Hardware & Software Verification

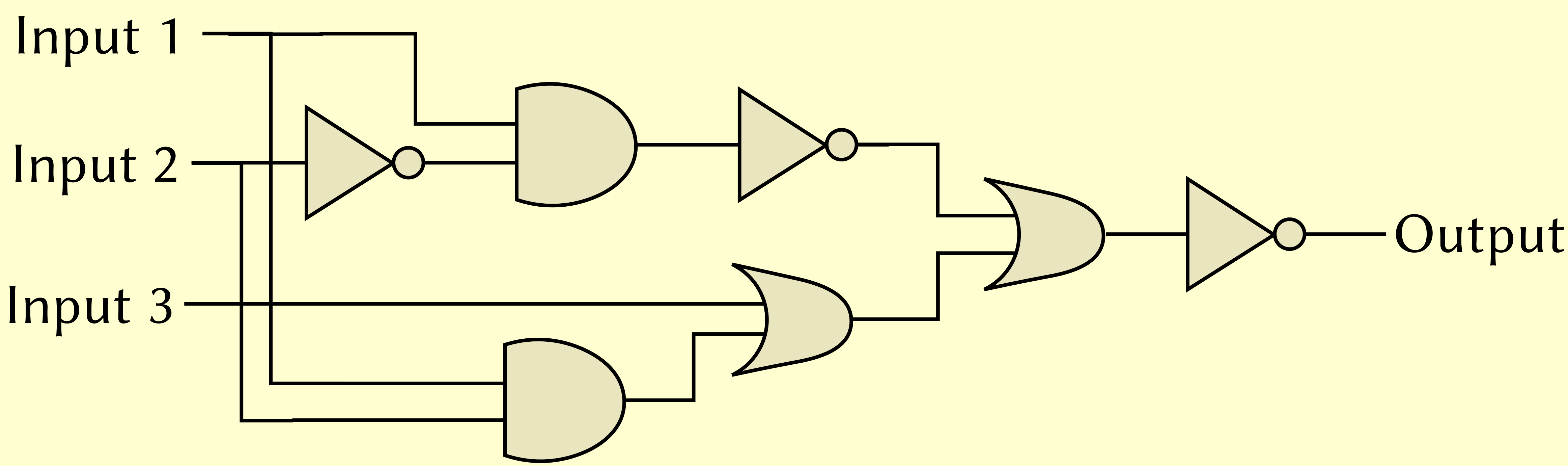
John Wickerson

Lecture 3: More Isabelle

Lecture Outline

- Proving the correctness of a logic synthesiser.

Representing circuits



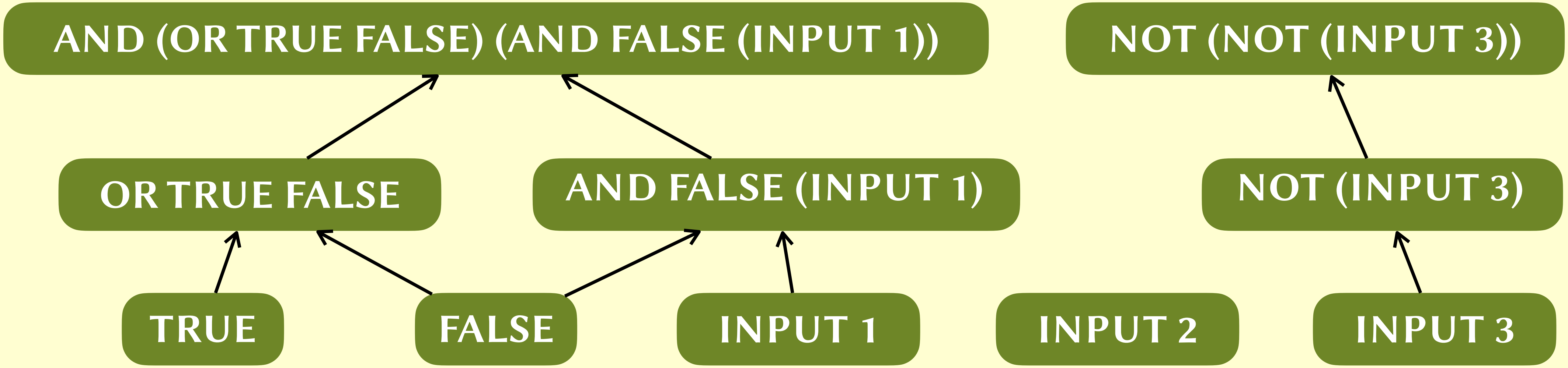

```
NOT
  (OR
    (NOT
      (AND (INPUT 1) (NOT (INPUT 2))))
    (OR
      (INPUT 3)
      (AND (INPUT 1) (INPUT 2))))
```


