Lecture 4
Proving correctness
Last time

- Specification format for a function \( F \)
  - **type**
  - **assumption** (REQUIRES)
  - **guarantee** (ENSURES)

For all (properly typed) \( x \) satisfying the **assumption**, \( F \ x \) satisfies the **guarantee**
Remember...

• Can use equivalence (a.k.a. equality, written \( = \)) to specify applicative behavior of functions

• Equality is compositional

• Equality is defined in terms of evaluation

• \( \implies^* \) is consistent with \( = \) and ML evaluation
**Example**

```plaintext
fun f(x:int):int = if x=0 then 1 else f(x-1)
```

\[ f : \text{int} \rightarrow \text{int} \]

REQUIRES  \( x \geq 0 \)

ENSURES  \( f \ x = 1 \)

\[ f \ x = 1 \]

means the same as

\[ f \ x \ \Rightarrow^* \ 1 \]
fun eval ([ ] : int list) : int = 0
| eval (d::L) = d + 10 * (eval L)

eval : int list -> int

REQUIRES
L is a list of decimal digits

ENSURES
(eval L) ⟷* a non-negative integer
fun eval ([ ]:int list) : int = 0
| eval (d::L) = d + 10 * (eval L)

fun eval : int list -> int

REQUIRES
L is a list of decimal digits

ENSURES
(eval L) ⟷⁺ a non-negative integer
fun eval ([ ]:int list) : int = 0
| eval (d::L) = d + 10 * (eval L)

fun eval : int list -> int

REQUIRES
  L is a list of decimal digits

ENSURES
  (eval L) \mapsto^* \text{a non-negative integer}
fun eval ([ ]: int list) : int = 0
| eval (d::L) = d + 10 * (eval L)

eval : int list -> int

REQUIRES
L is a list of decimal digits

ENSURES
(eval L) ⟷* a non-negative integer
fun decimal (n:int) : int list = 
  if n<10 then [n]
  else (n mod 10) :: decimal (n div 10)

REQUIRES n >= 0
ENSURES
decimal n =
a list L of decimal digits
such that (eval L) = n
fun decimal (n:int) : int list = 
  if n<10 then [n] 
  else (n mod 10) :: decimal (n div 10)

This REQUIRES property is just right, for the given ENSURES
Problem

How to show that a spec for a recursive function is valid

• Solution: Use induction to prove it
  • we offer templates to help with accuracy
• We focus on examples...

  program structure guides proof

But first, what’s a proof?
What is a proof?

A proof is a logical sequence of steps, leading to a conclusion.

- Each step must follow logically from math facts, or the results of earlier steps.
Simple induction

- To prove a property of the form
  \[ \forall n \geq 0. P(n) \]

- First, prove \( P(0) \).

- Then show that, for \( k \geq 0 \),
  \( P(k+1) \) follows logically from \( P(k) \).
Why this works

• P(0) gets a direct proof \textit{base}

• P(0) implies P(1) \textit{step (with }k=0\text{)}

• P(1) implies P(2) \textit{step (with }k=1\text{)}

• …

For each \(n \geq 0\) we can establish \(P(n)\)

\text{(follows from } base \text{ after } n \text{ uses of } step\text{)}
Example

fun f(x:int):int = if x=0 then 1 else f(x-1)

REQUIRES n ≥ 0
ENSURES f(n) = 1

• To prove:

For all n:int such that n≥0, f(n) = 1

type REQUIRES ENSURES
proof (part 1)

fun f(x:int):int = if x=0 then 1 else f(x-1)

Let P(n) be "f(n) = 1"

Theorem: ∀n ≥ 0. P(n)

Proof: By simple induction on n.

• **Base**: we prove P(0). Here's a proof:
Let \( P(n) \) be “\( f(n) = 1 \)”

Theorem: \( \forall n \geq 0. \ P(n) \)

Proof: By simple induction on \( n \).

- **Base**: we prove \( P(0) \). Here’s a proof:
  
  \[
  f(0)
  \]
proof (part 1)

fun f(x:int):int = if x=0 then 1 else f(x-1)

Let P(n) be “f(n) = 1”

Theorem: ∀n≥0. P(n)

Proof: By simple induction on n.

