[Collatz in Dafny]

Hardware & Software Verification

John Wickerson & Pete Harrod

Lecture 10: SAT and SMT solving

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

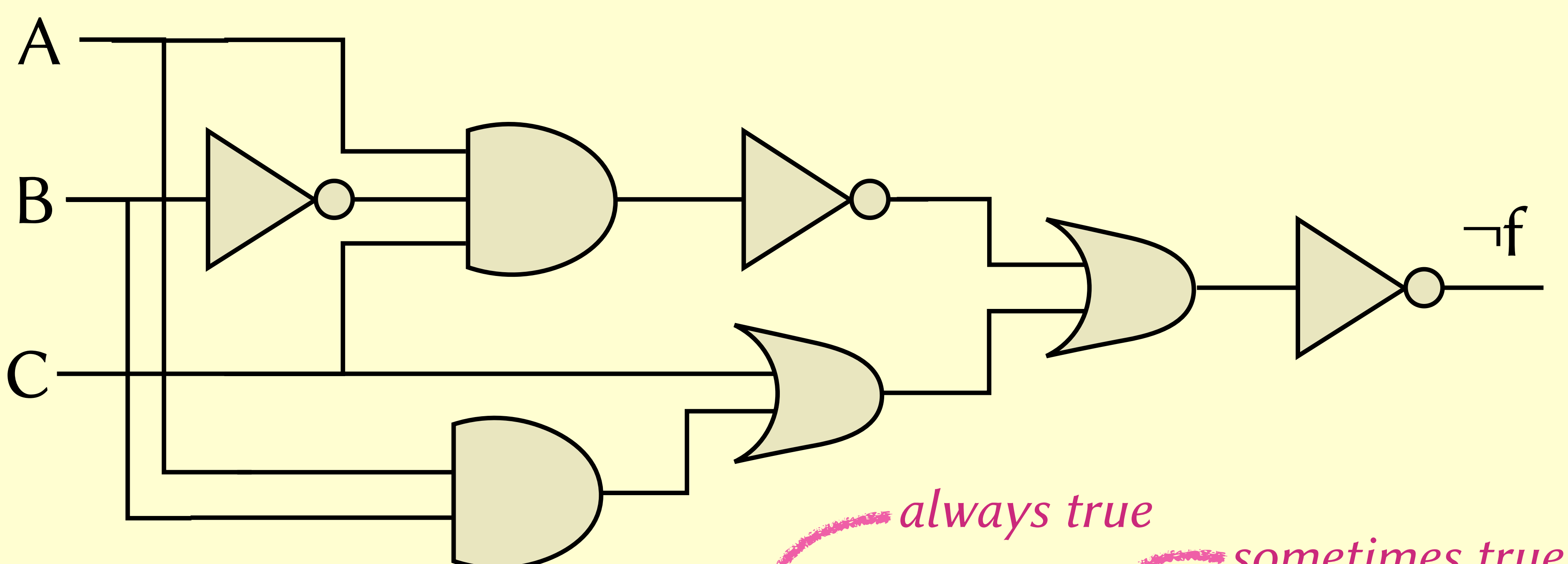
- Simple case: proofs about Boolean statements.

