Induction

Till Mossakowski

WiSe 2009/10
Induction is like a chain of dominoes. You need

- the dominoes must be close enough together $\Rightarrow$ one falling dominoe knocks down the next (*inductive step*)
- you need to knock down the first dominoe (*inductive basis*)

Note: in the inductive step, branching is possible.
Induction is like a chain of dominoes. You need

- the dominoes must be close enough together \( \Rightarrow \) one falling dominoe knocks down the next (*inductive step*)
- you need to knock down the first dominoe (*inductive basis*)

Note: in the inductive step, branching is possible.
Inductive definition: Natural numbers

1. 0 is a natural number.
2. If \( n \) is natural number, then \( \text{suc}(n) \) is a natural number.
3. There is no natural number whose successor is 0.
4. Two different natural numbers have different successors.
5. Nothing is a natural number unless generated by repeated applications of (1) and (2).
Recursive definition of functions

\[ \forall y \, (0 + y = y) \]
\[ \forall x \forall y \, (suc(x) + y = suc(x + y)) \]

\[ \forall y \, (0 \times y = 0) \]
\[ \forall x \forall y \, (suc(x) \times y = (x \times y) + y) \]
Formalization of Peano’s axioms

1. a constant 0
2. a unary function symbol $suc$
3. $\forall n \neg suc(n) = 0$
4. $\forall m \forall n suc(m) = suc(n) \rightarrow m = n$
5. $(\Phi(x/0) \land \forall n(\Phi(x/n) \rightarrow \Phi(x/suc(n)))) \rightarrow \forall n \Phi(x/n)$

if $\Phi$ is a formula with a free variable $x$, and $\Phi(x/t)$ denotes the replacement of $x$ with $t$ within $\Phi$
Inductive proofs

Take $\Phi(x) := \forall y \forall z (x + (y + z) = (x + y) + z)$. Then

$$(\Phi(x/0) \land \forall n(\Phi(x/n) \rightarrow \Phi(x/suc(n)))) \rightarrow \forall n \Phi(x/n)$$

is just

$$(\forall y \forall z (0 + (y + z) = (0 + y) + z)$$
$$\land \forall n \forall y \forall z (n + (y + z) = (n + y) + z$$
$$\rightarrow suc(n) + (y + z) = (suc(n) + y) + z))$$
$$\rightarrow \forall n \forall y \forall z (n + (y + z) = (n + y) + z)$$

With this, we can prove $\forall n \forall y \forall z (n + (y + z) = (n + y) + z)$
1. The empty list \([\ ]\) is a list.

2. If \(l\) is a list and \(n\) is a natural number, then \(\text{cons}(n, l)\) is a list.

3. Nothing is a list unless generated by repeated applications of (1) and (2).

*Note:* This needs *many-sorted* first-order logic.

We have two sorts of objects: natural numbers and lists.
Recursive definition of functions over lists

\[\text{length}([ ]) = 0\]
\[\forall n : \text{Nat} \ \forall l : \text{List} \ (\text{length}(\text{cons}(n, l)) = \text{suc}(\text{length}(l)))\]

\[\forall l : \text{List} \ ([ ] ++ l = l)\]
\[\forall n : \text{Nat} \ \forall l_1 : \text{List} \ \forall l_2 : \text{List} \]
\[\ (\text{cons}(n, l_1) ++ l_2 = \text{cons}(n, l_1 ++ l_2))\]
\( \forall l_1 : \text{List} \ \forall l_2 : \text{List} \ \forall l_3 : \text{List} \\
\quad \ ( l_1 \ +\ + \ (l_2 \ +\ + \ l_3) \ = \ (l_1 \ +\ + \ l_2) \ +\ + \ l_3 ) \)

\( \forall l_1 : \text{List} \ \forall l_2 : \text{List} \\
\quad \ ( \text{length}(l_1 \ +\ + \ l_2) \ = \ \text{length}(l_1) \ + \ \text{length}(l_2) ) \)
A recursive program computing $0 + 1 + 2 + \ldots + n$

```java
public natural sumToRec(natural n) {
    if (n == 0) return 0;
    else return n + sumToRec(n - 1);
}
```
public natural sumUpTo(natural n) {
    natural sum = 0;
    natural count = 0;
    while(count < n) {
        count += 1;
        sum += count;
    }
    return sum;
}

Invariant: sum = 0 + 1 + 2 + ... + count
public natural sumUpTo(natural n) {
    natural sum = 0;
    natural count = 0;
    while(count < n) {
        count += 1;
        sum += count;
    }
    return sum;
}

Invariant: \( sum = 0 + 1 + 2 + \ldots + \text{count} \)
public natural sumDownFrom(natural n) {
    natural sum = 0;
    natural count = n;
    while(count > 0) {
        sum += count;
        count -= 1;
    }
    return sum;
}

Invariant: \( \text{sum} = (\text{count} + 1) + \ldots + n \)
public natural sumDownFrom(natural n) {
    natural sum = 0;
    natural count = n;
    while(count > 0) {
        sum += count;
        count -= 1;
    }
    return sum;
}

