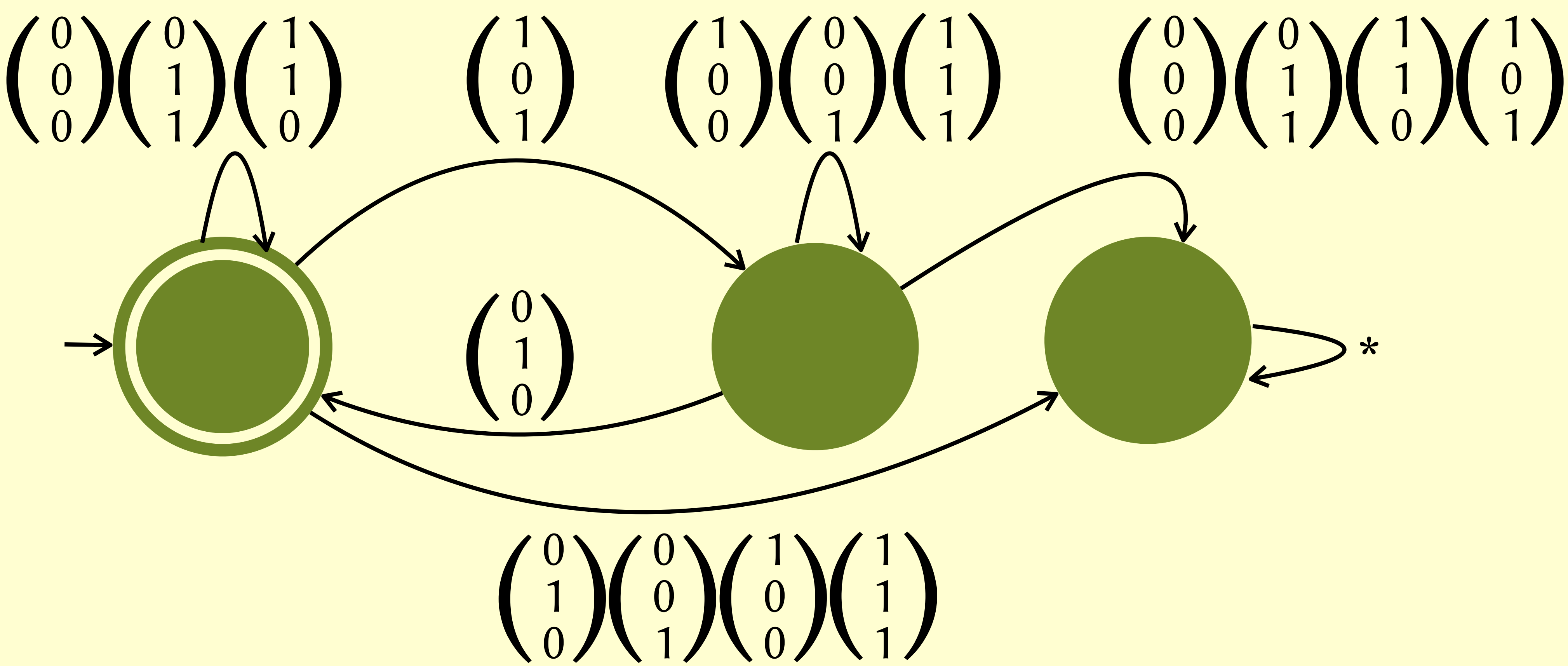


Julius Richard Büchi
1924–1984

x + y = z

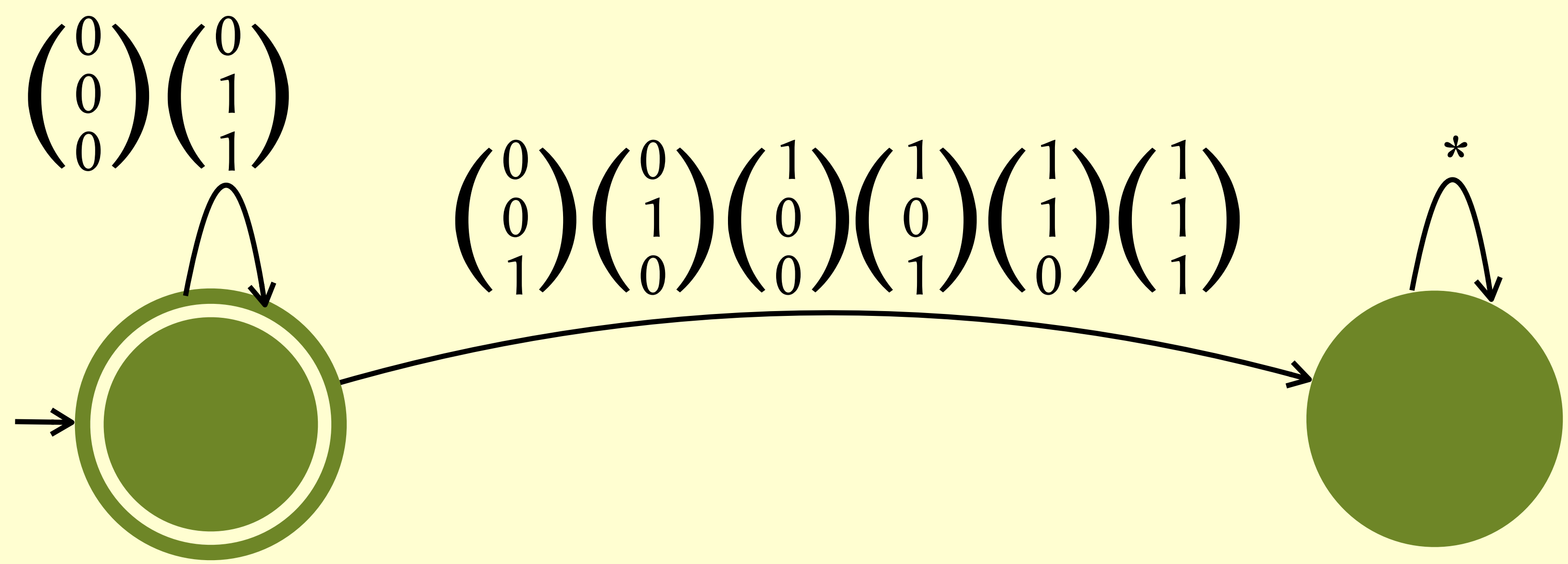


x + z = y



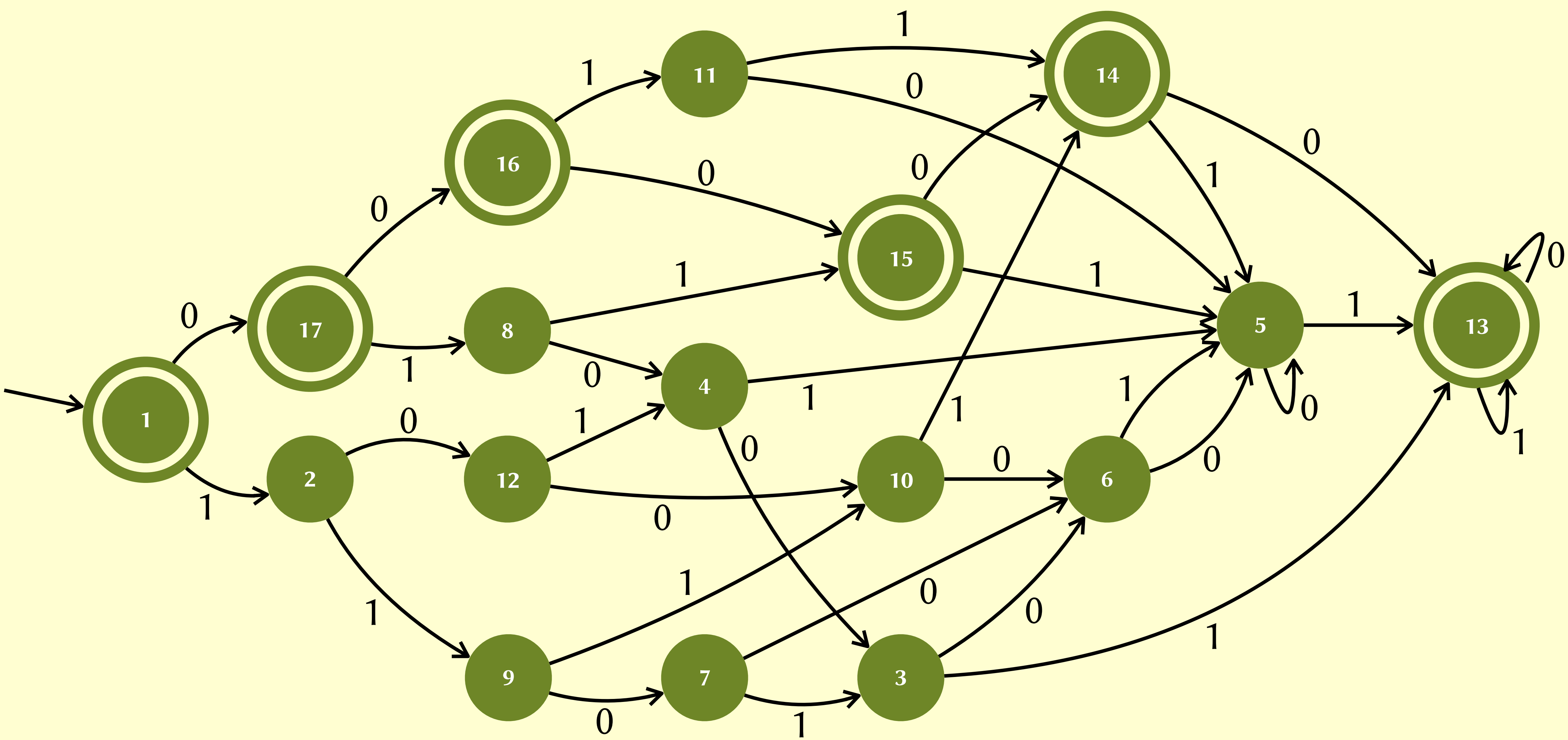
Decidability of Presburger

$$x + y = z$$
$$\wedge\ x + z = y$$



Decidability of Presburger

$\exists x\ y\ z.\ n = 6x + 9y + 20z$



Adding multiplication

- If we add multiplication, we can write a statement representing the Collatz conjecture: does there exist an infinite sequence of positive integers x_0, x_1, x_2, ... such that

$2 \times x_{i+1} = x_i$	if x_i is even
$x_{i+1} = 3 \times x_i + 1$	if x_i is odd

- So **if** arithmetic with multiplication were decidable, we could solve the Collatz conjecture automatically!

Automatic proof

- We often rely on automatic provers:
  - e.g. in Dafny, to show that `invariant P` is preserved,
  - e.g. in Isabelle methods like `by auto`.
- How do these automatic provers work?