SAT queries

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

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*

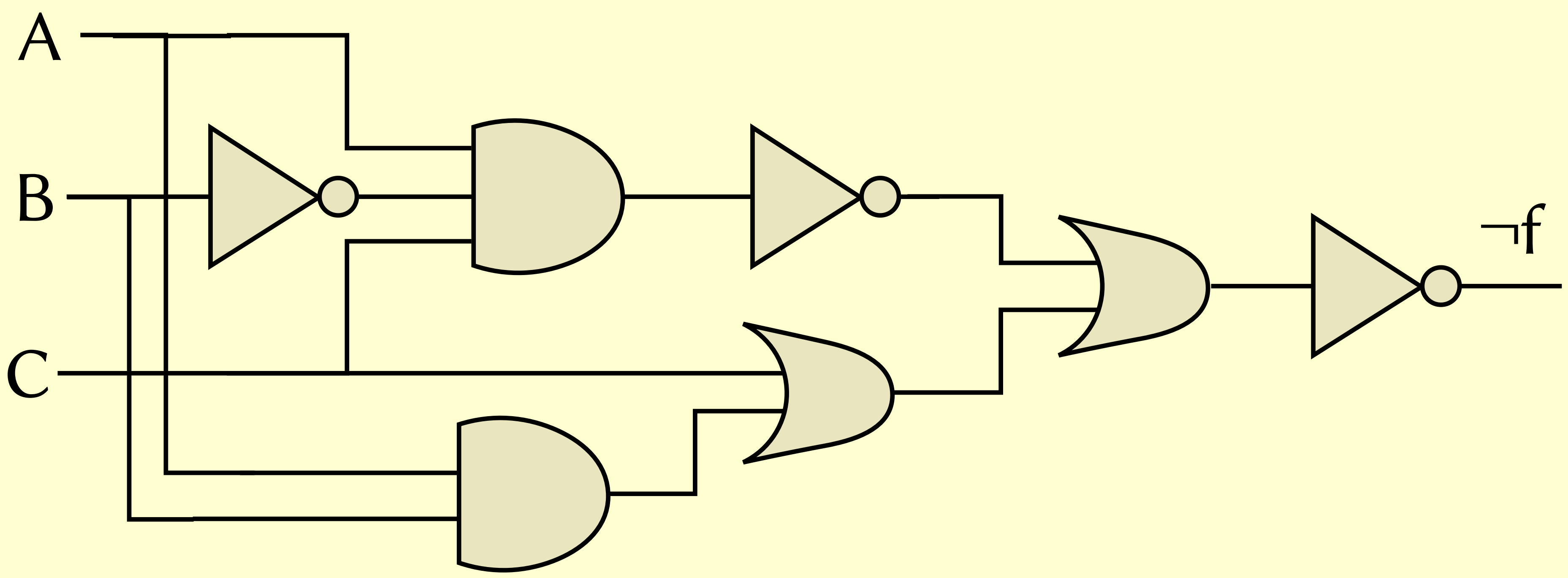
SAT solving

- 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")
```

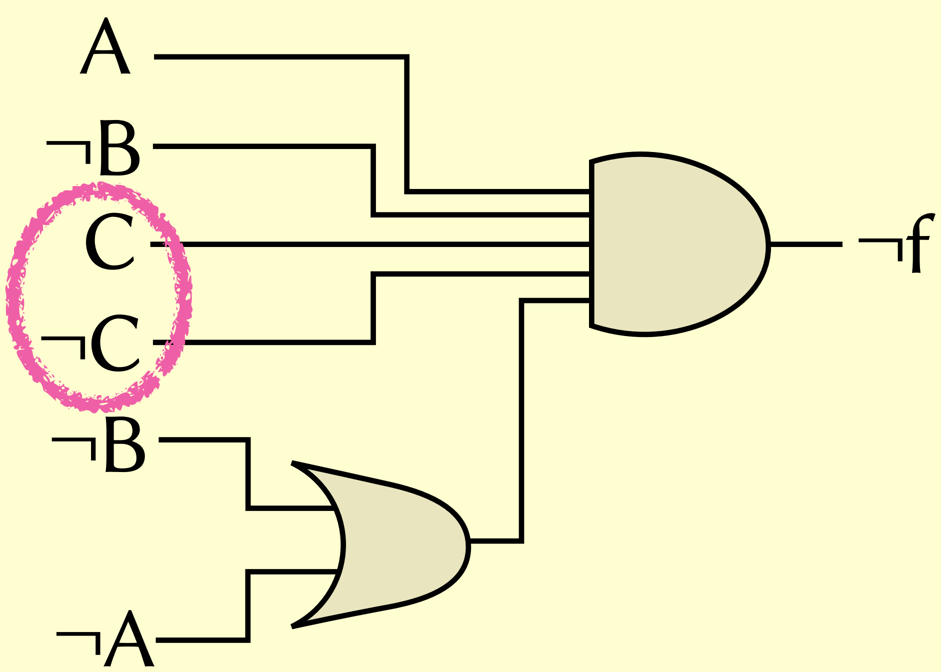
SAT solving

