

# Correctness

**David Basin**

Department of Computer Science  
ETH Zurich

# Correctness

- Correctness is important! What does your program compute?
- What does this mean? What properties should hold?
  - ▶ **Termination:** Important for many, but not all, programs. Examples?
  - ▶ **Functional behavior:** function should return “correct” value. Can be defined by another (mathematically defined) function or an input-output relation.
- Correctness is often not obvious. It must be proven!

## Correctness (example)

- The factorial function can be written as

```
fac 0 = 1
fac n = n * fac (n-1)
```

or as

```
fac2 (0,a) = a
fac2 (n,a) = fac2 (n-1,n*a)
```

- Do these functions terminate?

```
fac 3  $\rightsquigarrow$  3 * fac 2  $\rightsquigarrow$  3 * 2 * fac 1  $\rightsquigarrow$  3 * 2 * 1 * 1  $\rightsquigarrow$  6
```

```
fac2 (3, 1)  $\rightsquigarrow$  fac2 (2, 3 * 1)  $\rightsquigarrow$  fac2 (1, 2 * 3 * 1)  $\rightsquigarrow$   
fac2 (0, 1 * 2 * 3 * 1)  $\rightsquigarrow$  1 * 2 * 3 * 1  $\rightsquigarrow$  6
```

- Are `fac  $n$`  and `fac2  $n$  1` equivalent?

# Termination

- If  $f$  is defined in terms of functions  $g_1, \dots, g_k$  ( $g_i \neq f$ ), and each  $g_i$  terminates, then so does  $f$ .

```
g x = x * x + 15
f x = (g x + x + 2) / 13
```

- Problem is recursion, e.g., `fac n`, for  $n > 0$
- Sufficient condition for termination: Arguments are smaller along a well-founded order on function's domain.
  - ▶ An order  $>$  on a set  $S$  is **well-founded** iff there is no infinite decreasing chain  $x_1 > x_2 > x_3 > \dots$ , for  $x_i \in S$ .
  - ▶ Example:  $>_{\mathcal{N}}$ . Counter examples:  $>_{\mathcal{Z}}$ ,  $>_{\mathcal{R}}$ .  
Here we write  $>_S$  to denote the set  $S$ , where  $>_S \subseteq S \times S$

## An aside on well-founded relations

- How do we construct new relations from old?
- Let  $R_1$  and  $R_2$  be binary relations on a set  $S$ . The composition of  $R_1$  and  $R_2$  is defined as

$$R_2 \circ R_1 \equiv \{(a, c) \in S \times S \mid \exists b \in S. (a, b) \in R_1 \wedge (b, c) \in R_2\}$$

- Let  $R \subseteq S \times S$ . Define:

$$R^1 \equiv R$$

$$R^{n+1} \equiv R \circ R^n \text{ for } n \geq 1$$

$$R^+ \equiv \bigcup_{n \geq 1} R^n$$

## Well-founded relations (cont.)

- **Lemma:** Let  $R \subseteq S \times S$ . Let  $s_0, s_i \in S$  and  $i \geq 1$ . Then  $s_0 R^i s_i$  iff there are  $s_1, \dots, s_{i-1} \in S$  such that  $s_0 R s_1 R \dots R s_{i-1} R s_i$ .

► Proof: induction on  $i$  (easy exercise).

- **Theorem:** If  $>$  is a well-founded order on the set  $S$ , then  $>^+$  is also well-founded on  $S$ .

**Proof.** Assume that

$$a_1 >^+ a_2 >^+ a_3 >^+ \dots$$

is an infinite descending chain. Then, there exist  $i_j \geq 1$  such that  $a_j >^{i_j} a_{j+1}$ , for all  $j \geq 1$ . By the above lemma, the sequence

$$a_1 >^{i_1} a_2 >^{i_2} a_3 >^{i_3} \dots$$

contradicts the well-foundedness of  $>$ .

## Termination: examples

$$\text{fac } 0 = 1$$

$$\text{fac } n = n * \text{fac } (n-1)$$

$$\text{fac2 } (0, a) = a$$

$$\text{fac2 } (n, a) = \text{fac2 } (n-1, n*a)$$

- Factorial function `fac`

- ▶ `fac  $n$`  has only `fac ( $n - 1$ )` as recursive call and  $n > n - 1$ .
- ▶ Here  $>$  is the standard ordering over the natural numbers.

