So far, we have seen

- Defining types and sets recursively.
- Proving propositions quantified over recursively defined sets using structural induction.
- Proving propositions quantified over \( \mathbb{W} \) or \( \mathbb{N} \) using mathematical induction. Specifically, to prove \( \forall n \in \mathbb{W}, \; I(n) \),
  - Prove \( I(0) \)
  - Prove \( \forall n \in \mathbb{W}, \; I(n) \implies I(n + 1) \)

Today and Wednesday are about

- Proving the correctness of algorithms using mathematical induction
For Friday, Nov 12:

Pg 306: 6.10.(2-5)

Read 7 intro and 7.1 carefully
Read 7.2
Skim 7.3
\[ n! = \begin{cases} 
1 & \text{if } n = 0 \\
 n \cdot (n-1)! & \text{otherwise}
\end{cases} \]

\begin{verbatim}
fun factorial(0) = 1
| factorial(n) = n * factorial(n-1);
\end{verbatim}

**Theorem 6.6.** For all \( n \in \mathbb{W} \), \( \text{factorial}(n) = n! \)

**Proof.** By induction on \( n \).

**Base case.** Suppose \( n = 0 \). By definition of \( \text{factorial} \), \( \text{factorial}(0) = 1 = 0! \), by definition of \(!\). Hence there exists an \( N \geq 0 \) such that \( \text{factorial}(N) = N! \).

**Inductive case.** Suppose \( N \geq 0 \) such that \( \text{factorial}(N) = N! \), and suppose \( n = N + 1 \). Then

\[
\text{factorial}(n) = n \cdot \text{factorial}(n-1) \quad \text{by definition of \( \text{factorial} \)}
\]
\[
= n \cdot \text{factorial}(N) \quad \text{by algebra and substitution}
\]
\[
= n \cdot N! \quad \text{by the inductive hypothesis}
\]
\[
= n! \quad \text{by definition of \(!\)}
\]

Therefore, by math induction, \( \text{factorial} \) is correct for all \( n \in \mathbb{W} \). \( \square \)
What does *correctness* mean for an algorithm?

The outcome/result must always match the specification. For arithSum, the specification is

\[
arithSum(N) = \sum_{k=1}^{N} k
\]

To prove this, we need to reason about the *change of state* of the computation. The *state* of the computation is represented by the values of the variables.
We can reason about a single line of code in terms of *preconditions* and *postconditions*. Suppose the preconditions include $x = 5$.

\[
y := x + 1
\]

Then the postconditions include
\begin{itemize}
  \item $y = 6$
  \item $x = 5$
  \item $y = x - 1$
  \item $G = 6.674 \times 10^{-11} \text{ m}^3\text{kg}^{-1}\text{s}^{-2}$
\end{itemize}
fun remainder(a, b) =
  let
    val q = a div b;
    Suppose \( a, b \in \mathbb{Z} \)
    \( q = a \div b \) by assignment. By the QRT (Thm 4.21)
    and the definition of division, \( a = b \cdot q + R \) for some \( R \),
    \( 0 \leq R < b \). Then by algebra, \( q = \frac{a-R}{b} \).
    val p = q * b;
    \( p = q \cdot b \) by assignment, and \( p = a - R \) by substitution
    and algebra.
    val r = a - p;
    By assignment, \( r = a - p \). By substitution and algebra,
    \( r = a - (a - R) = R \).
  in
    r
  end;
Since \( r \) is the value returned and is equal to the specified result \( R \),
this program returns the correct result. \( \square \)
For arithSum, \( N \) is the limit on the summation. Let \( n \) be the number of iterations so far. Our claim is

\[
\text{After } n \text{ iterations, } \ s = \sum_{k=1}^{n} k
\]

Notice

- After 0 iterations, \( s = 0 \) and \( \sum_{k=1}^{0} k = 0 \). Our claim is true before we start.
- Each iteration changes the state, but maintains the fact above (or, so we claim).
- When we’re done, that’s \( N \) iterations, so \( \sum_{k=1}^{n} k = \sum_{k=1}^{N} k \) (or, so we claim).