- A cleverer way: use de Morgan's rules to convert the formula to *conjunctive normal form.*

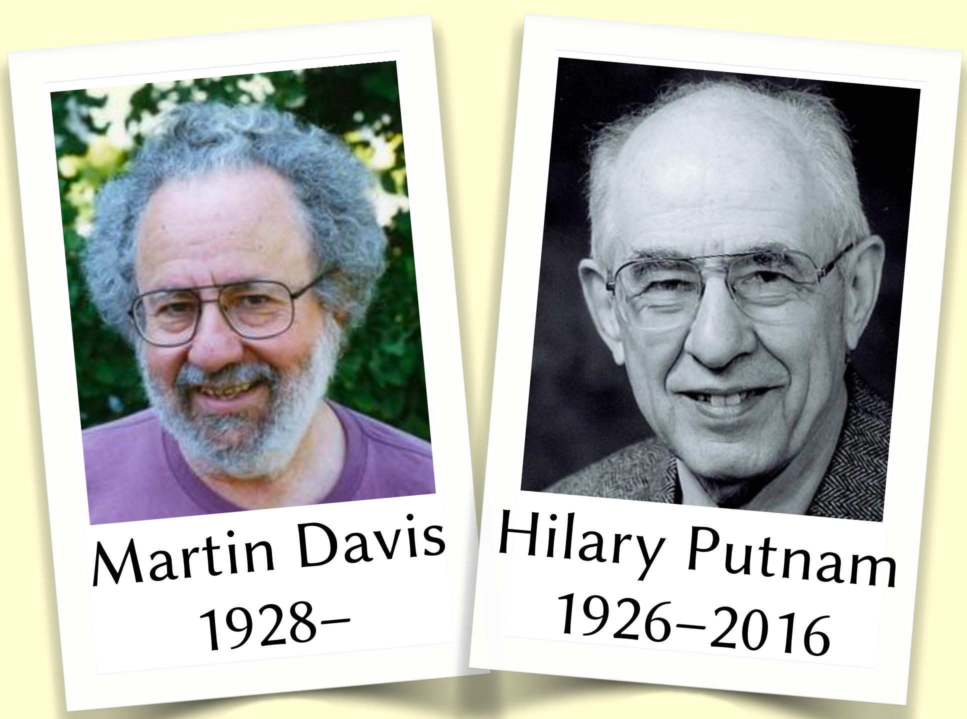


SAT solving

- A cleverer way: use de Morgan's rules to convert the formula to *conjunctive normal form.*

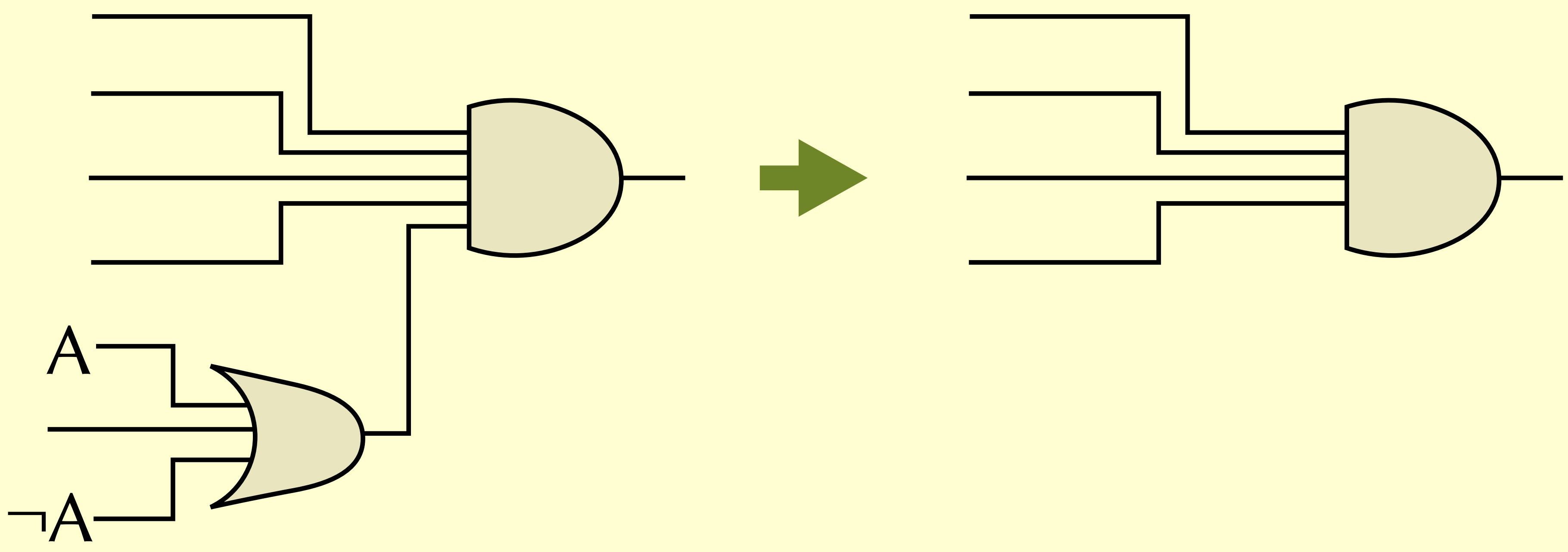


- It may then become obvious that ¬f is UNSAT.