• **Base**: we prove P(0). Here’s a proof:

\[
\text{f 0 = (fn x => if x=0 then 1 else f(x-1)) 0}
\]
proof (part 1)

fun f(x:int):int = if x=0 then 1 else f(x-1)

Let P(n) be “f(n) = 1”

Theorem: ∀n≥0. P(n)

Proof: By simple induction on n.

• **Base**: we prove P(0). Here’s a proof:

  f 0 = (fn x => if x=0 then 1 else f(x-1)) 0
  = if 0=0 then 1 else f(0-1)
proof (part 1)

fun \( f(x:\text{int}):\text{int} = \text{if } x=0 \text{ then } 1 \text{ else } f(x-1) \)

Let \( P(n) \) be “\( f(n) = 1 \)”

Theorem: \( \forall n \geq 0. \ P(n) \)

Proof: By simple induction on \( n \).

- **Base**: we prove \( P(0) \). Here’s a proof:

\[
\begin{align*}
f 0 &= (\text{fn } x => \text{if } x=0 \text{ then } 1 \text{ else } f(x-1)) \ 0 \\
&= \text{if } 0=0 \text{ then } 1 \text{ else } f(0-1) \\
&= \text{if true then } 1 \text{ else } f(0-1)
\end{align*}
\]
proof (part 1)

fun f(x:int):int = if x=0 then 1 else f(x-1)

Let P(n) be “f(n) = 1”

Theorem: ∀n ≥ 0. P(n)

Proof: By simple induction on n.

• Base: we prove P(0). Here’s a proof:

\[
f 0 = (fn x => if x=0 then 1 else f(x-1)) 0 \\
= if 0=0 then 1 else f(0-1) \\
= if true then 1 else f(0-1) \\
= 1
\]
proof (part 1)

fun f(x:int):int = if x=0 then 1 else f(x-1)

Let P(n) be “f(n) = 1”

Theorem: ∀n≥0. P(n)

Proof: By simple induction on n.

• **Base**: we prove P(0). Here’s a proof:

  \[
  f\ 0 \ = \ (fn \ x => \ if \ x=0 \ then \ 1 \ else \ f(x-1))\ 0 \\
  \ = \ if \ 0=0 \ then \ 1 \ else \ f(0-1) \\
  \ = \ if \ true \ then \ 1 \ else \ f(0-1) \\
  \ = \ 1 \\
  \]

  So f(0) = 1. That’s P(0).
proof (part 1)

Let \( P(n) \) be “\( f(n) = 1 \)”

Theorem: \( \forall n \geq 0. P(n) \)

Proof: By simple induction on \( n \).

- **Base**: we prove \( P(0) \). Here’s a proof:

\[
\begin{align*}
f(0) &= \text{(fn } x \Rightarrow \text{ if } x=0 \text{ then } 1 \text{ else } f(x-1)) \ 0 \\
&= \text{if } 0=0 \text{ then } 1 \text{ else } f(0-1) \\
&= \text{if true then } 1 \text{ else } f(0-1) \\
&= 1
\end{align*}
\]

So \( f(0) = 1 \). That’s \( P(0) \).
proof (part 2)

fun f(x:int):int = if x=0 then 1 else f(x-1)

• Inductive step:
  Let \( k \geq 0 \) and assume \( P(k) \), \( f(k) = 1 \).
  We prove \( P(k+1) \), \( f(k+1) = 1 \).

• Let \( v \) be the value of \( k+1 \), so \( v = k+1 \).
  \( f(k+1) \)
proof (part 2)

fun f(x:int):int = if x=0 then 1 else f(x-1)

• **Inductive step:**
  Let \( k \geq 0 \) and assume \( P(k) \), \( f(k) = 1 \).
  We prove \( P(k+1) \), \( f(k+1) = 1 \).

• Let \( v \) be the value of \( k+1 \), so \( v = k+1 \).
  \[
f(k+1) = (\text{fn } x \Rightarrow \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))(k+1)
  \]
proof (part 2)

fun f(x:int):int = if x=0 then 1 else f(x-1)

• Inductive step:
  Let $k \geq 0$ and assume $P(k)$, $f(k) = 1$.
  We prove $P(k+1)$, $f(k+1) = 1$.

