Discrete Mathematics with Applications I

COMPSCI&SFWRENG 2DM3

McMaster University, Fall 2019

Wolfram Kahl

2019-11-29

Descending Chains in Numbers

Consider numbers with the usual strict-order $<$
and consider descending chains, like $17 > 12 > 9 > 8 > 3 > \ldots$

Are there infinite descending chains in

- $\mathbb{Z}$?
- $\mathbb{N}$?
- $\mathbb{R}$?
- $\mathbb{R}^+$?
- $\mathbb{Q}^+$?
- $\{k, n : \mathbb{Q} \mid k \in \mathbb{N} \ni n \bullet n + \frac{k}{k+1}\}$?
- $\mathbb{C}$?

Relations with no infinite descending chains are **well-founded**.

Plan for Today

- **Induction, Induction Principles**
- **Relational Semantics of Imperative Programs**

Idea Behind Induction

- Goal: prove $(\forall\, x : U \bullet P\, x)$ for some property $P : U \to \mathbb{B}$
  (With $\neg occurs(`x', `P')$, or, P is **not** a metavariable.)
- Situation: Elements of U are related via $>$ with "simpler" elements (constituents, predecessors, parts, ...)
- If for every $x : U$ there is a proof that

  if $P\, y$ for all predecessors y of x, then $P\, x$,

  then for every $z : U$ with $\neg(P\, z)$:
  - there is a predecessor u of z with $\neg(P\, u)$ (by contraposition and generalised De Morgan)
  - there is an infinite $>$ chain (of elements c with $\neg(P\, c)$) starting at z.
- If there are no infinite $>$ chains in U,
  that is, **if** $<$ **is well-founded**, then:

  Theorem (12.19) **Mathematical induction over** $(U, <)$**:**

$$(\forall\, x \bullet P\, x) \quad \equiv \quad (\forall\, x \bullet (\forall\, y \mid y < x \bullet P\, y) \Rightarrow P\, x)$$

Mathematical Induction in $\mathbb{N}$

Consider $\_succ\_ : \mathbb{N} \leftrightarrow \mathbb{N}$ with $y\ succ\ x \ \equiv\ \mathsf{suc}\ y = x$

Mathematical induction over $(\mathbb{N}, succ)$**:**

$$\begin{aligned}
& (\forall\, x : \mathbb{N} \bullet P\, x) \\
= \ & \langle \text{(12.19) Math. induction; Def. } succ \rangle \\
& (\forall\, x : \mathbb{N} \bullet (\forall\, y : \mathbb{N} \mid \mathsf{suc}\ y = x \bullet P\, y) \Rightarrow P\, x) \\
= \ & \langle \text{(8.18) Range split, with } true \equiv x = 0 \vee x > 0 \rangle \\
& (\forall\, x : \mathbb{N} \mid x = 0 \bullet (\forall\, y : \mathbb{N} \mid \mathsf{suc}\ y = x \bullet P\, y) \Rightarrow P\, x) \wedge \\
& (\forall\, x : \mathbb{N} \mid x > 0 \bullet (\forall\, y : \mathbb{N} \mid \mathsf{suc}\ y = x \bullet P\, y) \Rightarrow P\, x) \\
= \ & \langle \text{(8.14) One-point rule; (8.22) Change of dummy} \rangle \\
& ((\forall\, y : \mathbb{N} \mid \mathsf{suc}\ y = 0 \bullet P\, y) \Rightarrow P\, 0) \wedge \\
& (\forall\, z : \mathbb{N} \bullet (\forall\, y : \mathbb{N} \mid \mathsf{suc}\ y = \mathsf{suc}\ z \bullet P\, y) \Rightarrow P\, (\mathsf{suc}\ z)) \\
= \ & \left\langle \begin{array}{l} \text{(8.13) Empty range, with } \mathsf{suc}\ y = 0 \equiv false; \\ \text{Cancellation of } \mathsf{suc}\text{, (8.14) One-point rule for } \forall \end{array} \right\rangle \\
& P\, 0 \wedge (\forall\, z : \mathbb{N} \bullet P\, z \Rightarrow P\, (\mathsf{suc}\ z))
\end{aligned}$$

Mathematical Induction in $\mathbb{N}$ (ctd.)