- If not, we can use the Davis–Putnam algorithm...



The DP method

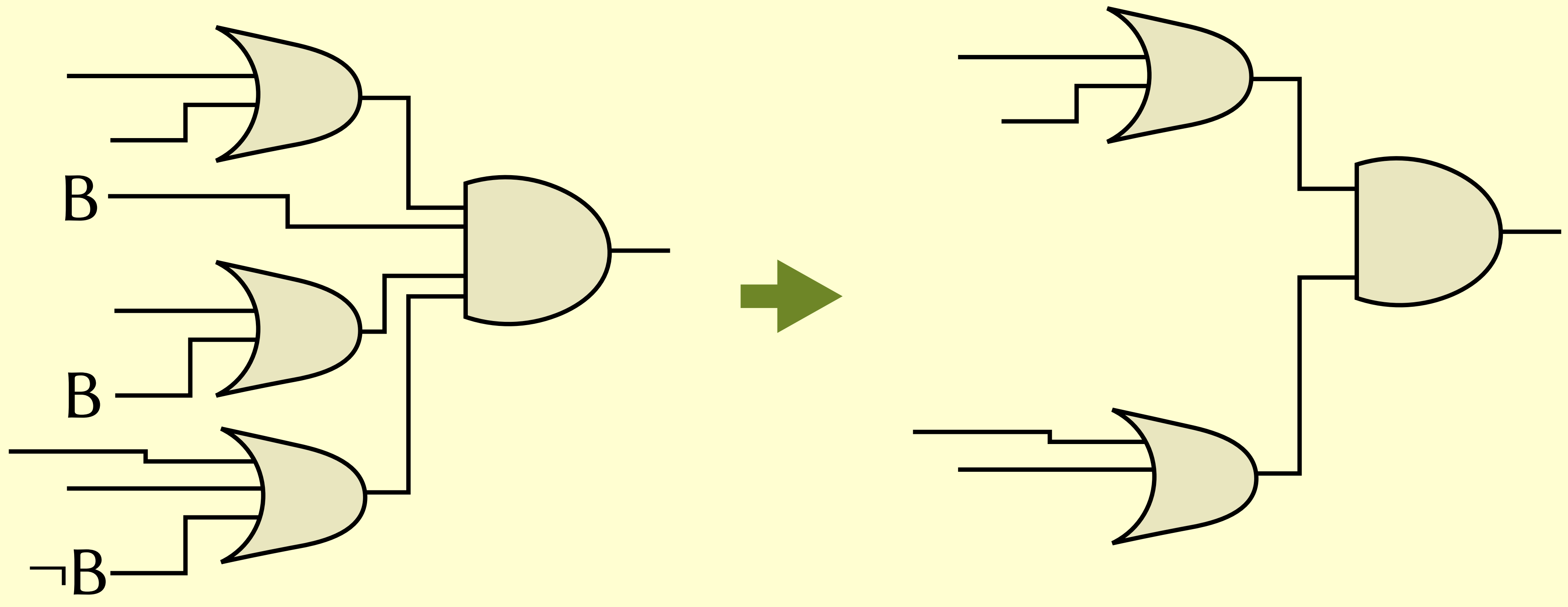
1. If an OR-gate takes both L and ¬L, delete it.



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.)

