Hardware & Software Verification

John Wickerson

Lecture 3: More Isabelle
23 October 2025

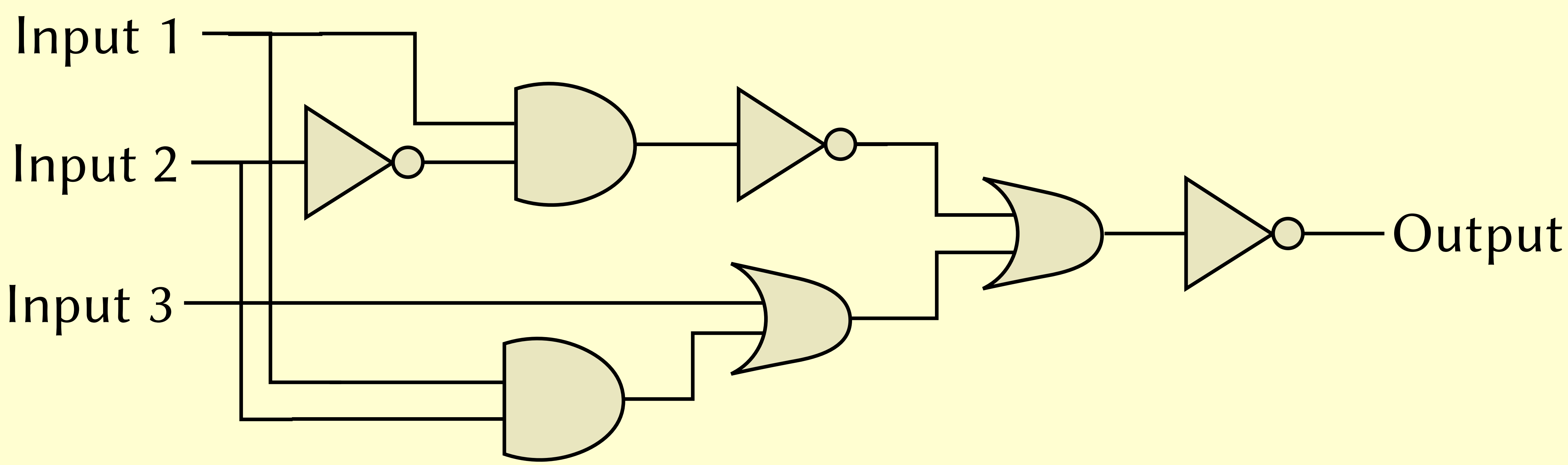
Coursework Admin

- All paired up?
- Coursework deadline is **Friday 12 December 2025** at 15:00.
- Isabelle coursework will be available after today's lecture.
- Dafny and SymbiYosys coursework specification not yet available.

Lecture Outline

- Proving the correctness of a logic synthesiser.

Representing circuits



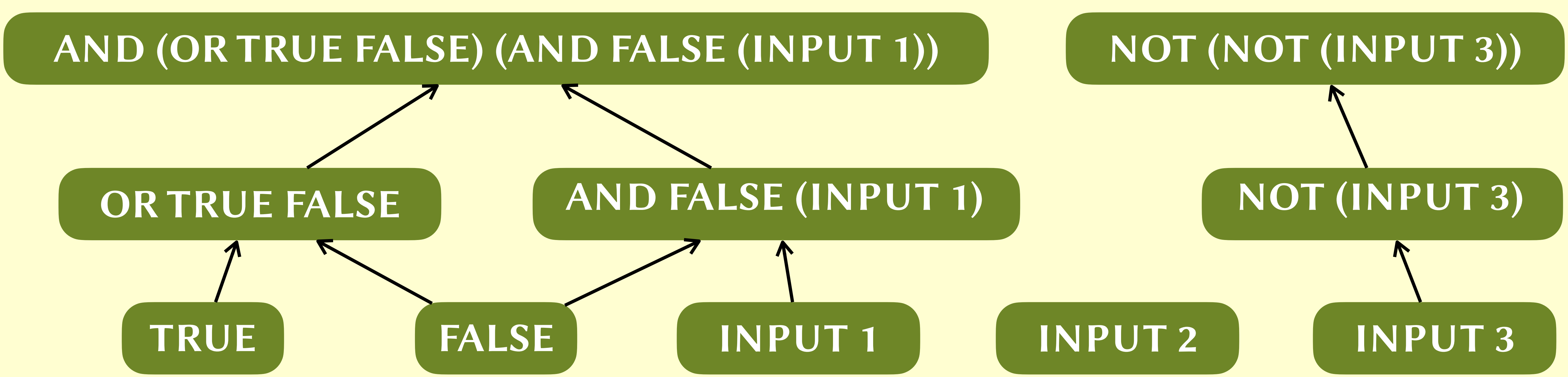

```
NOT
  (OR
    (NOT
      (AND (INPUT 1) (NOT (INPUT 2))))
    (OR
      (INPUT 3)
      (AND (INPUT 1) (INPUT 2))))
```