Mathematical induction over $(\mathbb{N}, \mathsf{suc})$**:**

$$(\forall\, x : \mathbb{N} \bullet P\, x) \quad \equiv \quad P\, 0 \wedge (\forall\, z : \mathbb{N} \bullet P\, z \Rightarrow P\, (\mathsf{suc}\ z))$$

$$(\forall\, x : \mathbb{N} \bullet P\, x) \quad \equiv \quad P\, 0 \wedge (\forall\, z : \mathbb{N} \bullet P\, z \Rightarrow P\, (z + 1))$$

Absence of infinite $\mathsf{suc}\,\breve{}$ chains is due to the **inductive definition of** $\mathbb{N}$ **with constructors 0 and** suc : "...and nothing else is a natural number."

Mathematical induction over $(\mathbb{N}, <)$ **"Complete induction over** $\mathbb{N}$**":**

$$(\forall\, x : \mathbb{N} \bullet P\, x) \equiv (\forall\, x : \mathbb{N} \bullet (\forall\, y : \mathbb{N} \mid y < x \bullet P\, y) \Rightarrow P\, x)$$

Complete induction gives you a **stronger induction hypothesis** for non-zero x — some proofs become easier.

Example for Complete Induction in $\mathbb{N}$

Mathematical induction over $(\mathbb{N}, <)$ **"Complete induction over** $\mathbb{N}$**":**

$$\frac{(\forall\, x : \mathbb{N} \bullet P\, x) \equiv (\forall\, x : \mathbb{N} \bullet (\forall\, y : \mathbb{N} \mid y < x \bullet P\, y) \Rightarrow P\, x)}{}$$

Theorem: Every natural number greater than 1 is a product of (one or more) prime numbers.

Formalisation: $\forall\, n : \mathbb{N} \bullet 1 < n \Rightarrow (\exists B : \mathit{Bag}\ \mathbb{N} \mid (\forall p \mid p \sqsubseteq B \bullet \mathsf{isPrime}\ p) \bullet \mathit{bagProd}\ B = n)$

Proof:

- **Using** "Complete induction":
  - **For any** \`$n$\`:
    - **Assuming** \`$\forall\, m \mid m < n \bullet 1 < m \Rightarrow (\exists B : \mathit{Bag}\ \mathbb{N} \mid (\forall p \mid p \sqsubseteq B \bullet \mathsf{isPrime}\ p) \bullet \mathit{bagProd}\ B = m)$\`:
      - **Assuming** \`$1 < n$\`:
        - **By cases:** \`$\mathsf{isPrime}\ n$\`, \`$\neg(\mathsf{isPrime}\ n)$\`
          - **Completeness:** By "Excluded middle"
          - **Case** \`$\mathsf{isPrime}\ n$\`:
            - ..."$\exists$-Introduction": $B := \wr n \wr$ ...
          - **Case** \`$\neg(\mathsf{isPrime}\ n)$\`:
            - ...then $n = n_1 \cdot n_2$ with $n_1 < n > n_2$
            - ...with witness: $\mathit{bagProd}\ B_1 = n_1$ and $\mathit{bagProd}\ B_2 = n_2$
            - ...then $\mathit{bagProd}\ (B_1 \cup B_2) = n$ q.e.d.

Mathematical Induction on Sequences

Cons induction: Mathematical induction over $(\mathsf{Seq}\ A, \prec)$ where

$$\prec\ := \{x : A; xs, ys : \mathsf{Seq}\ A \mid x \lhd xs = ys \bullet \langle xs, ys \rangle\}$$

$$(\forall\ xs : \mathsf{Seq}\ A \bullet P\ xs) \quad \equiv \quad P\ \epsilon \wedge (\forall\ xs : \mathsf{Seq}\ A \mid P\ xs \bullet (\forall\ x : A \bullet P(x \lhd xs)))$$

Snoc induction: Mathematical induction over $(\mathsf{Seq}\ A, \prec)$ where

$$\prec\ := \{x : A; xs, ys : \mathsf{Seq}\ A \mid xs \rhd x = ys \bullet \langle xs, ys \rangle\}$$

$$(\forall\ xs : \mathsf{Seq}\ A \bullet P\ xs) \quad \equiv \quad P\ \epsilon \wedge (\forall\ xs : \mathsf{Seq}\ A \mid P\ xs \bullet (\forall\ x : A \bullet P(xs \rhd x)))$$

