Software Verification and Testing

Lecture Notes: Z II

Sequences

idea: sequences are a fundamental data structure in Z. They are used for modelling stacks, queues or lists

implementation: sequences are finite functions from the natural numbers

further use: we will use sequences to learn about inductive definitions and proofs by induction

Sequences

intuition:

- sequences over some set A are functions $f : \mathbb{N} \to A$
- finite sequences over A are functions $f : 1 \mathinner{..} n \to A$ for some n

formalisation:

- (first definition) $\mathrm{seq}\, X == \{s : \mathbb{N} \to X \mid \exists\, n : \mathbb{N} \bullet \mathrm{dom}\, s = 1 \mathinner{..} n\}$
- but types can be sharpened

sequences in Z: $\mathrm{seq}\, X == \{s : \mathbb{N} \mathrel{+\!\!\!\!\twoheadrightarrow} X \mid \exists\, n : \mathbb{N} \bullet \mathrm{dom}\, s = 1 \mathinner{..} n\}$

notation: we write

- $\langle\, \rangle$ for the empty sequence
- $\langle s, h, e, f, f, i, e, l, d \rangle$ when we explicitly enumerate a sequence

Sequences

concatenation:

$$
\begin{array}{l}
[X] \\
\hline
\_ \frown \_ : \operatorname{seq} X \times \operatorname{seq} X \rightarrow \operatorname{seq} X \\
\hline
\forall s, t : \operatorname{seq} X \bullet \\
\qquad \#(s \frown t) = \#s + \#t \\
\qquad \forall i : 1 \mathinner{\ldotp\ldotp} \#s \bullet (s \frown t)\ i = s\ i \\
\qquad \forall j : 1 \mathinner{\ldotp\ldotp} \#t \bullet (s \frown t)\ (j + \#s) = t\ j \\
\hline
\end{array}
$$

example: $\langle a, b, c \rangle \frown \langle d, e, f, g \rangle = \langle a, b, c, d, e, f, g \rangle$

Sequences

head:

$$
\begin{array}{l}
[X] \\
\hline
head : \operatorname{seq} X \pfun X \\
\hline
\forall s : \operatorname{seq} X \mid s \neq \langle\, \rangle \bullet head\ s = s\ 1
\end{array}
$$

example:

- $head\langle a, b, c\rangle = a$
- $head\langle\, \rangle$ is undefined

Sequences

tail:

$$
\begin{array}{l}
[X] \\
\hline
tail : \mathrm{seq}\, X \nrightarrow \mathrm{seq}\, X \\
\hline
\forall\, s : \mathrm{seq}\, X \mid s \neq \langle\,\rangle \bullet \\
\qquad \#\, tail\; s = \#s - 1 \\
\qquad \forall\, i : 1 \mathrel{..} \#s - 1 \bullet (tail\; s)\; i = s\; (i + 1)
\end{array}
$$

example:

- $tail\langle a, b, c\rangle = \langle b, c\rangle$
- $tail\langle\,\rangle$ is undefined

Sequences

restriction: $\_ \upharpoonright \_ : \operatorname{seq} X \times \mathbb{P}\, X \to \operatorname{seq} X$ is more difficult to define [cf. Using Z]

here: we only provide intuition

example: $\langle a, b, x, n, x, b, x, n, b, a \rangle \upharpoonright \{a, n\} = \langle a, n, n, a \rangle$

Sequences

special case: injective sequences are repetition-free

theorem: sequence s is repetition-free iff $\#s = \#\,\mathrm{ran}(s)$

Sequences

further operations:

- cons

$$[X]$$

$$\_ : \_ : X \times \operatorname{seq} X \to \operatorname{seq} X$$

$$\begin{array}{l} \forall\, x : X \bullet \forall\, s : \operatorname{seq} X \bullet \\ \qquad \#(x : s) = 1 + \#s \\ \qquad (x : s)\ 1 = x \\ \qquad \forall\, i : 2 \mathinner{\ldotp\ldotp} \#s + 1 \bullet (x : s)\ i = s\ (i - 1) \end{array}$$

- example: $a : \langle b, c \rangle = \langle a, b, c \rangle$

Sequences

further operations: (introduced by example)

- $front\langle a, b, c\rangle = \langle a, b\rangle$
- $last\langle a, b, c\rangle = c$

remark: a definition will be given as an exercise

Sequences

observation:

- cons looks more basic that concatenation; it builds a sequence stepwise from the empty sequence
- every sequence can be written as a term of the form $x_1 : (x_2 : \cdots : (s_n : \langle\,\rangle) \ldots)$
- so every sequence is either empty or a cons of some element and a sequence; but not both
- concatenation can be defined in terms of cons

$$\langle\,\rangle \frown s = s$$

$$(x : s) \frown t = x : (s \frown t)$$

- this has two steps
  1. the definition for the sequence constructor $\langle\,\rangle$
  2. the definition for the sequence constructor $:$ (cons)

Cons and Cat

property: $\langle x \rangle \frown s = (x : s)$

proof: $\langle x \rangle \frown s = (x : \langle\,\rangle) \frown s = x : (\langle\,\rangle \frown s) = (x : s)$

remark: therefore, obviously, cons can also be defined in terms of cat. . .