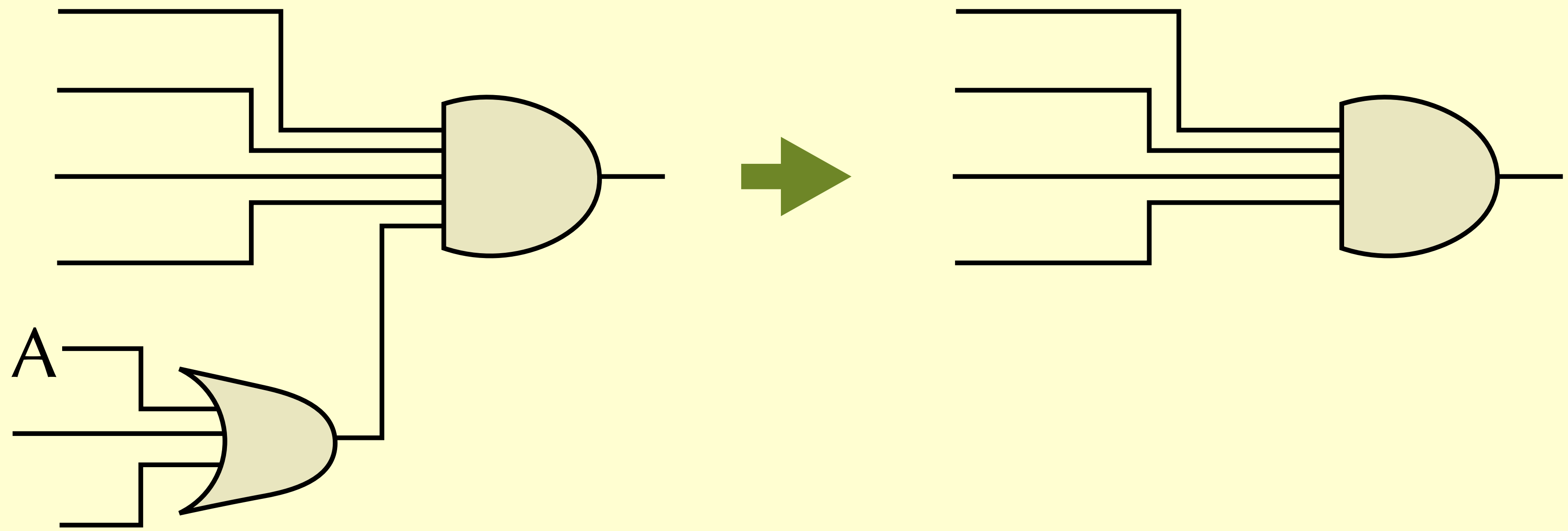


The DP method

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.)

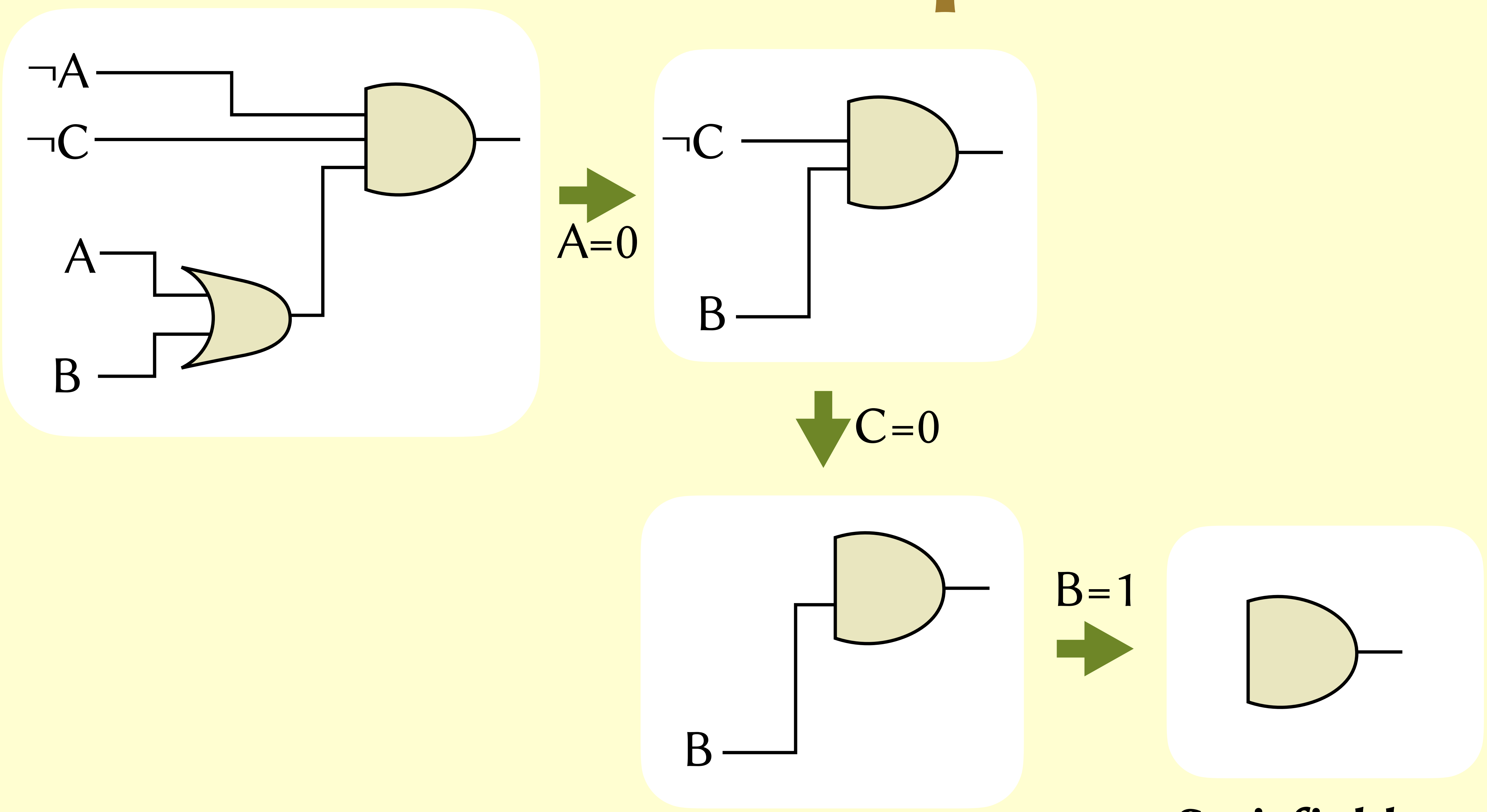
3. If L is unused, delete all OR-gates that take ¬L.
   (The solution, if it exists, will surely involve setting L=0.)

