Discrete Mathematics with Applications I

COMPSCI&SFWRENG 2DM3

McMaster University, Fall 2019

Wolfram Kahl

2019-10-30

What does it mean?

- $\_\sqsubseteq\_$ is an order
- s is a state
- P is valid
- P is a sentence
- $P \Rrightarrow[\, C \,]\, Q$

Plan for Today

- Textbook Section 8.1: **Typing**
- **Sequences** (Textbook Chapter 13)
  - Inductive view from empty sequence (ϵ) and "cons" ($\triangleleft$)

Types

A **type** **denotes a set of values** that

- can be associated with a variable
- an expression might evaluate to

Some basic types: $\mathbb{B}$, $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$

Every expression has a type.

"E : t" means: "Expression E is declared to have type t".

Examples:

- constants: $\mathit{true} : \mathbb{B}$, $\pi : \mathbb{R}$, $2 : \mathbb{Z}$, $2 : \mathbb{N}$
- variable declarations: $p : \mathbb{B}$, $k : \mathbb{N}$, $d : \mathbb{R}$
- type annotations in expressions:
  - $(x + y) \cdot x \quad \longrightarrow \quad (x : \mathbb{N} + y) \cdot x$
  - $(x + y) \cdot x \quad \longrightarrow \quad (((x : \mathbb{N} + y : \mathbb{N}) : \mathbb{N}) \cdot x : \mathbb{N}) : \mathbb{N}$

Function Types — Textbook Version

- If the parameters of function f have types $t_1, \ldots, t_n$
- and the result has type r,
- then f has type $t_1 \times \cdots \times t_n \to r$

We write: $\boxed{f : t_1 \times \cdots \times t_n \to r}$

Examples: $\neg\_ : \mathbb{B} \to \mathbb{B}$ $\quad$ $\_+\_ : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ $\quad$ $\_<\_ : \mathbb{Z} \times \mathbb{Z} \to \mathbb{B}$

Forming expressions using $\_<\_ : \mathbb{Z} \times \mathbb{Z} \to \mathbb{B}$:

- if expression a_1 has type $\mathbb{Z}$, and a_2 has type $\mathbb{Z}$
- then $a_1 < a_2$ is a (well-typed) expression
- and has type $\mathbb{B}$.

In general: For $f : t_1 \times \cdots \times t_n \to r$,

- if expression a_1 has type t_1, and ..., and a_n has type t_n
- then function application $f(a_1, \ldots, a_n)$ is an expression
- and has type r.

Function Types — Mechanised Mathematics Version

- If the parameters of function f have types $t_1, \ldots, t_n$
- and the result has type r,
- then f has type $t_1 \to \cdots \to t_n \to r$

We write: $\boxed{f : t_1 \to \cdots \to t_n \to r}$

The function type constructor $\to$ **associates to the right!**

Examples: $\neg : \mathbb{B} \to \mathbb{B}$ $\quad$ $\_+\_ : \mathbb{Z} \to \mathbb{Z} \to \mathbb{Z}$ $\quad$ $\_<\_ : \mathbb{Z} \to \mathbb{Z} \to \mathbb{B}$

Forming expressions using $\_<\_ : \mathbb{Z} \to \mathbb{Z} \to \mathbb{B}$:

- if expression a_1 has type $\mathbb{Z}$, and a_2 has type $\mathbb{Z}$
- then $a_1 < a_2$ is a (well-typed) expression
- and has type $\mathbb{B}$.

In general: For $f : t \to r$,

- if expression a has type t,
- then function application $f\ a$ is an expression
- and has type r.

Modeling English Propositions

- The number x is less than y or z.
  - Obvious assumptions: $$\begin{array}{lcl} x & : & \mathbb{Z} \\ y & : & \mathbb{Z} \\ z & : & \mathbb{Z} \end{array}$$
  - Since $\_{<}\_ : \mathbb{Z} \to \mathbb{Z} \to \mathbb{B}$: $$(x < y) : \mathbb{B}$$ $$(x < z) : \mathbb{B}$$
  - Since $\_{\vee}\_ : \mathbb{B} \to \mathbb{B} \to \mathbb{B}$: $((x < y) \vee (x < z)) : \mathbb{B}$ $\checkmark$
  - Since $\_{\vee}\_ : \mathbb{B} \to \mathbb{B} \to \mathbb{B}$ and $y, z : \mathbb{Z}$:

The number x is less than y or z. $\quad x \quad < \quad (y \vee z)$

Type Error!

CALCCHECK does not yet flag all type errors. **:-(**

Function Application — Textbook Version

Consider function g defined by: (1.6) $\quad g(z) \; = \; 3 \cdot z + 6$

- Special **function application** syntax for argument that is identifier or constant:

$$g.z \; = \; 3 \cdot z + 6$$

- **Function application** via substitution: If

  (1.7) $\quad g.z := \; E$

  defines function g, then for any argument X:

$$g.X \; = \; E[z := X]$$

Variant for function application:

(1.8) **Leibniz:** $\qquad \dfrac{X = Y}{g.X = g.Y}$

Function Application — Mechanised Mathematics Version

Consider function g defined by: (1.6) $\quad g\ z \; = \; 3 \cdot z + 6$

- **Function application** is denoted by **juxtaposition** (“putting side by side”)
- **Lexical separation** for argument that is identifier or constant: **space required:**

$$h\ z \; = \; g\ (g\ z)$$

  Superfluous parentheses (e.g., “$h(z) \; = \; g(g(z))$”) are allowed, **ugly**, and bad style.
- Function application still has **higher precedence than anything but substitution**.
- As non-associative binary infix operator, function application **associates to the left:** If $f : \mathbb{Z} \to (\mathbb{Z} \to \mathbb{Z})$, then $f\ 2\ 3 = (f\ 2)\ 3$, and $f\ 2 \; : \; \mathbb{Z} \to \mathbb{Z}$
- Typing rule for function application:

$$\frac{f : A \to B \qquad x : A}{f\ x : B}$$

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:
$$\begin{aligned}[33, 22, 11] &= 33 \triangleleft [22, 11] \\ &= 33 \triangleleft 22 \triangleleft [11] \\ &= 33 \triangleleft 22 \triangleleft 11 \triangleleft \epsilon\end{aligned}$$
- 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.

Concatenation

```
Axiom (13.17) “Left-identity of ⌢”
              “Definition of ⌢ for ϵ”:          ϵ ⌢ ys = ys
Axiom (13.18) “Mutual associativity of ◃ with ⌢”
              “Definition of ⌢ for ◃”:          (x ◃ xs) ⌢ ys = x ◃ (xs ⌢ ys)
```

Membership

```
Axiom “Membership in ϵ”:  x ∈ ϵ        ≡ false
Axiom “Membership in ◃”:  x ∈ y ◃ ys   ≡  x = y ∨ x ∈ ys
```

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
```