DEMO

Recursive data structures

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

$$\begin{aligned} \mathit{circuit} ::= {} & \texttt{NOT}\ \mathit{circuit} \\ \mid {} & \texttt{AND}\ \mathit{circuit}\ \mathit{circuit} \\ \mid {} & \texttt{OR}\ \mathit{circuit}\ \mathit{circuit} \\ \mid {} & \texttt{TRUE} \\ \mid {} & \texttt{FALSE} \\ \mid {} & \texttt{INPUT}\ \mathit{int} \end{aligned}$$



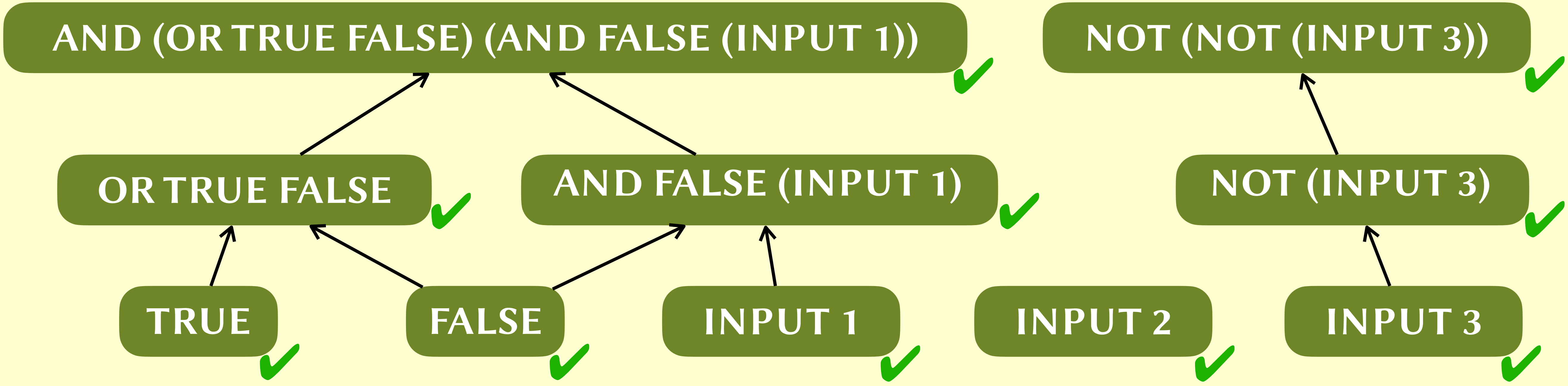

DEMO

Structural induction

- Suppose we want to show that property P holds for all circuits.

- It suffices to show that each constructor preserves P.