Strict prefix induction: Mathematical induction over $(\mathsf{Seq}\ A, \prec)$ where

$$\prec := \{us, xs, ys : \mathsf{Seq}\ A \mid us \neq \epsilon \wedge xs \frown us = ys \bullet \langle xs, ys \rangle\}$$

$$(\forall\ xs : \mathsf{Seq}\ A \bullet P\ xs) \quad \equiv \quad (\forall\ xs : \mathsf{Seq}\ A \bullet (\forall\ ys : \mathsf{Seq}\ A \mid ys \prec xs \bullet P\ ys) \Rightarrow P\ xs)$$

Different induction hypotheses make certain proofs easier.

Structural Induction

Structural induction is mathematical induction over, *e.g.*,

- **finite sequences** with the strict suffix relation
- **expressions** with the direct constituent relation
- **propositional formulae** with the strict subformula relation
- **trees** with the appropriate strict subtree relation
- **proofs** with appropriate strict sub-proof relation
- **programs** with appropriate strict sub-program relation
- ...

Expressions as Inductive Datatype

Induction Principles

$$
\begin{aligned}
P[xs := \epsilon] &\Rightarrow (\forall\, xs : \mathsf{Seq}\, A \mid P \bullet (\forall\, x : A \bullet P[xs := x \triangleleft xs]) \\
&\Rightarrow (\forall\, xs : \mathsf{Seq}\, A \bullet P)
\end{aligned}
$$

$$P[m := 0] \Rightarrow (\forall\, m : \mathbb{N} \mid P \bullet P[m := \mathsf{suc}\, m]) \Rightarrow (\forall\, m : \mathbb{N} \bullet P)$$

- Induction principles are just certain kinds of formulalae
- They can be introduced as axioms, or proven as theorems
- `Using` induction principles makes you independent from the hard-coded induction principles underlying "`By induction`"

```
Axiom "Induction over sequences":
    P[xs ≔ ϵ]
    ⇒ (∀ xs : Seq A ❙ P • (∀ x : A • P[xs ≔ x ◃ xs]))
    ⇒ (∀ xs : Seq A • P)
```

```
Axiom "Induction over ℕ":
   P[n ≔ 0]
   ⇒ (∀ n : ℕ ❙ P • P[n ≔ S n])
   ⇒ ∀ n : ℕ • P
```

The "While" Rule — Induction for Partial Correctness

$$P[m := 0] \Rightarrow (\forall\, m : \mathbb{N} \mid P \bullet P[m := \mathsf{suc}\, m]) \Rightarrow (\forall\, m : \mathbb{N} \bullet P)$$

```
          `B ∧ Q  ⇒[ C ]  Q`
 ⊢─────────────────────────────────────
     `Q  ⇒[ while B do C od ]  ¬ B ∧ Q`
```

Relational Semantics of Imperative Programs

- Imperative programs, such as Cmd, transform a State that assigns values to variables.
- Program execution induces a **state transformation relation**.

```
Axiom “Definition of `State`”: State = Var → Value
Declaration: eval: State → ExprV → Value
Declaration: sat: Expr𝔹 → set State

Declaration: ⟦_⟧ : Cmd → (State ↔ State)

Axiom “Semantics of ;”:  ⟦ C1 ; C2 ⟧ = ⟦ C1 ⟧ ; ⟦ C2 ⟧
Axiom “Semantics of `if`”:
   ⟦ if B then C1 else C2 fi ⟧ = (sat B ◁ ⟦ C1 ⟧) ∪ (sat B ⩤ ⟦ C2 ⟧)

Axiom “Semantics of `while`”:
   ⟦ while B do C od ⟧ = (sat B ◁ ⟦ C ⟧) * ⩥ sat B
```

Informal sketch

```
Theorem “Partial Correctness”:  P ⇒⁅ C ⁆ Q  ≡   ⟦ C ⟧ ⦇ sat P ⦈ ⊆ sat Q
```