• Let $v$ be the value of $k+1$, so $v = k+1$.
  
  $f(k+1) = (fn x => if x=0 then 1 else f(x-1))(k+1)$
  = $(fn x => if x=0 then 1 else f(x-1))(v)$
proof (part 2)

fun f(x:int):int = if x=0 then 1 else f(x-1)

• Inductive step:
Let k ≥ 0 and assume P(k), f k = 1.
We prove P(k+1), f(k+1) = 1.

• Let v be the value of k+1, so v = k+1.
  f(k+1) = (fn x => if x=0 then 1 else f(x-1))(k+1)
  = (fn x => if x=0 then 1 else f(x-1))(v)
  = if v=0 then 1 else f(v-1)
proof (part 2)

fun f(x:int):int = if x=0 then 1 else f(x-1)

- **Inductive step:**
  Let \(k \geq 0\) and assume \(P(k), \quad f(k) = 1\).
  We prove \(P(k+1), \quad f(k+1) = 1\).

- Let \(v\) be the value of \(k+1\), so \(v = k+1\).
  \[
  f(k+1) = \left( \text{fn } x \Rightarrow \text{ if } x=0 \text{ then } 1 \text{ else } f(x-1) \right)(k+1)
  = \left( \text{fn } x \Rightarrow \text{ if } x=0 \text{ then } 1 \text{ else } f(x-1) \right)(v)
  = \text{ if } v=0 \text{ then } 1 \text{ else } f(v-1)
  = \text{ if false then } 1 \text{ else } f(v-1)
  \]
proof (part 2)

\[
\text{fun } f(x:\text{int}):\text{int} = \text{if } x=0 \text{ then } 1 \text{ else } f(x-1)
\]

- **Inductive step:**
  Let \(k \geq 0\) and assume \(P(k), \quad f(k) = 1\).
  We prove \(P(k+1), \quad f(k+1) = 1\).

- Let \(v\) be the value of \(k+1\), so \(v = k+1\).
  \[
  f(k+1) = (\text{fn } x => \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))(k+1)
  = (\text{fn } x => \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))(v)
  = \text{if } v=0 \text{ then } 1 \text{ else } f(v-1)
  = \text{if } \text{false} \text{ then } 1 \text{ else } f(v-1)
  = f(v-1)
  \]
proof (part 2)

\[ \text{fun } f(x: \text{int}): \text{int} = \text{if } x = 0 \text{ then } 1 \text{ else } f(x-1) \]

- **Inductive step:**
  Let \( k \geq 0 \) and assume \( P(k), \quad f(k) = 1 \).
  We prove \( P(k+1), \quad f(k+1) = 1 \).

- Let \( v \) be the value of \( k+1 \), so \( v = k+1 \).
  \[ f(k+1) = (\text{fn } x => \text{if } x = 0 \text{ then } 1 \text{ else } f(x-1))(k+1) \]
  \[ = (\text{fn } x => \text{if } x = 0 \text{ then } 1 \text{ else } f(x-1))(v) \]
  \[ = \text{if } v = 0 \text{ then } 1 \text{ else } f(v-1) \]
  \[ = \text{if false then } 1 \text{ else } f(v-1) \]
  \[ = f(v-1) \]
  \[ = f(k) \]
proof (part 2)

fun f(x:int):int = if x=0 then 1 else f(x-1)

• Inductive step:
  Let k \geq 0 and assume P(k), \quad f(k) = 1.
  We prove P(k+1), \quad f(k+1) = 1.

• Let v be the value of k+1, so v = k+1.
  \[
  f(k+1) = (\text{fn } x \Rightarrow \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))(k+1)
  = (\text{fn } x \Rightarrow \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))(v)
  = \text{if } v=0 \text{ then } 1 \text{ else } f(v-1)
  = \text{if false then } 1 \text{ else } f(v-1)
  = f(v-1)
  = f(k) \quad \text{since } v=k+1
proof (part 2)

fun f(x:int):int = if x=0 then 1 else f(x-1)

• Inductive step:
  Let \( k \geq 0 \) and assume \( P(k), \quad f(k) = 1 \).
  We prove \( P(k+1), \quad f(k+1) = 1 \).