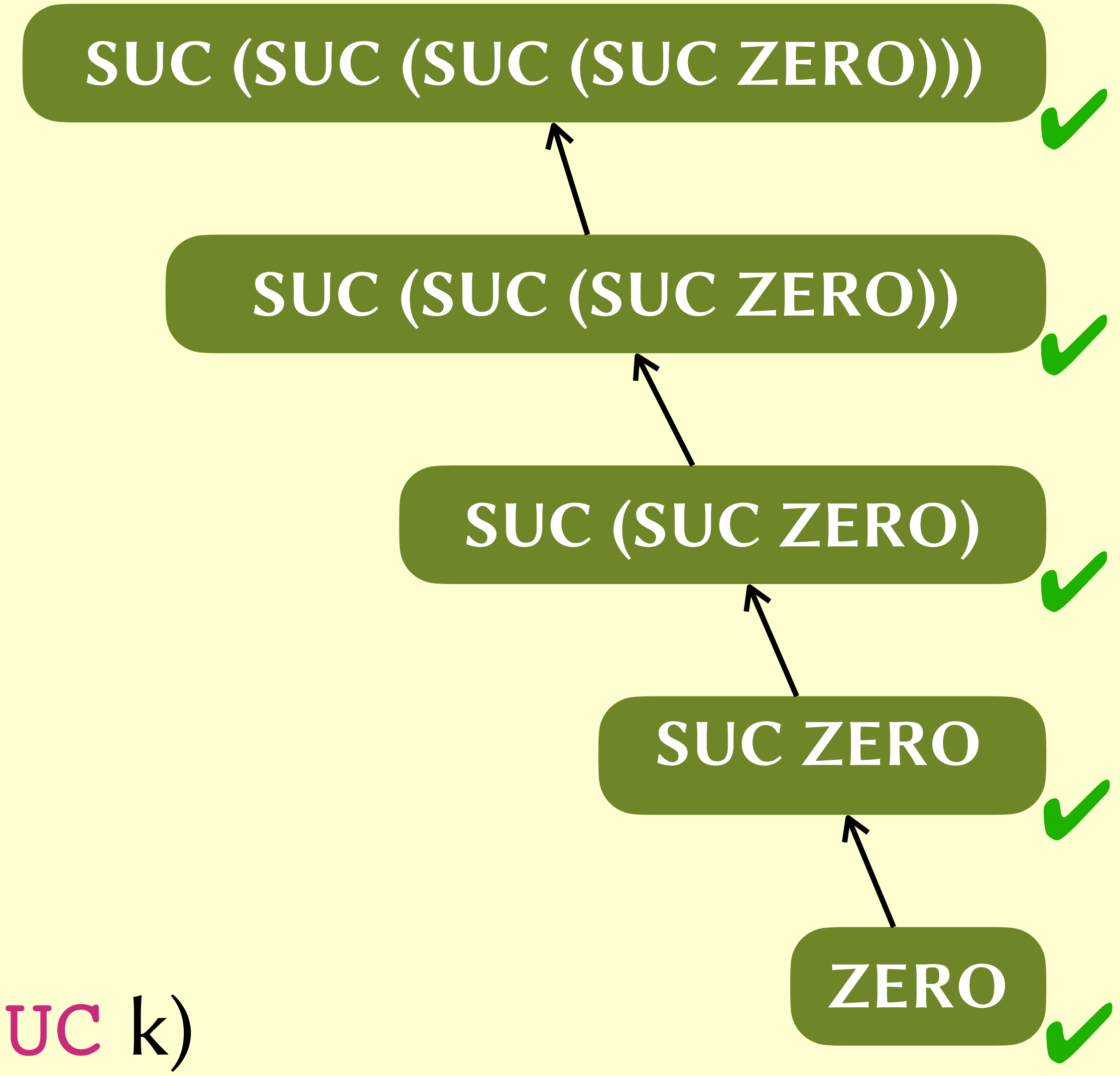
1. $\forall c.\ P(c) \Longrightarrow P(\texttt{NOT}\ c)$
2. $\forall c_1,c_2.\ (P(c_1) \wedge P(c_2)) \Longrightarrow P(\texttt{AND}\ c_1\ c_2)$
3. $\forall c_1,c_2.\ (P(c_1) \wedge P(c_2)) \Longrightarrow P(\texttt{OR}\ c_1\ c_2)$
4. $P(\texttt{TRUE})$
5. $P(\texttt{FALSE})$
6. $\forall i.\ P(\texttt{INPUT}\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



Mathematical induction

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

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



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.** We proceed by induction on the structure of c.
  - *Case "NOT":* Fix arbitrary k and assume
    simulate (mirror k) ρ = simulate k ρ
    as our induction hypothesis. We must prove that
    simulate (mirror (NOT k)) ρ = simulate (NOT k) ρ
    which we do as follows:

simulate (mirror (NOT k)) ρ	
= simulate (NOT (mirror k)) ρ	[by definition of mirror]
= ¬ simulate (mirror k) ρ	[by definition of simulate]
= ¬ simulate k ρ	[using induction hypothesis]
= simulate (NOT k) ρ	[by definition of simulate]


DEMO

Rule induction

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

- **Theorem.** $f(n) \geq n$.

- **Proof.** Define $P(n) = (f(n) \geq n \wedge f(n) \geq 1)$.
  Rule induction here requires us to prove:

  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