DEMO

Recursive data structures

```
datatype "circuit" =
  NOT "circuit"
| AND "circuit" "circuit"
| OR "circuit" "circuit"
| TRUE
| FALSE
| INPUT "int"
```

circuit ::= NOT *circuit*
| AND *circuit* *circuit*
| OR *circuit* *circuit*
| TRUE
| FALSE
| INPUT *int*

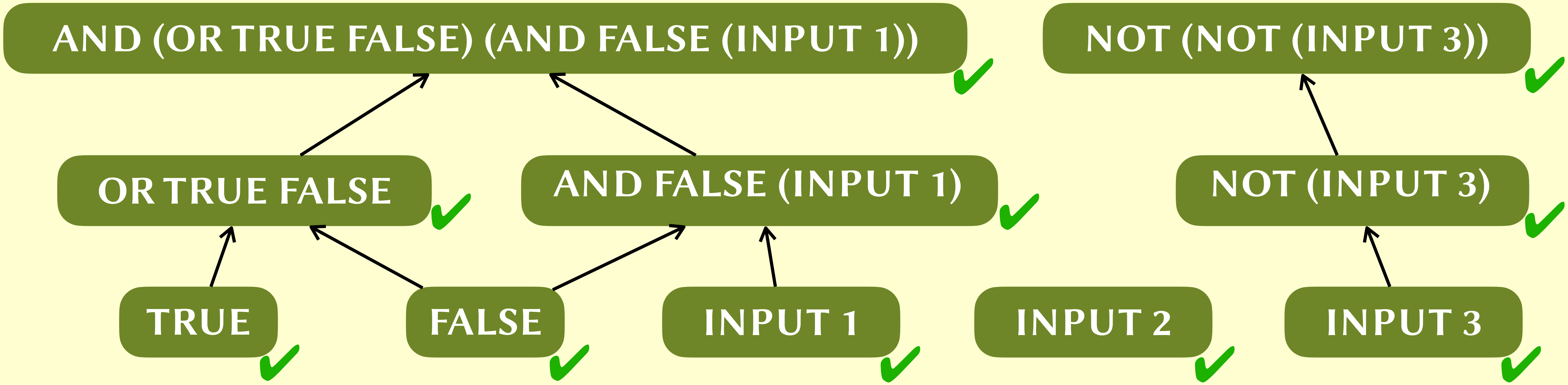



DEMO

Structural induction

- Suppose we want to show that property P holds for all circuits.
- It suffices to show that each constructor preserves P.

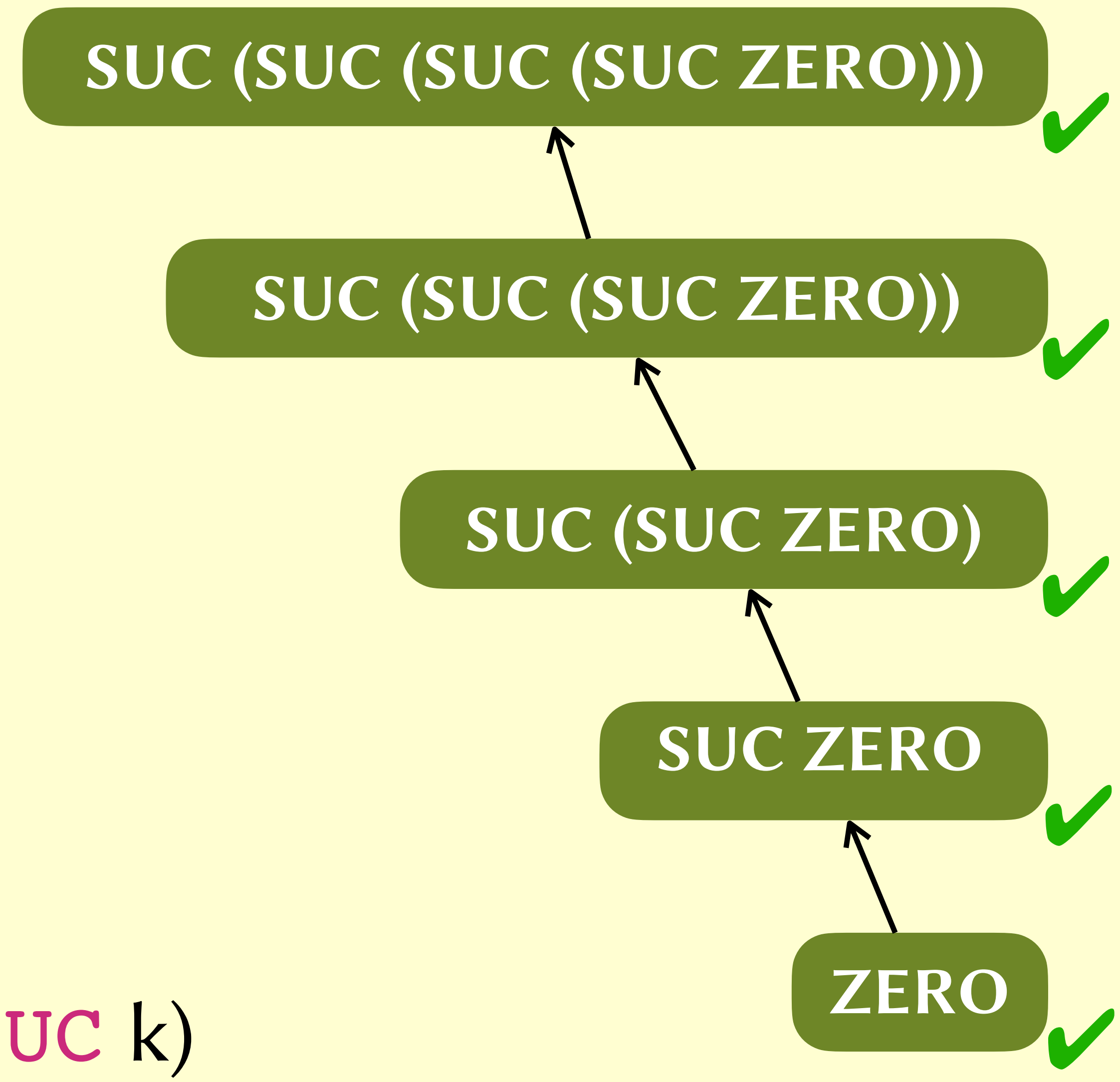
1. $\forall c.\ P(c) \Longrightarrow P(\texttt{NOT}\ c)$
2. $\forall c_1,c_2.\ (P(c_1) \wedge P(c_2)) \Longrightarrow P(\texttt{AND}\ c_1\ c_2)$
3. $\forall c_1,c_2.\ (P(c_1) \wedge P(c_2)) \Longrightarrow P(\texttt{OR}\ c_1\ c_2)$
4. $P(\texttt{TRUE})$
5. $P(\texttt{FALSE})$
6. $\forall i.\ P(\texttt{INPUT}\ i)$