\textit{Invariant: } \textit{sum} = (\textit{count} + 1) + \ldots + n
http://www.cs.ucf.edu/~leavens/JML
Exercises

- Language, proof and logic, 16.27 - 16.31
- write a small Java program annotated with JML, such that universal and existential quantifiers are used
First-order structures: motivation

- How to make the notion of *logical consequence* more formal?
- For propositional logic: truth tables $\Rightarrow$ tautological consequence
- For first-order logic, we need also to interpret quantifiers and identity
- The notion of *first-order structure* models Tarski’s world situations and real-world situations using *set theory*
Example

Hausaufgaben 5 - Lösungen
Aufgabe 3: 3 Punkte

Till Mossakowski
Logic
A first-order structure $\mathcal{M}$ consists of:

- a nonempty set $D^\mathcal{M}$, the *domain of discourse*;
- for each $n$-ary predicate $P$ of the language, a set $\mathcal{M}(P)$ of $n$-tuples $\langle x_1, \ldots, x_n \rangle$ of elements of $D^\mathcal{M}$, called the *extension* of $P$.
  The extension of the identity symbol $=$ must be $\{\langle x, x \rangle \mid x \in D^\mathcal{M}\}$;
- for any name (individual constant) $c$ of the language, an element $\mathcal{M}(c)$ of $D^\mathcal{M}$. 
Hausaufgaben 5 - Lösungen

Aufgabe 3: 3 Punkte

Till Mossakowski  Logic
An interpretation according to Tarski’s World

Assume the language consists of the predicates *Cube*, *Dodec* and *Larger* and the names *a*, *b* and *c*.

- $D = \{ \text{Cube}1, \text{Cube}2, \text{Dodec}1 \}$;
- $M(\text{Cube}) = \{ \text{Cube}1, \text{Cube}2 \}$;
- $M(\text{Dodec}) = \{ \text{Dodec}1 \}$;
- $M(\text{Larger}) = \{ (\text{Cube}1, \text{Cube}2), (\text{Cube}1, \text{Dodec}1) \}$;
- $M(=) = \{ (\text{Cube}1, \text{Cube}1), (\text{Cube}2, \text{Cube}2), (\text{Dodec}1, \text{Dodec}1) \}$;
- $M(a) = \text{Cube}1$; $M(b) = \text{Cube}2$; $M(c) = \text{Dodec}1$. 
An interpretation not conformant with Tarski’s World

- $D^M = \{\text{Cube1, Cube2, Dodec1}\}$;
- $M(\text{Cube}) = \{\text{Dodec1, Cube2}\}$;
- $M(\text{Dodec}) = \emptyset$;
- $M(\text{Larger}) = \{(\text{Cube1, Cube1}), (\text{Dodec1, Cube2})\}$;
- $M(=) =$
  \{(\text{Cube1, Cube1}), (\text{Cube2, Cube2}), (\text{Dodec1, Dodec1})\}$;
- $M(a) = \text{Cube1}; M(b) = \text{Cube2}; M(c) = \text{Dodec1}$. 
A *variable assignment* in $\mathcal{M}$ is a (possibly partial) function $g$ defined on a set of variables and taking values in $D^\mathcal{M}$. Given a well-formed formula $P$, we say that the variable assignment $g$ is *appropriate* for $P$ if all the free variables of $P$ are in the domain of $g$, that is, if $g$ assigns objects to each free variable of $P$. 
$D^m = \{\text{Cube1, Cube2, Dodec1}\}$
g1 assigns Cube1, Cube2, Dodec1 to the variables x, y, z, respectively.
g1 is appropriate for $\text{Between}(x, y, z) \land \exists u(\text{Large}(u))$, but not for $\text{Larger}(x, v)$.
g2 is the empty assignment.
g2 is only appropriate for well-formed formulas without free variables (that is, for sentences).
If $g$ is a variable assignment, $g[v/b]$ is the variable assignment

- whose domain is that of $g$ plus the variable $v$, and
- which assigns the same values as $g$, except that
- it assigns $b$ to the variable $v$. 

Till Mossakowski
$[t]_g^m$ is

- $M(t)$ if $t$ is an individual constant, and
- $g(t)$ if $t$ is a variable.
1 $M \models R(t_1, \ldots, t_n)[g]$ iff $\langle [t_1]_g^M, \ldots, [t_n]_g^M \rangle \in M(R)$;
2 $M \models \neg P[g]$ iff it is not the case that $M \models P[g]$;
3 $M \models P \land Q[g]$ iff both $M \models P[g]$ and $M \models Q[g]$;
4 $M \models P \lor Q[g]$ iff $M \models P[g]$ or $M \models Q[g]$ or both;
5 $M \models P \rightarrow Q[g]$ iff not $M \models P[g]$ or $M \models Q[g]$ or both;
6 $M \models P \leftrightarrow Q[g]$ iff ($M \models P[g]$ iff $M \models Q[g]$);
7 $M \models \forall x \ P[g]$ iff for every $d \in D^m$, $M \models P[g[x/d]]$;
8 $M \models \exists x \ P[g]$ iff for some $d \in D^m$, $M \models P[g[x/d]]$. 