• Let \( v \) be the value of \( k+1 \), so \( v = k+1 \).
  \[
  f(k+1) = (\text{fn } x => \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))(k+1) \\
  = (\text{fn } x => \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))(v) \\
  = \text{if } v=0 \text{ then } 1 \text{ else } f(v-1) \\
  = \text{if } \text{false} \text{ then } 1 \text{ else } f(v-1) \\
  = f(v-1) \\
  = f(k) \quad \text{since } v=k+1 \\
  = 1 \quad \text{by assumption } P(k)
  \]
proof (part 2)

fun f(x:int):int = if x=0 then 1 else f(x-1)

• Inductive step:
  Let \( k \geq 0 \) and assume \( P(k) \), \( f(k) = 1 \).
  We prove \( P(k+1) \), \( f(k+1) = 1 \).

• Let \( v \) be the value of \( k+1 \), so \( v = k+1 \).
  \[
  f(k+1) = (\text{fn } x => \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))(k+1)
  = (\text{fn } x => \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))(v)
  = \text{if } v=0 \text{ then } 1 \text{ else } f(v-1)
  = \text{if false then } 1 \text{ else } f(v-1)
  = f(v-1)
  = f(k) \quad \text{since } v = k+1
  = 1 \quad \text{by assumption } P(k)
  
  So \( P(k+1) \) holds.
Notes

• State the *induction hypothesis* clearly

• Use induction hypothesis only when *justified*

• Use equations and rules only when *justified*

• Use math and logic accurately

• Give explanation for non-trivial steps
Warning

• It’s easy to write *bogus* proofs

• We want you to learn how to write *excellent* proofs

• Here are some bad examples, not to be copied…
Is this a proof of $f^0 = 1$?
Is this a proof of $f^0 = 1$?
Is this a proof of $f \ 0 = 1$?

$$f \ 0 = 1$$

$$(\text{fn } x \Rightarrow \text{if } x=0 \text{ then } 1 \text{ else } f(x-1)) \ 0 = 1$$
Is this a proof of \( f \, 0 = 1 \)?

\[
\begin{align*}
  f \, 0 &= 1 \\
  (\text{fn} \ x \Rightarrow \text{if } x=0 \text{ then } 1 \text{ else } f(x-1)) \, 0 &= 1 \\
  \text{if } 0=0 \text{ then } 1 \text{ else } f(0-1) &= 1
\end{align*}
\]
Is this a proof of \( f \, 0 = 1 \)?

\[
\begin{align*}
f \, 0 &= 1 \\
(f \, x \Rightarrow \text{if } x=0 \text{ then } 1 \text{ else } f(x-1)) \, 0 &= 1 \\
\text{if } 0=0 \text{ then } 1 \text{ else } f(0-1) &= 1 \\
\text{if true then } 1 \text{ else } f(0-1) &= 1
\end{align*}
\]
Is this a proof of \( f(0) = 1 \)?

\[
\begin{align*}
  f(0) &= 1 \\
  (\text{fn } x \Rightarrow \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))\ 0 &= 1 \\
  \text{if } 0=0 \text{ then } 1 \text{ else } f(0-1) &= 1 \\
  \text{if } \text{true} \text{ then } 1 \text{ else } f(0-1) &= 1 \\
  1 &= 1
\end{align*}
\]
Is this a proof of $f(0) = 1$?

\[
\text{(fn } x \Rightarrow \text{ if } x=0 \text{ then } 1 \text{ else } f(x-1)) \ 0 = 1
\]

\[
\text{if } 0=0 \text{ then } 1 \text{ else } f(0-1) = 1
\]

\[
\text{if true then } 1 \text{ else } f(0-1) = 1
\]

\[
1 = 1
\]

true
Is this a proof of $f \, 0 = 1$?

$$f \, 0 = 1$$

$$(fn \, x \Rightarrow \text{if } x=0 \text{ then } 1 \text{ else } f(x-1)) \, 0 = 1$$

$$(\text{if } 0=0 \text{ then } 1 \text{ else } f(0-1)) = 1$$

$$(\text{if true then } 1 \text{ else } f(0-1)) = 1$$

$$1 = 1$$

true
Is this a proof of $f \ 0 = 1$?

