Motivation

Q: How do you

• define an infinite domain, or
• prove properties of an infinite domain?

A: Use induction.

Examples of infinite domains: Natural numbers \( \mathbb{N} \), set of all predicate logic formulas, languages generated by finite state automata, etc.

These can be defined recursively.

Recall definition of predicate logic formulas:

**Def:** A *formula* is defined as follows:

1. If \( t_1, \ldots, t_n \) are terms and \( P \) is an \( n \)-ary predicate symbol \( P(t_1, \ldots, t_n) \) is an (*atomic*) formula.

2. If \( \phi \) and \( \psi \) are formulas, so are:

\[
\neg \phi, (\phi \land \psi), (\phi \lor \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)
\]

\( \top \) and \( \bot \) are also formulas.

3. If \( x \) is a variable and \( \phi \) is a formula, then so are \( (\forall x \phi) \) and \( (\exists x \phi) \).

Formula is defined in terms of itself.
Misuse of Induction

Consider function \( f(n) = \frac{1}{100.0001 + n^2 - n^3} \):

\[
\begin{align*}
  f(1) &= 0.01 \\
  f(4) &= 0.000651 \\
  f(5) &= 0.000421 \\
  f(6) &= 0.000296
\end{align*}
\]

Therefore for every \( n \geq 1 \), \( f(n) \leq 0.01 \).

Wrong! \( f(100) = 10 \)

It is not sufficient to show \( \phi \) is true for several \( n \) to conclude \( \forall n \phi \).

Peano Arithmetic

How do we define \( \mathbb{N} \) rigorously?

Use 0 and successor function \( s : \mathbb{N} \to \mathbb{N} \). Can define \( + \) and \( \cdot \) in terms of \( s \).

Then \( s^M(n) = n + 1 \) as expected.

1. 0 is a natural number.
2. If \( n \) is a natural number then so is \( s(n) \).
3. 0 is not a successor: \( \forall x(s(x) \neq 0) \)
4. Uniqueness of successors:
   \( \forall x \forall y (s(x) = s(y) \to x = y) \)
5. Induction postulate: For any formula \( \phi \)
   \( \phi[0/x] \land \forall y (\phi[y/x] \to \phi[s(y)/x]) \to \forall x \phi \)

Mathematical Induction

Rule MI: Let \( \phi \) be any formula of Peano Arithmetic. Then if

1. Base Step: \( \vdash \phi[0/n] \), and
2. Inductive Step:
   \( \vdash \forall n (\phi[m/n] \to \phi[m + 1/n]) \)

Then \( \vdash \forall n \phi \) by Rule MI.

Why is this a valid rule of inference? By 1 and repeatedly applying \( \forall e \) followed by \( \to e \) (modus ponens) on 2 can create proof for any natural number \( k \).

Do informal proof using mathematical induction of:

\( \forall n (2(n + 2) \leq (n + 2)^2) \)
Changing the Base Case

How do we prove $2^n < n!$ for $n \geq 4$ using mathematical induction?

More generally, how do we show:

$$\forall n (n \geq n_0 \rightarrow \phi)$$

1. Base Step: $\vdash \phi[0/n]$
2. Inductive Step: Show
   $$\vdash \forall m (m \geq n_0 \land \phi[m/n] \rightarrow \phi[m + 1/n])$$

Then conclude $\forall n (n \geq n_0 \rightarrow \phi)$ by Rule MI.

Ex. Informal proof of $\forall n (n \geq 4 \rightarrow 2^n < n!)$

Complete Induction

**Thm:** Complete Induction (CI) Let $\phi$ be a formula of Peano Arithmetic s.t. $x \in FV(\phi)$ and $y, z$ do not occur in $\phi$. Then

$$\phi[0/x] \land \forall y [\forall z (z \leq y \rightarrow \phi[z/x]) \rightarrow \phi[y + 1/x]]$$

is a theorem of Peano Arithmetic (i.e. its true).

**Interpretation:** If you can show

1. $\phi$ is true at 0, and
2. By assuming $\phi$ is true for every natural number up to and including $y$, you can prove $\phi[y + 1/x]$ is true.

Then conclude $\phi$ is true for every natural number.

Application: Correctness of Loops

**Assertion:** Any statement about a program state.

**Def:** Let $C$ be a program statement or sequence of statements, $\{P\}$ be **precondition** of $C$, an assertion on the initial state and $\{Q\}$ be a **postcondition**, an assertion on the final state. Then $\{P\}C\{Q\}$ is a **Hoare triple**.

Ex 1: $\{True\}a := b(a = b)$ or equivalently $\{a := b(a = b)\}$.

Ex 2: $\{y \neq 0\}x := 1/y\{x = 1/y\}$

**The While Rule:** Let $C$ be a piece of code such that: $\{D \land I\}C\{I\}$. Then

$$\{D \land I\} \text{ while } D \text{ do } C \{\neg D \land I\}$$

$\neg D$ is the **exit condition** and $I$ is the **loop invariant**.
Proof of While Rule:
Assume loop terminates in \( n \) iteration.
Must show \( \neg D \land I \) upon termination. But \( \neg D \) must be true upon termination so remains to show \( I \).


**Base case:** \( I \) is true before entering loop so \( I \) true for 0 iterations

**Inductive case:** Assume \( I \) true after \( m \) iterations for \( 0 \leq m < n \).
Must show \( I \) is true after \( m + 1 \) iterations.
But \( D \) is true before executing \( C \) for the \( m + 1 \)th time since loop does not terminate after \( m \) iterations \((m < n)\).
Also \( I \) is true before execution by inductive hyp.
\( \{D \land I\} \) is a precondition for \( m + 1 \) execution \( C \).
Therefore \( \{I\} \) is a postcondition since \( \{D \land I\} \subseteq \{I\} \).
Q.E.D.

Suppose you have a very RISCy CPU that uses addition to do multiplication \( n \cdot a \) with the code:

\[
\begin{align*}
\text{sum} & := 0; \\
n & := 0; \\
\text{while } j < n \\
\text{Begin} \\
& \quad \text{sum} := \text{sum} + a; \\
& \quad j := j + 1; \\
\text{End}
\end{align*}
\]
Assume \( n \geq 0 \). Take
\( D : j \neq n \)
\( I : \neg j \leq n \land \text{sum} = j \cdot a \)

Checklist for proving loop correct:

1. \( I \) true before loop
2. \( I \) is loop invariant: \( \{D \land I\} \subseteq \{I\} \)
3. Execution terminates
4. Use \( \neg D \land I \) to prove desired property
   (e.g. \( \text{sum} = n \cdot a \))