Mathematical induction

```
datatype "nat" =
  ZERO
| SUC "nat"
```

$$nat ::= \texttt{ZERO} \mid \texttt{SUC}\ nat$$



1. $P(\texttt{ZERO})$
2. $\forall k.\ P(k) \Longrightarrow P(\texttt{SUC}\ k)$

Proof by structural induction

Theorem. simulate (mirror c) ρ = simulate c ρ.

Proof by structural induction

Define P(c) = (∀ρ. simulate (mirror c) ρ = simulate c ρ).

Theorem. P(c) holds for all c.

Proof. We proceed by structural induction on c.

Proof by structural induction

Define P(c) = (∀ρ. simulate (mirror c) ρ = simulate c ρ).

Theorem. P(c) holds for all c.

Proof. We proceed by induction on the structure of c.

Case `NOT`. Fix arbitrary c, and assume P(c) as induction hypothesis.

simulate (mirror (`NOT` c)) ρ
= simulate (`NOT` (mirror c)) ρ *[by defn of mirror]*
= ¬ simulate (mirror c) ρ *[by defn of simulate]*
= ¬ simulate c ρ *[by induction hypothesis]*
= simulate (`NOT` c) ρ *[by defn of simulate]*

Thus P(`NOT` c).


DEMO

Rule induction

```
fun f where
  "f (Suc (Suc n)) = f n + f (Suc n)"
| "f (Suc 0) = 1"
| "f 0 = 1"
```

$$\frac{\texttt{f(n)} = A \qquad \texttt{f(Suc n)} = B}{\texttt{f(Suc (Suc n))} = A + B} \qquad \frac{}{\texttt{f(Suc 0)} = 1} \qquad \frac{}{\texttt{f(0)} = 1}$$

$$\frac{P(\texttt{n}) \qquad P(\texttt{Suc n})}{P(\texttt{Suc (Suc n)})} \qquad \frac{}{P(\texttt{Suc 0})} \qquad \frac{}{P(\texttt{0})}$$

Rule induction

```
fun f where
  "f (Suc (Suc n)) = f n + f (Suc n)"
| "f (Suc 0) = 1"
| "f 0 = 1"
```

$$\frac{\texttt{f(n)} = A \qquad \texttt{f(Suc n)} = B}{\texttt{f(Suc (Suc n))} = A + B} \qquad \frac{}{\texttt{f(Suc 0)} = 1} \qquad \frac{}{\texttt{f(0)} = 1}$$

1. $\forall n.\ (P(n) \wedge P(\texttt{Suc}\ n)) \Longrightarrow P(\texttt{Suc}\ (\texttt{Suc}\ n))$
2. $P(\texttt{Suc 0})$
3. $P(\texttt{0})$

Summary

- Recursive data structures
- Recursive functions
- Structural induction
- Rule induction