$$f \ 0 = 1$$

$$(fn \ x \Rightarrow \ if \ x=0 \ then \ 1 \ else \ f(x-1)) \ 0 = 1$$

$$\ if \ 0=0 \ then \ 1 \ else \ f(0-1) = 1$$

$$\ if \ true \ then \ 1 \ else \ f(0-1) = 1$$

$$1 = 1$$

No, this just shows that

“if $f \ 0 = 1$ then true is true”
Is this a proof of $f\ 0 = 1$?

\[
f\ 0 = 1
\]

\[
(fn\ x => \text{if } x=0 \text{ then } 1 \text{ else } f(x-1))\ 0 = 1
\]

\[
\text{if } 0=0 \text{ then } 1 \text{ else } f(0-1) = 1
\]

\[
\text{if } \text{true then } 1 \text{ else } f(0-1) = 1
\]

\[
1 = 1
\]

No, this just shows that

“if $f\ 0 = 1$ then true is true”

The first line in this “proof” isn’t (yet) a math fact!
is this a proof?

2 = 1
1 = 2
2+1 = 1+2
3=3
true

by symmetry
by adding
by arithmetic

Is this a proof that 2 = 1?
Is this a proof?

\[
\begin{align*}
2 &= 1 \\
1 &= 2 & \text{by symmetry} \\
2+1 &= 1+2 & \text{by adding} \\
3 &= 3 & \text{by arithmetic} \\
\text{true} & \\
\end{align*}
\]

Is this a proof that \(2 = 1\)?
Every non-empty set of dinosaurs has the same color.

**Theorem**

All dinosaurs are the same color!

Base case: any one dinosaur is the same color as itself.

**Induction step:**

Assume that any group of \( n \) dinosaurs is the same color. Consider a group of \( n+1 \) dinosaurs. The first \( n \) (dino 1 to n) are all the same color.

And the LAST \( n \) (dino 2 to \( n+1 \)) must all be the same color! So all \( n+1 \) are the same color.

And by induction, all dinosaurs are the same color!

Holy cow! Math is BROKEN.

No, WAIT! There is a lesson here!

Hey guys! All dinosaurs are the SAME COLOR!
The proof is wrong

- The *inductive step* is inaccurate

\[ P(2) \text{ does not follow from } P(1) \]
Is this a proof?

\[
\begin{align*}
\text{fun} & \quad \text{silly}(x: \text{int}): \text{int} = \text{silly}(x) \\
\text{fun} & \quad \text{hitchhiker}(n: \text{int}): \text{int} = 42
\end{align*}
\]

Claim

For all values \( x : \text{int} \), \( \text{hitchhiker}(\text{silly } x) = 42 \).

Proof

\[
\begin{align*}
\text{hitchhiker}(\text{silly } x) & = (\text{fn } n \Rightarrow 42) (\text{silly } x) \\
& = [(\text{silly } x)/n] 42 \\
& = 42 \quad \text{... QED}
\end{align*}
\]
Is this a proof?

```
fun silly(x:int):int = silly(x)
fun hitchhiker(n:int):int = 42
```

Claim

For all values x : int, hitchhiker(silly x) = 42.

Proof

```
hitchhiker(silly x) = (fn n => 42) (silly x)
= [(silly x)/n] 42
= 42
```

No! The substitution step isn’t justified, because (silly x) is not a value.
What is a proof?

A proof is a logical sequence of steps, leading to a conclusion.

- Each step must follow logically from math facts, or the results of earlier steps.

An excellent proof has a true conclusion

A bogus proof can have a false conclusion

(again)
Using simple induction

• **Q**: When can I use *simple* induction to prove a property of a recursive function $f$?