Refining the claim:

\[
\forall \ n \in \mathbb{W}, \ \text{after } n \text{ iterations } s = \sum_{k=1}^{n} k \text{ and } i = n + 1
\]
**Theorem.** \( \text{arithSum}(N) \) returns \( \sum_{k=1}^{N} k \).

**Lemma.** \( \forall \ n \in \mathbb{N}, \) after \( n \) iterations, \( s = \sum_{k=1}^{n} k \) and \( i = n + 1 \).

**Proof (of lemma).** By induction on the number of iterations, \( n \).

**Initialization.** After 0 iterations, \( s = 0 = \sum_{k=1}^{0} k \) by assignment, arithmetic, and definition of summation. \( i = 1 = 0 + 1 \), by assignment and arithmetic.

**Maintenance.** Suppose after \( n \geq 0 \) iterations, \( s = \sum_{k=1}^{n} k \) and \( i = n + 1 \).

Let \( s_{\text{old}} \) be \( s \) after \( n \) iterations and \( s_{\text{new}} \) be \( s \) after \( n + 1 \) iterations. Similarly define \( i_{\text{old}} \) and \( i_{\text{new}} \). Then

\[
\begin{align*}
    s_{\text{new}} &= s_{\text{old}} + i_{\text{old}} & \text{by assignment} \\
    &= (\sum_{k=1}^{n} k) + n + 1 & \text{by the inductive hypothesis} \\
    &= \sum_{k=1}^{n+1} k & \text{by the definition of summation} \\
    i_{\text{new}} &= i_{\text{old}} + 1 & \text{by assignment} \\
    &= n + 1 + 1 & \text{by the inductive hypothesis} \\
    &= (n + 1) + 1 & \text{by associativity}
\end{align*}
\]

Therefore the invariant holds. \( \square \)
**Theorem.** \( \text{arithSum}(N) \) returns \( \sum_{k=1}^{N} k \).

**Lemma.** \( \forall n \in \mathbb{W}, \text{after } n \text{ iterations, } s = \sum_{k=1}^{n} k \text{ and } i = n + 1. \)

**Proof (of theorem).** Suppose \( N \in \mathbb{W} \) is the input to \( \text{arithSum} \).

**Termination.** The lemma tells us that after \( N \) iterations, \( i = N + 1 \not\leq N \), so the guard fails and the loop terminates.

At loop exit, \( s = \sum_{k=1}^{N} k \), which is return.

Therefore the program \( \text{arithSum} \) is correct. \( \square \)
Principles of using loop invariants to prove correctness

- A loop invariant is a proposition that is true before and after each iteration of a loop, including before the entire loop starts and after it terminates. A useful loop invariant captures an aspect of the progress of the loop’s work.

- The steps in a loop invariant proof, to prove and apply something in the form, “∀ \( n \in \mathbb{W} \), after \( n \) iterations, . . . .”
  - **Initialization.** Prove that the property is true before the loop starts, that is, after 0 iterations. This is the base case in the inductive proof.
  - **Maintenance.** Prove that if the property is true before an iteration, then it is true after that iteration. This is the inductive case of the inductive proof.
  - **Termination.** Prove that the loop will terminate, and then apply the loop invariant to deduce a postcondition for the entire loop.
fun aaa(m) = 
    let
        val x = ref 0;
        val i = ref 0;
    in
        (while !i < m do
            (x := !x + 2 * !i;
             i := !i + 1);
            !x)
    end;

After \( n \) iterations, \( x \) is even.

**Proof.** By induction on the number of iterations.

**Initialization.** Before the loop starts, \( x = 0 \) by assignment. Moreover, \( x = 2 \cdot 0 \), so \( x \) is even by definition.