- Function `fac2`

- ▶ `fac2 ( $n, a$ )` has only `fac2 ( $n - 1, n * a$ )` as a recursive call.
- ▶ The first argument is always smaller under  $>$ .

- Do `fac` and `fac2` terminate when  $n < 0$ ?

## Termination (cont.)

Do the following functions terminate?

$$f(0) = 0$$

$$f(1) = 1$$

$$f(n) = f(n-1) + f(n-2)$$

$$g(0) = 1$$

$$g(1) = 1$$

$$g(n) = g(n+1) + g(n-2)$$

$$h(0, y) = y$$

$$h(x, y) = h(x-1, y+1)$$

What do they actually compute?



## Correctness — behavior

- Do `fac` and `fac2` compute the same function in following sense?

$$\forall n \in \text{Nat}. \text{fac } n = \text{fac2 } (n, 1)$$

- Testing sufficient to find errors

$$\text{fac } 0 = 1 = \text{fac2 } (0, 1)$$

$$\text{fac } 1 = 1 = \text{fac2 } (1, 1)$$

$$\text{fac } 2 = 2 = \text{fac2 } (2, 1)$$

- Correctness requires a formal proof!

## Correctness — equational reasoning

- Proofs based on simple idea: **functions are equations**

```
swap :: (Int, Int) -> (Int, Int)
swap (a, b) = (b, a)
```

**Meaning:** For all possible values of  $a$  and  $b$ :

$$\text{swap } (a, b) = (b, a)$$

More formally:

$$\forall a \in \text{Int}. \forall b \in \text{Int}. \text{swap}(a, b) = (b, a)$$

- Some properties can be shown through equational reasoning.

E.g.,  $\text{swap } (\text{swap } (a, b)) = (a, b)$ . Proof?

- More generally: proofs in first-order logic with equality.

## Correctness — reasoning by cases

```
maxi :: Int -> Int -> Int
maxi n m
  | n >= m    = n
  | otherwise = m
```

Can we prove that  $\text{maxi } n \ m \geq n$ ?

**We have**  $n \geq m \vee \neg(n \geq m)$

**Now show**  $\text{maxi } n \ m \geq n$  for both cases

**Case 1:**  $n \geq m$ ,

then  $\text{maxi } n \ m = n$  and  $n \geq n$

**Case 2:**  $\neg(n \geq m)$ ,

then  $\text{maxi } n \ m = m$ . But  $m > n$ , so  $\text{maxi } n \ m \geq n$

## Correctness (cont.)

The previous proof used two inference rules:

**Excluded Middle:** For all propositions  $P$ :

$$P \vee \neg P$$

**Case split:** Given  $Q \vee R$ ,

to prove any  $P$ , we must prove:

1.  $P$  follows from  $Q$  and
2.  $P$  follows from  $R$

Compare with ND-rules ( $\vee$ -elimination)

## Proof by induction

- Suppose you wish to prove a formula  $P(n)$ , for all  $n \in \text{Nat}$

$$\forall n \in \text{Nat}. 0 + 1 + 2 + \dots + n = n \cdot (n + 1) / 2$$

- Proof by cases not possible here:  $P(0), P(1), P(2), P(3), \dots$
- Domino principle: prove  $P(0)$  and, for every  $n \in \text{Nat}$ ,  $P(n + 1)$  follows from  $P(n)$
- **Induction proof rule:** To prove  $\forall n \in \text{Nat}. P(n)$ 
  - Base case:** Prove  $P(0)$
  - Step case:** For an arbitrary  $n$ , prove  $P(n + 1)$  under the assumption  $P(n)$ .
- Exercise: formulate rule ND style and prove example in FOL.

# Induction as the dual of recursion

$$\begin{array}{ll} \text{fac } 0 & = 1 \\ \text{fac } n & = n * \text{fac } (n-1) \end{array}$$

1. Base case, like in induction

2. Step case

- States how to compute  $\text{fac } n$ , given value for  $\text{fac } (n - 1)$ .
- Corresponds to proving  $P(n)$  follows from  $P(n - 1)$ , for  $n > 0$  or equivalently: proving  $P(n + 1)$  follows from  $P(n)$ , for  $n \geq 0$

3. Correspondence can be precisely formulated

Induction is not more complicated or more circular than (terminating) recursive definitions!