Inductive Definitions

a generalisation: the definition of $\frown$ as a function on sequences can be generalised to arbitrary functions from sequences:

inductive/recursive definitions: let c be a constant in set B and let $g : X \times B \rightarrow B$ be a function. Then the function $f : \operatorname{seq} X \rightarrow B$ is uniquely defined by

$$f\ \langle\,\rangle = c \qquad f\ (x : s) = g(x, f(s))$$

examples:

- length of a sequence $\quad \#\langle\,\rangle = 0 \qquad \#(x : s) = 1 + \#xs$
- reversion of a sequence $\quad rev\langle\,\rangle = \langle\,\rangle \qquad rev(x : s) = (rev\ s) \frown \langle x\rangle$

Inductive Definitions

remember: terms and formulae were also inductively defined

- first on some "atoms", i.e., constants or atomic formulae
- then with respect to function symbols as term constructors and logical operation symbols as formula constructors

remark: also the natural numbers can be defined inductively using the constructors

- 0 and $s : \mathbb{N} \rightarrow \mathbb{N}$
- 4, e.g., is represented by the term $s(s(s(s(0))))$
- addition can be defined inductively as

$$n + 0 = n \qquad n + s(m) = s(n + m)$$

Inductive Definitions

example: $s \upharpoonright \{x\}$ can be defined inductively by

$$\begin{aligned} \langle\,\rangle \upharpoonright \{x\} &= \langle\,\rangle \\ (y : s) \upharpoonright \{x\} &= \mathit{if}\ x = y\ \mathit{then}\ y : (s \upharpoonright \{x\})\ \mathit{else}\ s \upharpoonright \{x\} \end{aligned}$$

task: extend this definition to $s \upharpoonright \{x, y\}$

- why doesn't that work?
- how can we simulate this?

remark: this is not a specification, but an implementation problem. . .

Proofs by Induction

observation: many systems, data types and behaviours can be defined inductively

theorem: (principle of mathematical induction) Let $P(.)$ be a property of natural numbers. If $P(0)$ holds and $P(m)$ implies $P(m+1)$ for all $m \in \mathbb{N}$. Then $P(n)$ holds for all $n \in \mathbb{N}$

Proofs by Induction

example: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

proof:

- base case $\sum 0 = 0 = \frac{0 \cdot 1}{2}$
- induction hypothesis $\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$
- induction step:

$$\sum_{i=1}^{k+1} i = (k+1) + \sum_{i=1}^{k} i = (k+1) + \frac{k(k+1)}{2}$$
$$= \frac{2(k+1) + k(k+1)}{2} = \frac{(k+1)(k+2)}{2}$$

Proofs by Induction

theorem: (principle of structural induction) Let P be a property of sequences. If $P(\langle\,\rangle)$ holds and for all $x \in X$ and $s \in \operatorname{seq} X$ $P(s)$ implies $P(x : s)$, then $P(s)$ holds for all $s \in \operatorname{seq} X$

proof: by contradiction assume that $P(s)$ does not hold for all sequences s.
Then there must be a minimal sequence t (wrt length) such that $\neg P(t)$ holds.
t cannot be empty, since $P(\langle\,\rangle)$ holds. So $t = x : t'$.
But then, by contraposition, $\neg P(t')$ also holds.
This contradicts the minimality of t.

Proofs by Induction

example: $\#(s \frown t) = \#s + \#t$.

proof:

- base case: $\#(\langle\,\rangle \frown t) = \#t = 0 + \#t = \#\langle\,\rangle + \#t$
- induction hypothesis: $\#(s \frown t) = \#s + \#t$
- induction step:

$$\begin{aligned}
\#((x : s) \frown t) &= \#(x : (s \frown t)) \\
&= 1 + \#(s \frown t) \\
&= 1 + \#s + \#t \\
&= \#(x : s) + \#t
\end{aligned}$$

Proofs by Induction

example: $(s \frown t) \frown u = s \frown (t \frown u)$

proof:

- base case: $(\langle\,\rangle \frown t) \frown u = t \frown u = \langle\,\rangle \frown (t \frown u)$
- induction hypothesis: $(s \frown t) \frown u = s \frown (t \frown u)$
- induction step:

$$\begin{aligned} ((x : s) \frown t) \frown u &= (x : (s \frown t)) \frown u \\ &= x : ((s \frown t) \frown u) \\ &= x : (s \frown (t \frown u)) \\ &= (x : s) \frown (t \frown u)) \end{aligned}$$

Proofs by Induction

example: $\#s = \#\, rev\ s$

proof:

- base case: $\#\langle\,\rangle = 0 = \#\langle\,\rangle = \#\, rev\langle\,\rangle$
- induction hypothesis: $\#s = \#\, rev\ s$
- induction step:

$$\begin{aligned}
\#(x : s) &= 1 + \#s \\
&= 1 + \#\, rev\ s \\
&= \#\, rev\ s + \#\langle x \rangle \\
&= \#(rev\ s \frown \langle x \rangle) \\
&= \#\, rev(x : s)
\end{aligned}$$

Further properties

we have

$$
\begin{aligned}
head(x : s) &= x \\
tail(x : s) &= s \\
last\ s &= head\ rev\ s \\
front\ s &= tail\ rev\ s
\end{aligned}
$$

Induction and Verification

observation: many functions/data-types can be inductively defined (factorials, Fibonacci numbers, trees, formulae,. . .)

structural induction can be generalised from sequences to arbitrary inductively defined expressions

example: show that every term is either bracket-free or contains an even number of brackets. . .

induction and verification:

- reasoning about inductively defined properties requires inductive proofs
- inductive reasoning is often creative; assumptions must be strengthened or modified
- some properties cannot be proved in a straight way

Induction and Verification

theorem proving: a theorem prover is a tool that carries out mathematical proofs on a machine

- proofs in FOL can often be automated
- if the claim is a theorem of FOL, it can be detected
- if it is not a theorem, the prover may run forever

problem: induction is not part of FOL

solution: interactive theorem provers

- many simple inductive proofs can still be automated