**Maintenance.** Suppose that after \( n \) iterations \( x \) is even, for some \( n \geq 0 \). Let \( x_{\text{old}} \) and \( x_{\text{new}} \) be \( x \) after \( n \) and \( n+1 \) iterations, respectively.

\( x_{\text{old}} = 2j \) for some \( j \in \mathbb{Z} \) by the inductive hypothesis and definition of even. Then

\[
    x_{\text{new}} = x_{\text{old}} + 2i \quad \text{by assignment}
\]

\[
    = 2j + 2i \quad \text{by substitution}
\]

\[
    = 2(j + i) \quad \text{by algebra}
\]

Hence \( x_{\text{new}} \) is even by definition.

Therefore, by the principle of mathematical induction, that \( x \) is even is a loop invariant. \( \square \)
fun pow(x, y) =
  let
    val a = ref 1;
    val i = ref y;
  in
    (while !i > 0 do
      (i := !i - 1;
        a := !a * x);
      !a)
  end;

After \( n \) iterations, \( a = x^n \) and \( i = y - n \).

Proof. By induction on the number of iterations.

Initialization. Suppose \( n = 0 \), that is, the conditions before the loop starts. Then \( a = 1 \) by assignment, and hence \( a = x^0 = x^n \) by algebra. Similarly, \( i = y \) by assignment, and hence \( i = y - 0 = y - n \) by algebra.

Maintenance. Suppose that \( a = x^n \) and \( i = y - n \) after \( n \) iterations for some \( n \geq 0 \). Let \( a_{\text{old}}, a_{\text{new}}, i_{\text{old}}, \) and \( i_{\text{new}} \) be defined in the usual way. Then

\[
\begin{align*}
  i_{\text{new}} &= i_{\text{old}} - 1 & \text{by assignment} \\
  &= y - n - 1 & \text{by the inductive hypothesis} \\
  &= y - (n + 1) & \text{by algebra} \\
  a_{\text{new}} &= a_{\text{old}} \cdot x & \text{by assignment} \\
  &= x^n \cdot x & \text{by the inductive hypothesis} \\
  &= x^{n+1} & \text{by algebra}
\end{align*}
\]

Therefore, by the principle of mathematical induction, \( a = x^n \) and \( i = y - n \), where \( n \) is the number of iterations completed, is a loop invariant. \( \Box \)
After \( n \) iterations, \( x + y = m \).

fun xxx(m) =
  let
    val x = ref m;
    val y = ref 0;
    val i = ref 1;
  in
    (while !i < m div 2 do
      (x := !x - i;
        y := !y + i;
        i := !i * 2);
      !x - !y)
  end;
After \( n \) iterations, \( x + y = m \).

**Proof.** By induction on the number of iterations.

```plaintext
fun xxx(m) =
  let
    val x = ref m;
    val y = ref 0;
    val i = ref 1;
  in
    (while !i < m div 2 do
      (x := !x - i;
       y := !y + i;
       i := !i * 2);
      !x - !y)
  end;
```
fun xxx(m) =
  let
    val x = ref m;
    val y = ref 0;
    val i = ref 1;
  in
    (while !i < m div 2 do
    x := !x - i;
    y := !y + i;
    i := !i * 2;
    !x - !y)
  end;

After \( n \) iterations, \( x + y = m \).

**Proof.** By induction on the number of iterations.

**Initialization.** Before the loop starts, \( x = m \) and \( y = 0 \) by assignment. Hence \( x + y = m \) by algebra.

**Maintenance** Suppose \( x + y = m \) after \( n \) iterations for some \( n \geq 0 \). Let \( x_{\text{old}}, x_{\text{new}}, y_{\text{old}}, \) and \( y_{\text{new}} \) be defined in the usual way. Then

\[
x_{\text{new}} = x_{\text{old}} - i \quad \text{by assignment}
\]

\[
y_{\text{new}} = y_{\text{old}} + i \quad \text{by assignment}
\]

\[
x_{\text{new}} + y_{\text{new}} = x_{\text{old}} - i + y_{\text{old}} + i
\]

\[
= x_{\text{old}} + y_{\text{old}} \quad \text{by substitution}
\]

\[
= m \quad \text{by the inductive hypothesis}
\]