## Example: powers

```
power2 :: Int -> Int      -- computes 2^r as 2 * 2 * ... * 1
power2 0 = 1
power2 r = 2 * power2 (r-1)
```

```
sumPowers :: Int -> Int   -- computes 1 + 2 + 4 + ... + 2^r
sumPowers 0 = 1
sumPowers r = sumPowers (r-1) + power2 r
```

Examples:

$$\text{sumPowers } 3 = 15$$

$$\text{power2 } 4 = 2^4 = 16$$

$$\text{sumPowers } 5 = 63$$

$$\text{power2 } 6 = 2^6 = 64$$

How are these functions related?

# Powers

- **Conjecture:**  $\forall n \in \text{Nat}. (\text{sumPowers } n) + 1 = \text{power2 } (n + 1)$
- **Proof:** Let  $P(n) \equiv (\text{sumPowers } n) + 1 = \text{power2 } (n + 1)$ .  
We show for  $\forall n. P(n)$  by induction on  $n$ .

**Base case:** Show  $P(0)$

$$(\text{sumPowers } 0) + 1 = 1 + 1 = 2$$

and

$$\text{power2 } (0 + 1) = 2 \cdot \text{power2 } 0 = 2 \cdot 1 = 2$$

**Step case:** Assume  $P(n)$  for an arbitrary  $n$ , i.e.,

$$(\text{sumPowers } n) + 1 = \text{power2 } (n + 1)$$

and prove  $P(n + 1)$

$$(\text{sumPowers } (n + 1)) + 1 = \text{power2 } (n + 2)$$



## Powers proof (cont.)

$$\begin{aligned}(\text{sumPowers } (n + 1)) + 1 &= \text{sumPowers } ((n + 1) - 1) + \text{power2 } (n + 1) + 1 \text{ (def.)} \\&= \text{sumPowers } (n) + 1 + \text{power2 } (n + 1) \text{ (arithmetic)} \\&= \text{power2 } (n + 1) + \text{power2 } (n + 1) \text{ (ind. hypothesis)} \\&= 2 \cdot \text{power2 } (n + 1) \text{ (arithmetic)} \\&= \text{power2 } (n + 2) \text{ (def.)}\end{aligned}$$

- We have proven  $(\text{sumPowers } n) + 1 = \text{power2 } (n + 1)$

Equivalently:  $\text{sumPowers } n = (\text{power2 } (n + 1)) - 1$

- Equation above yields a better definition of *sumPowers*:
  - ▶ *power2*:  $O(n)$  iterations
  - ▶ *sumPowers* =  $O(n)$  iterations + with each iteration a call to *power2*. Overall:  $O(n^2)$  calls of *power2*.

# Noetherian induction



- Induction schema

**Induction:** To prove  $P(n)$ , for all natural numbers  $n$ .

**To prove:**  $P(n)$  for an arbitrary  $n$  under the assumption that  $P(m)$  holds, for all  $m < n$

- In general, we can use any well-founded ordering  $<$ .

Here, the transitive closure of the predecessor relation on  $Nat$

- Same principle applies to any set, not just  $Nat$ .

# Deriving 1-step (structural, weak) induction from Noetherian induction

To derive 1-step induction we must derive  $P(n)$  for an arbitrary  $n \in \text{Nat}$ , from the assumptions

$$P(0) \tag{1}$$

$$\forall n \in \text{Nat}. P(n) \rightarrow P(n+1) \tag{2}$$

By Noetherian induction, we may assume that

$$\forall m \in \text{Nat}. (m < n) \rightarrow P(m) \tag{3}$$

We establish  $P(n)$  by a case split on  $n = 0 \vee n > 0$ .

**Case 1:**  $n = 0$ , and  $P(0)$  follows from (1).

**Case 2:**  $n > 0$ , so  $n = k + 1$  for some  $k \in \text{Nat}$ , i.e., we must prove  $P(k + 1)$ .

By (3),  $k < k + 1 \rightarrow P(k)$ . Hence  $P(k)$  holds and  $P(k + 1)$  follows from (2).