Discrete Mathematics with Applications I

COMPSCI&SFWRENG 2DM3

McMaster University, Fall 2019

Wolfram Kahl

2019-11-01

Plan for Today

- **Sequences**
  - Induction proofs, quantified theorem statements
- **Command Correctness**
  - Conditional statements
- **Textbook Chapter 11: Set Theory**

Sequences

- We consider the type Seq A of sequences with elements of type A as generated inductively by the following two constructors:

ϵ	:	$\mathsf{Seq}\ A$	`\eps`	empty sequence
$\_\triangleleft\_$	:	$A \to \mathsf{Seq}\ A \to \mathsf{Seq}\ A$	`\cons`	"cons"

  $\triangleleft$ associates to the right.

- Therefore: $[33, 22, 11] = 33 \triangleleft [22, 11] = 33 \triangleleft 22 \triangleleft [11] = 33 \triangleleft 22 \triangleleft 11 \triangleleft \epsilon$

- Appending single elements "at the end":

  $\_\triangleright\_ \ : \ \mathsf{Seq}\ A \to A \to \mathsf{Seq}\ A$ `\snoc` "snoc"

  $\triangleright$ associates to the left.

- (Con-)catenation:

  $\_\frown\_ \ : \ \mathsf{Seq}\ A \to \mathsf{Seq}\ A \to \mathsf{Seq}\ A$ `\catenate`

  $\frown$ associates to the right.

Subsequences

```
Axiom (13.25) “Empty subsequence”: ϵ ⊆ ys
Axiom (13.26) “Subsequence” “Cons is not a subsequence of ϵ”: ¬ (x ◃ xs ⊆ ϵ)
Axiom (13.27) “Subsequence anchored at head”: x ◃ ys ⊆ x ◃ zs  ≡  ys ⊆ zs
Axiom (13.28) “Subsequence without head”: x ≠ y  ⇒  (x ◃ xs ⊆ y ◃ ys  ≡  x ◃ xs ⊆ ys)
```

Prefixes and Segments — “Contiguous Subsequences”

```
Axiom (13.36) “Empty prefix”:
    isprefix ϵ xs
Axiom (13.37) “Not Prefix” “Cons is not prefix of ϵ”:
    isprefix (x ◃ xs) ϵ ≡ false
Axiom (13.38) “Prefix” “Cons prefix”:
    isprefix (x ◃ xs) (y ◃ ys) = x = y ∧ isprefix xs ys


Axiom (13.39) “Segment” “Segment of ϵ”: isseg xs ϵ  ≡  xs = ϵ
Axiom (13.40) “Segment” “Segment of ◃”:
  isseg xs (y ◃ ys)  ≡  isprefix xs (y ◃ ys) ∨ isseg xs ys
```

Sequences — Induction Proofs

Induction principle for sequences:

- if $P(\epsilon)$ — If P holds for ϵ
- and if $P(xs)$ implies $P(x \triangleleft xs)$ **for all** $x : A$, — and whenever P holds for xs, it also holds for any $x \triangleleft xs$,
- then for all $xs : \mathsf{Seq}\ A$ we have $P(xs)$. — then P holds for all sequences over A.

An **induction proof** using this looks as follows:

Theorem: P

Proof:

  By induction on $xs : \mathsf{Seq}\ A$**:**

    Base case:

      Proof for $P[xs := \epsilon]$

    Induction step:

      Proof for $(\forall x : A \bullet P[xs := x \triangleleft xs])$

        using **Induction hypothesis** P

(13.7) Tail is different: $x \triangleleft xs \neq xs$

(13.7) Tail is different: $\forall\, xs : \mathsf{Seq}\, A \bullet \forall\, x : A \bullet x \triangleleft xs \neq xs$

Precondition-Postcondition Specifications in Dynamix Logic Notation

- Program correctness statement in LADM (and much current use):
$$\{\, P \,\}\; C \;\{\, Q \,\}$$
This is called a "Hoare triple".
- **Meaning:** If command C is started in a state in which the **precondition** P holds then it will terminate in a state in which the **postcondition** Q holds.
- **Dynamic logic** notation (used in CalcCheck):
$$P \Rightarrow\!\![\; C \;]\; Q$$
- **Assignment Axiom:** $\{\, Q[x := E] \,\}\; x := E \;\{\, Q \,\}$ $\qquad Q[x := E] \Rightarrow\!\![\; x := E \;]\; Q$
- **Sequential composition:**

```
Primitive inference rule “Sequence”:
    `P  ⇒[ C₁ ]  Q`,   `Q  ⇒[ C₂ ]  R`
 ⊢───────────────────────────────────
       `P  ⇒[ C₁ ; C₂ ]  R`
```

Transitivity Rules for Calculational Command Correctness Reasoning

Primitive inference rule "Sequence":

$$\vdash \frac{`P \Rrightarrow[\ C_1\ ]\ Q`, \quad `Q \Rrightarrow[\ C_2\ ]\ R`}{`P \Rrightarrow[\ C_1\ ;\ C_2\ ]\ R`}$$

Strengthening the precondition:

$$\vdash \frac{`P_1 \Rightarrow P_2`, \quad `P_2 \Rrightarrow[\ C\ ]\ Q`}{`P_1 \Rrightarrow[\ C\ ]\ Q`}$$