Therefore, by the principle of mathematical induction, \( x + y = m \) is a loop invariant.  \( \square \)
fun xxx(m) = 
let
  val x = ref m;
  val y = ref 0;
  val i = ref 1;
in
  (while !i < m div 2 do
    (x := !x - i;
     y := !y + i;
     i := !i * 2);
  !x - !y)
end;

After n iterations, \(x + y = m\).

\textbf{Proof.} By induction on the number of iterations.

\textbf{Initialization.} Before the loop starts, \(x = m\) and \(y = 0\) by assignment. Hence \(x + y = m\) by algebra.

\textbf{Maintenance} Suppose \(x + y = m\) after \(n\) iterations for some \(n \geq 0\). Let \(x_{\text{old}}, x_{\text{new}}, y_{\text{old}},\) and \(y_{\text{new}}\) be defined in the usual way. Then
fun xxx(m) =
    let
        val x = ref m;
        val y = ref 0;
        val i = ref 1;
    in
        (while !i < m div 2 do
            (x := !x - i;
            y := !y + i;
            i := !i * 2);
        !x - !y)
    end;

After n iterations, \( x + y = m \).

Proof. By induction on the number of iterations.
Initialization. Before the loop starts, \( x = m \) and \( y = 0 \) by assignment. Hence \( x + y = m \) by algebra.

Maintenance Suppose \( x + y = m \) after \( n \) iterations for some \( n \geq 0 \). Let \( x_{\text{old}}, x_{\text{new}}, y_{\text{old}}, \) and \( y_{\text{new}} \) be defined in the usual way. Then

\[
\begin{align*}
    x_{\text{new}} &= x_{\text{old}} - i & \text{by assignment} \\
    y_{\text{new}} &= y_{\text{old}} + i & \text{by assignment} \\
    x_{\text{new}} + y_{\text{new}} &= x_{\text{old}} - i + y_{\text{old}} + i & \text{by substitution} \\
    &= x_{\text{old}} + y_{\text{old}} & \text{by algebra} \\
    &= m & \text{by the inductive hypothesis}
\end{align*}
\]
fun xxx(m) = 
  let
    val x = ref m;
    val y = ref 0;
    val i = ref 1;
  in
  (while !i < m div 2 do
    (x := !x - i;
     y := !y + i;
     i := !i * 2);
  !x - !y)
end;

After \( n \) iterations, \( x + y = m \).

**Proof.** By induction on the number of iterations.

**Initialization.** Before the loop starts, \( x = m \) and \( y = 0 \) by assignment. Hence \( x + y = m \) by algebra.

**Maintenance** Suppose \( x + y = m \) after \( n \) iterations for some \( n \geq 0 \). Let \( x_{\text{old}}, x_{\text{new}}, y_{\text{old}}, \text{ and } y_{\text{new}} \) be defined in the usual way. Then

\[
\begin{align*}
  x_{\text{new}} &= x_{\text{old}} - i & \text{by assignment} \\
  y_{\text{new}} &= y_{\text{old}} + i & \text{by assignment} \\
  x_{\text{new}} + y_{\text{new}} &= x_{\text{old}} - i + y_{\text{old}} + i & \text{by substitution} \\
  &= x_{\text{old}} + y_{\text{old}} & \text{by algebra} \\
  &= m & \text{by the inductive hypothesis}
\end{align*}
\]

Therefore, by the principle of mathematical induction, \( x + y = m \) is a loop invariant. \( \square \)
Reminder: Ex 6.10.(2-5) for next time.
Also (very important):
  ▶ Read 7 intro and 7.1 *carefully*
  ▶ Read 7.2
  ▶ Skim 7.3