• **A**: When we can find a *non-negative* measure of *argument size* and show that if $f(x)$ calls $f(y)$ then $size(y) = size(x) - 1$

pick a notion of size appropriate for $f$
fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [ ] = 0
| sum (x::R) = x + sum R
Which of the following can be proven by simple induction?

fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [ ] = 0
   | sum (x::R) = x + sum R
Examples

fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [] = 0
  | sum (x::R) = x + sum R

Which of the following can be proven by simple induction?

fact is total
Examples

fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [ ] = 0
| sum (x::R) = x + sum R

Which of the following can be proven by simple induction?
Examples

fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [ ] = 0
| sum (x::R) = x + sum R

Which of the following can be proven by simple induction?

For all \( n \geq 0 \), \( \text{fact } n \) evaluates to an integer value
Examples

fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [ ] = 0
| sum (x::R) = x + sum R

Which of the following can be proven by simple induction?
Examples

fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [ ] = 0
| sum (x::R) = x + sum R

Which of the following can be proven by simple induction?

sum is total
Examples

fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [] = 0
| sum (x::R) = x + sum R

Which of the following can be proven by simple induction?
Examples

fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [] = 0
| sum (x::R) = x + sum R

Which of the following can be proven by simple induction?

For all \( n \geq 0 \), \( \text{fact } n > n \)
Examples

fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [ ] = 0
| sum (x::R) = x + sum R

Which of the following can be proven by simple induction?
fun fact (x:int) : int = if x=0 then 1 else x * fact(x-1)

fun sum [ ] = 0
|  sum (x::R) = x + sum R

Which of the following can be proven by simple induction?

For all n>1, fact n > n
To prove:

For all integer lists $L$

there is an integer $n$ such that

$\text{eval } L \implies \ast n$
**eval again**

**eval**: int list -> int

```ml
fun eval [ ] = 0
| eval (d::L) = d + 10 * (eval L)
```

(The *length of the argument list* decreases in the recursive call)

**To prove:**

For all integer lists L, there is an integer n such that

\[
eval L \implies^* n
\]
Exercise

• Prove the specification for eval
• It’s a simple induction on list length

This shows that eval : int list -> int is a total function.

For all values L : int list, eval L evaluates to a value.
Life’s not simple

You cannot use **simple** induction on \( n \) for

```haskell
fun decimal (n:int) : int list =
  if n<10 then [n]
  else (n mod 10) :: decimal (n div 10)
```

Why not?

We need a *stronger* form of induction…
Strong induction

• To prove a property of the form

\[ P(n), \text{ for all non-negative integers } n \]

Show that, for all \( k \geq 0, \)

\[ P(k) \text{ follows logically from } P(0), \ldots, P(k-1). \]

you can use any, all, or none to establish \( P(k) \)
Why this works

• P(0) gets a direct proof
  
• P(0) implies P(1)
  
• P(0), P(1) imply P(2)
  
• P(0), P(1), P(2) imply P(3)

For each \( k \geq 0 \) we can establish P(k) with \( k \) uses of step
Using strong induction

• **Q:** When can I use strong induction to prove a property of a recursive function $f$?

• **A:** When we can find a non-negative measure of *argument size* and show that if $f(x)$ calls $f(y)$ then $size(y) < size(x)$
Notes

• Sometimes, even for simple induction, it’s convenient to handle several “base” cases at the same time.

• A proof using strong induction may not need a separate “base” case analysis.
  • can sometimes handle all possible arguments in the “inductive step”
Example

fun decimal (n:int) : int list =
  if n<10 then [n]
  else (n mod 10) :: decimal (n div 10)

To prove:

For all values $n \geq 0$,
$\text{eval}(\text{decimal } n) = n$
Example

fun decimal (n:int) : int list =
    if n<10 then [n]
    else (n mod 10) :: decimal (n div 10)

When n ≥ 10, we get 0 ≤ n div 10 < n

To prove:

For all values n ≥ 0,
eval(decimal n) = n
fun decimal (n:int) : int list = 
  if n<10 then [n] 
  else (n mod 10) :: decimal (n div 10)

When \( n \geq 10 \), we get \( 0 \leq n \div 10 < n \)

so the argument value decreases,
stays non-negative,
in the recursive call

To prove:

For all values \( n \geq 0 \),
eval(decimal n) = n
Proof by strong induction

• For $0 \leq n < 10$, show directly that $\text{eval(decimal n)} = n$

• For $n \geq 10$, assume that

  For each $m$ such that $0 \leq m < n$, $\text{eval(decimal m)} = m$

  Then show that $\text{eval(decimal n)} = n$

use inductive analysis for cases that make a recursive call

multiple base cases handled together
fun eval [ ]  = 0
  | eval (d::L) = d + 10 * (eval L)

fun decimal n  = 
    if n<10 then [n]
    else (n mod 10) :: decimal (n div 10)

We want to prove:

For all values \( n \geq 0 \),

\[ \text{eval}(\text{decimal } n) = n \]

Proof: will be by strong induction on \( n \)
Proof sketch
(the base cases)

• For $0 \leq n < 10$ we have

\[
\text{eval(decimal n)} = \text{eval} [n] = n
\]

(That was easy!)

(We used the function definitions!)
Proof sketch
(the inductive part)
Proof sketch
(the inductive part)

• For $n \geq 10$ let $r = n \text{ mod } 10, q = n \text{ div } 10$. 
Proof sketch
(the inductive part)

• For $n \geq 10$ let $r = n \mod 10, q = n \div 10$.
  
  eval(decimal n)
  = eval ((n mod 10) :: decimal(n div 10))
  = eval (r :: decimal q)
Proof sketch

(the inductive part)

• For $n \geq 10$ let $r = n \mod 10, q = n \div 10$.

  eval(decimal n)
  = eval ((n mod 10) :: decimal(n div 10))
  = eval (r :: decimal q)

• Since $0 \leq q < n$ it follows from IH that
Proof sketch
(the inductive part)

• For $n \geq 10$ let $r = n \mod 10, q = n \div 10$.
  $\text{eval(decimal n)}$
  $= \text{eval} ((n \mod 10) :: \text{decimal(n div 10)})$
  $= \text{eval} (r :: \text{decimal q})$

• Since $0 \leq q < n$ it follows from IH that
  $\text{eval(decimal q)} = q$
Proof sketch
(the inductive part)

• For \( n \geq 10 \) let \( r = n \mod 10, q = n \div 10 \).
  
  \[
  \text{eval(decimal n)}
  = \text{eval \((n \mod 10) :: \text{decimal(n div 10)})}\n  = \text{eval \(r :: \text{decimal q}\)}
  
• Since \( 0 \leq q < n \) it follows from IH that
  
  \[
  \text{eval(decimal q)} = q
  
• Hence there is a list value \( Q \) such that
Proof sketch (the inductive part)

- For \( n \geq 10 \) let \( r = n \mod 10, q = n \div 10 \).
  
  \[
  \text{eval(decimal n)} = \text{eval ((n mod 10) :: decimal(n div 10))} = \text{eval (r :: decimal q)}
  \]

- Since \( 0 \leq q < n \) it follows from IH that
  \[
  \text{eval(decimal q)} = q
  \]

- Hence there is a list value \( Q \) such that
  \[
  \text{decimal q} = Q \text{ and eval } Q = q
  \]
Proof sketch
(the inductive part)

• For \( n \geq 10 \) let \( r = n \ mod \ 10, q = n \ div \ 10 \).

  \[
  \begin{align*}
  \text{eval(decimal n)} &= \text{eval} ((n \ mod \ 10) :: \text{decimal}(n \ div \ 10)) \\
  &= \text{eval} (r :: \text{decimal} q)
  \end{align*}
  \]

• Since \( 0 \leq q < n \) it follows from IH that

  \[
  \text{eval(decimal q)} = q
  \]

• Hence there is a list value \( Q \) such that

  \[
  \text{decimal} q = Q \ \text{and} \ \text{eval} \ Q = q
  \]

So
Proof sketch
(the inductive part)

• For \( n \geq 10 \) let \( r = n \mod 10, q = n \div 10 \).
  \[
  \text{eval(decimal n)} = \text{eval } ((n \mod 10) :: \text{decimal(n div 10)}) = \text{eval (r :: decimal q)}
  \]

• Since \( 0 \leq q < n \) it follows from IH that
  \[
  \text{eval(decimal q)} = q
  \]