Weakening the postcondition:

$$\vdash \frac{`P \Rrightarrow[\ C\ ]\ Q_1`, \quad `Q_1 \Rightarrow Q_2`}{`P \Rrightarrow[\ C\ ]\ Q_2`}$$

- Activated as transitivity rules
- Therefore used implicitly in calculations, e.g., proving $P \Rrightarrow[\ C_1 ⨾ C_2\ ]\ R$ to the right
- No need to refer to these rules explicitly.

$$\begin{array}{ll}
 & P \\
\Rrightarrow[\ C_1\ ] & \langle \ldots \rangle \\
 & Q \\
\Rightarrow & \langle \ldots \rangle \\
 & Q' \\
\Rrightarrow[\ C_2\ ] & \langle \ldots \rangle \\
 & R
\end{array}$$

Using converse operator for backward presentation:

$\_[\_] \Lleftarrow \_$

Fact: $x = 5 \Rrightarrow[\ (y := x + 1\ ;\ x := y + y)\ ]\ x = 12$

Proof:

$$\begin{array}{ll}
 & x = 12 \\
[\ x := y + y\ ] \Lleftarrow & \langle \text{“Assignment” with Substitution} \rangle \\
 & y + y = 12 \\
\equiv & \langle \text{“Identity of } \cdot \text{”} \rangle \\
 & 1 \cdot y + 1 \cdot y = 12 \\
\equiv & \langle \text{“Distributivity of } \cdot \text{ over } + \text{”} \rangle \\
 & (1 + 1) \cdot y = 12 \\
\equiv & \langle \text{Evaluation} \rangle \\
 & 2 \cdot y = 2 \cdot 6 \\
\equiv & \langle \text{“Cancellation of } \cdot \text{” with Fact } `2 \neq 0` \rangle \\
 & y = 6 \\
[\ y := x + 1\ ] \Lleftarrow & \langle \text{“Assignment” with Substitution} \rangle \\
 & x + 1 = 6 \\
\equiv & \langle \text{Fact } `5 + 1 = 6` \rangle \\
 & x + 1 = 5 + 1 \\
\equiv & \langle \text{“Cancellation of } + \text{”} \rangle \\
 & x = 5
\end{array}$$

Conditional Rule

Primitive inference rule "Conditional":

$$\vdash \frac{`B \wedge P \Rrightarrow[\ C_1\ ]\ Q`, \quad `\neg B \wedge P \Rrightarrow[\ C_2\ ]\ Q`}{`P \Rrightarrow[\ \text{if } B \text{ then } C_1 \text{ else } C_2\ ]\ Q`}$$

The Language of Set Theory — Overview

- The type $set(t)$ of sets with elements of type t
- Set membership: for $e : t$ and $S : set(t)$: $\quad e \in S$
- Set enumeration: $\{6, 7, 9\}$
- Set size: $\#\{6, 7, 9\} = 3$
- Set inclusion: $\subset, \subseteq, \supset, \supseteq$
- Set union and intersection: $\cup, \cap$
- Set difference: $S - T$ $\qquad$ Set complement: $\sim S$
- Power set (set of subsets): $\mathbb{P}\, S$
- Cartesian product (cross product, direct product) of sets: $S \times T$ $\qquad$ (Section 14.1)

Set Membership versus Type Annotation

Let T be a **type**; let S be a **set**, that is, an expression of type $set(T)$,
and let e be an expression ot type T, then

- $e \in S$ is an expression
- of type $\mathbb{B}$
- and denotes "e is **in** S"

  or "e is an **element of** S"

Because: $\_\in\_ : T \to set(T) \to \mathbb{B}$

Example, considering $\mathbb{N}$ as a subset of $\mathbb{Z}$:
(8.2) $\quad i \in \mathbb{N} \quad \Rightarrow \quad -i \leq 0$

Note:

- $e : T$ is nothing but the expression e, with type annotation T.
- If e has type T, then $e : T$ has the same value as e.
- If e has type T, then $e \in T$ evaluates to *true* in all states in which e is well-defined — **using the type T as a set**

The Axioms of Set Theory — Overview

(11.2e) **Membership in Set Enumerations:**

$$v \in \{e_1, \ldots, e_n\} \quad \equiv \quad v = e_1 \lor \cdots \lor v = e_n$$

(11.2f) **Empty Set:** $v \in \{\} \equiv false$

(11.4) **Axiom, Extensionality:** Provided $\neg occurs(\text{'}x\text{'}, \text{'}S, T\text{'})$,

$$S = T \quad \equiv \quad (\forall x \bullet x \in S \equiv x \in T)$$

(11.13T) **Axiom, Subset:** Provided $\neg occurs(\text{'}x\text{'}, \text{'}S, T\text{'})$,

$$S \subseteq T \quad \equiv \quad (\forall x \bullet x \in S \Rightarrow x \in T)$$

(11.14) **Axiom, Proper subset:** $\quad S \subset T \quad \equiv \quad S \subseteq T \land S \neq T$

(11.20) **Axiom, Union:** $\quad v \in S \cup T \quad \equiv \quad v \in S \lor v \in T$

(11.21) **Axiom, Intersection:** $\quad v \in S \cap T \quad \equiv \quad v \in S \land v \in T$

(11.22) **Axiom, Set difference:** $\quad v \in S - T \quad \equiv \quad v \in S \land v \notin T$

(11.23) **Axiom, Power set:** $\quad v \in \mathbb{P}\, S \quad \equiv \quad v \subseteq S$