4. If any OR-gate has no inputs, the formula is false.

5. If the AND-gate has no inputs, the formula is true.

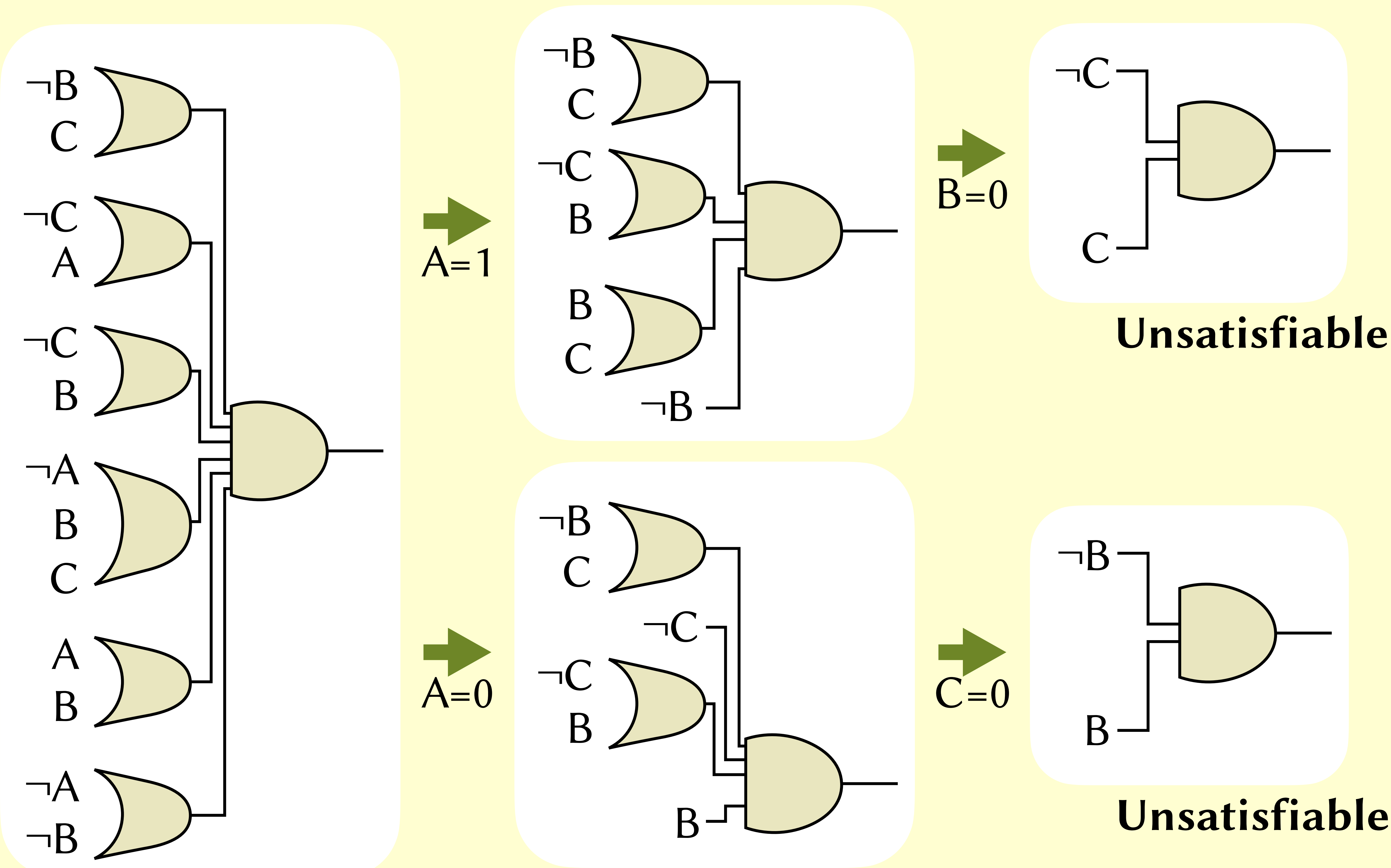
6. Pick a literal L and repeat the above for the cases L=0 and L=1.

DP example 1



Satisfiable, e.g.
when A=0, B=1, C=0

DP example 2

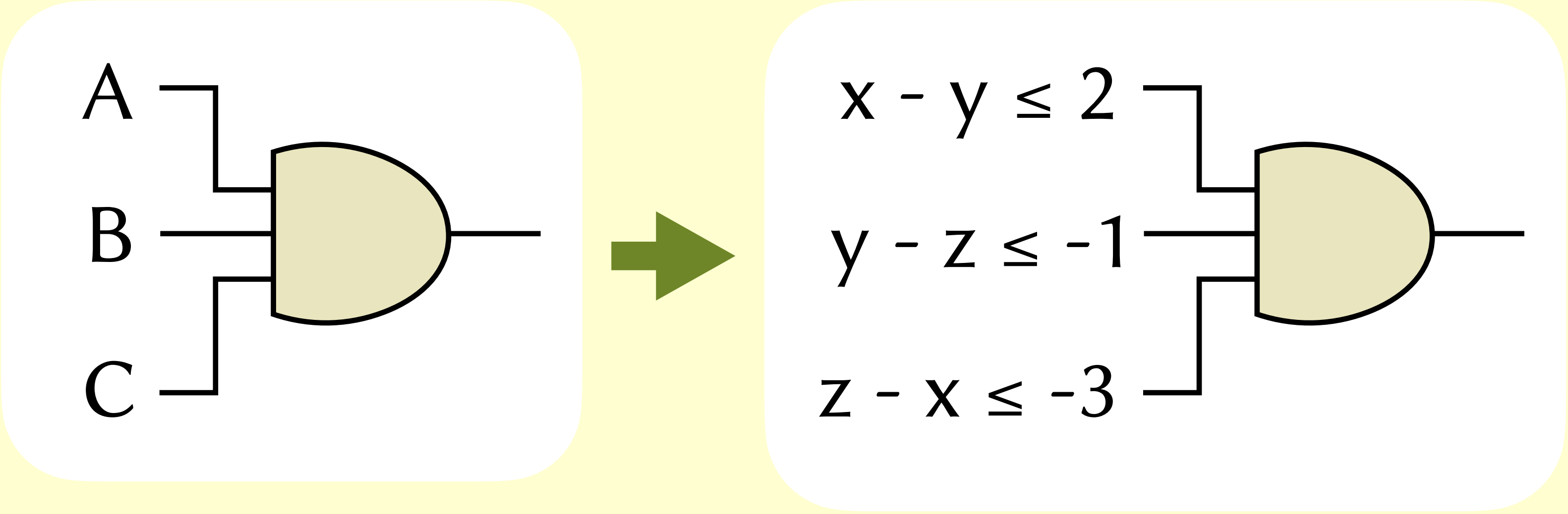

¬B
C
¬C
A
¬C
B
¬A
B
C
A
B
¬A
¬B
A=1
A=0
¬B
C
¬C
B
B
C
¬B
¬B
C
¬C
¬C
B
B
B=0
C=0
¬C
C
Unsatisfiable
¬B
B
Unsatisfiable

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 you can't have x=y and y=z without x=z, and that you can't have x=y without 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 =.