• Hence there is a list value \( Q \) such that
  \[
  \text{decimal q} = Q \text{ and eval } Q = q
  \]
  So
  \[
  \text{eval (r :: decimal q)} = \text{eval (r::Q)} = r + 10 \times (\text{eval Q}) = r + 10 \times q = n
  \]
Proof sketch
(the inductive part)

• For \( n \geq 10 \) let \( r = n \mod 10, q = n \div 10 \).

\[
\begin{align*}
\text{eval}(\text{decimal n}) &= \text{eval} \ ((n \mod 10) :: \text{decimal}(n \div 10)) \\
&= \text{eval} \ (r :: \text{decimal q})
\end{align*}
\]

• Since \( 0 \leq q < n \) it follows from IH that

\[
\begin{align*}
\text{eval(\text{decimal q})} &= q
\end{align*}
\]

• Hence there is a list value \( Q \) such that

\[
\text{decimal q} = Q \text{ and eval } Q = q
\]

So

\[
\begin{align*}
\text{eval} \ (r :: \text{decimal q}) &= \text{eval} \ (r::Q) \\
&= r + 10 \times (\text{eval } Q) \\
&= r + 10 \times q = n
\end{align*}
\]

This shows that \( \text{eval(\text{decimal n})} = n \)
Proof sketch
(conclusion)

Let $P(n)$ be “eval(decimal n) = n”

• The base analysis shows $P(0), P(1),..., P(9)$

• The inductive analysis shows that for $n \geq 10$, $P(n)$ follows from \{P(0),...P(n-1)\}

• Hence, for all $n \geq 0$, $P(n)$ holds
Notes

• We used equational reasoning to show that for all \textit{values} \( n \geq 0 \), \( \text{eval(\text{decimal } n)} = n \)

• It follows that for all \textit{expressions} \( e : \text{int} \), if \( e \rightarrow^{*} n \) and \( n \geq 0 \), then
\[
\text{eval(\text{decimal } e)} \rightarrow^{*} n
\]
So far

• Simple and strong induction
• Examples of their use
• Just the beginning…
fun log(x:int):int =
    if x=1 then 0 else 1 + log(x div 2)

• What would you do?
fun log(x:int):int = 
  if x=1 then 0 else 1 + log(x div 2)
fun \text{log}(x:\text{int}) : \text{int} = \\
\text{if } x = 1 \text{ then } 0 \text{ else } 1 + \log(x \div 2)
fun log(x:int):int = 
    if x=1 then 0 else 1 + log(x div 2)

log : int -> int
REQUIRES n > 0
**fun** \( \log(x: \text{int}): \text{int} = \)

\[
\text{if } x = 1 \text{ then } 0 \text{ else } 1 + \log(x \div 2)
\]

\( \log : \text{int} \rightarrow \text{int} \)

**REQUIRES** \( n > 0 \)

**ENSURES** \( \log n \) keeps dividing \( n \) by 2 until it gets to 1
fun log(x:int):int =  
\text{if } x = 1 \text{ then } 0 \text{ else } 1 + \log(x \div 2)

log : int -> int

REQUIRES n > 0

ENSURES log n keeps dividing n by 2 until it gets to 1

\textit{too vague... doesn’t describe the result}
fun log(x:int):int = 
    if x=1 then 0 else 1 + log(x div 2)

log : int -> int

REQUIRES n > 0
fun log(x:int):int = 
  if x=1 then 0 else 1 + log(x div 2)

log : int -> int

REQUIRES n > 0

ENSURES log n evaluates to an integer k
fun log(x:int):int = 
   if x=1 then 0 else 1 + log(x div 2)

log : int -> int

REQUIRES n > 0

ENSURES log n evaluates to an integer k such that $2^k \leq n < 2^{k+1}$
fun log(x:int):int = 
  if x=1 then 0 else 1 + log(x div 2)

log : int -> int

REQUIRES n > 0

ENSURES log n evaluates to an integer k such that $2^k \leq n < 2^{k+1}$

describes the key properties of the result value
Exercise

• Show that for each integer \( n > 0 \), there is a unique integer \( k \) such that \( 2^k \leq n < 2^{k+1} \)
  
  • this \( k \) is called the logarithm (base 2) of \( n \)

• Prove the spec for \( \log \)
  
  \( \log \) computes logarithms (base 2)