Mojżesz Presburger
1904–c.1943

Some theories

- **Equality and uninterpreted functions**, which knows that you can't have x=y and y=z without x=z, and that you can't have x=y without 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 =.
- **Non-linear arithmetic**, which allows queries like:

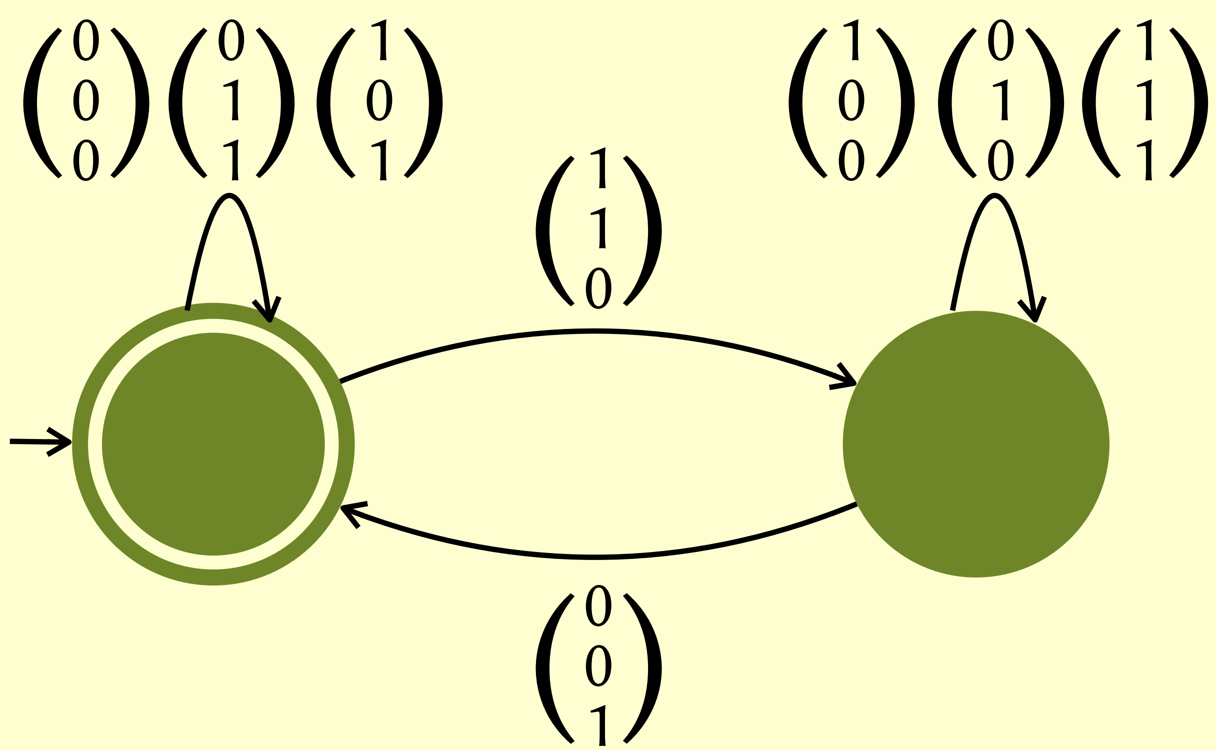
$$(\sin(x)^3 = \cos(\log(y) \cdot x) \vee b \vee -x^2 \geq 2.3y) \wedge \left(\neg b \vee y < -34.4 \vee \exp(x) > \frac{y}{x}\right)$$

- **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

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?