Functional Programming and Interactive Theorem Proving

with Isabelle/HOL

Peter Lammich

School of Computer Science
The University of Manchester

2019-8-1

Introduction

Programs may have bugs.

Introduction

Programs may have bugs. These can have severe effects!

Introduction

Programs may have bugs. These can have severe effects!

Hunting bugs:

Introduction

Programs may have bugs. These can have severe effects!

Hunting bugs: Testing?

Introduction

Programs may have bugs. These can have severe effects!

Hunting bugs: Testing? Not guaranteed to find them all!

Introduction

Programs may have bugs. These can have severe effects!

Hunting bugs: Testing? Not guaranteed to find them all!

Mathematical proof that program is correct.

Introduction

Programs may have bugs. These can have severe effects!

Hunting bugs: Testing? Not guaranteed to find them all!

Mathematical proof that program is correct. Finds all bugs!

Introduction

Programs may have bugs. These can have severe effects!

Hunting bugs: Testing? Not guaranteed to find them all!

Mathematical proof that program is correct. Finds all bugs!

BUT: when done one paper, its likely to have errors in proof!

Introduction

Programs may have bugs. These can have severe effects!

Hunting bugs: Testing? Not guaranteed to find them all!

Mathematical proof that program is correct. Finds all bugs!

BUT: when done one paper, its likely to have errors in proof!

This lecture: Using a computer to check proofs

Material

The Theorem Prover Isabelle/HOL:
`https://isabelle.in.tum.de/`

Lecture Material:
`https://github.com/lammich/MCR_SS_2019_FunProgProve`

Material

The Theorem Prover Isabelle/HOL:
`https://isabelle.in.tum.de/`

Lecture Material:
`https://github.com/lammich/MCR_SS_2019_FunProgProve`

Download it and follow this lecture on your laptop!

Material

The Theorem Prover Isabelle/HOL:
`https://isabelle.in.tum.de/`

Lecture Material:
`https://github.com/lammich/MCR_SS_2019_FunProgProve`

Download it and follow this lecture on your laptop!

Relax and enjoy! There is no exam on this lecture!

Short Poll

Raise your hand if you

- Have ever written a computer program

Short Poll

Raise your hand if you

- Have ever written a computer program
  - in C, C++, Java, BASIC, PASCAL

Short Poll

Raise your hand if you

- Have ever written a computer program
    - in C, C++, Java, BASIC, PASCAL
    - in JavaScript

Short Poll

Raise your hand if you

- Have ever written a computer program
  - in C, C++, Java, BASIC, PASCAL
  - in JavaScript
  - in PHP, Python, Ruby, PERL

Short Poll

Raise your hand if you

- Have ever written a computer program
  - in C, C++, Java, BASIC, PASCAL
  - in JavaScript
  - in PHP, Python, Ruby, PERL
  - in Haskell, Scala, OCaml, SML, LISP, Scheme, F#

Short Poll

Raise your hand if you

- Have ever written a computer program
  - in C, C++, Java, BASIC, PASCAL
  - in JavaScript
  - in PHP, Python, Ruby, PERL
  - in Haskell, Scala, OCaml, SML, LISP, Scheme, F#
  - Others?

Short Poll

Raise your hand if you

- Have ever written a computer program
  - in C, C++, Java, BASIC, PASCAL
  - in JavaScript
  - in PHP, Python, Ruby, PERL
  - in Haskell, Scala, OCaml, SML, LISP, Scheme, F#
  - Others?
- Know what quicksort is

Short Poll

Raise your hand if you

- Have ever written a computer program
  - in C, C++, Java, BASIC, PASCAL
  - in JavaScript
  - in PHP, Python, Ruby, PERL
  - in Haskell, Scala, OCaml, SML, LISP, Scheme, F#
  - Others?
- Know what quicksort is
- Know what (structural) induction means

Short Poll

Raise your hand if you

- Have ever written a computer program
  - in C, C++, Java, BASIC, PASCAL
  - in JavaScript
  - in PHP, Python, Ruby, PERL
  - in Haskell, Scala, OCaml, SML, LISP, Scheme, F#
  - Others?
- Know what quicksort is
- Know what (structural) induction means
- Have ever used an interactive theorem prover

Lists

$[]$	Empty list
$a \# l$	List with first element a and then list l

Lists

$[]$	Empty list
$a \# l$	List with first element a and then list l
	Example: $1\#2\#3\#4\#[]$

Lists

$[]$	Empty list
$a \# l$	List with first element a and then list l
	Example: $1\#2\#3\#4\#[]$
	Notation: $[1, 2, 3, 4]$

Lists

$[]$ Empty list

$a \# l$ List with first element a and then list l

Example: $1\#2\#3\#4\#[]$

Notation: $[1, 2, 3, 4]$

l_1 @ l_2: concatenate two lists
Example: $[1, 2, 3]$ @ $[4, 5, 6] = [1, 2, 3, 4, 5, 6]$

Append @

How to define @ ?

Append @

How to define @ ? Using only [] and #?

Append @

How to define @ ? Using only [] and #?

Case distinction whether first list is empty:

$[] \ @ \ l_2 =$

Append @

How to define $@$? Using only $[]$ and $\#$?

Case distinction whether first list is empty:

$[] \mathbin{@} l_2 = l_2$

Append @

How to define @ ? Using only [] and #?

Case distinction whether first list is empty:

$[] \mathbin{@} l_2 = l_2$
$(x \# l_1) \mathbin{@} l_2 =$

Append @

How to define @ ? Using only [] and #?

Case distinction whether first list is empty:

$[] \mathbin{@} l_2 = l_2$
$(x \# l_1) \mathbin{@} l_2 = x \# (l_1 \mathbin{@} l_2)$

Append @

How to define @ ? Using only [] and #?

Case distinction whether first list is empty:

$[] \mathbin{@} l_2 = l_2$
$(x \# l_1) \mathbin{@} l_2 = x \# (l_1 \mathbin{@} l_2)$

Example:
$([1,2] \mathbin{@} [3])$

Append @

How to define @ ? Using only [] and #?

Case distinction whether first list is empty:

$[] \mathbin{@} l_2 = l_2$
$(x \mathbin{\#} l_1) \mathbin{@} l_2 = x \mathbin{\#} (l_1 \mathbin{@} l_2)$

Example:
$([1,2] \mathbin{@} [3]) \;\; = (1 \mathbin{\#} 2 \mathbin{\#} []) \mathbin{@} (3 \mathbin{\#} [])$

Append @

How to define $@$? Using only $[]$ and $\#$?

Case distinction whether first list is empty:

$[] \ @ \ l_2 = l_2$
$(x \ \# \ l_1) \ @ \ l_2 = x \ \# \ (l_1 \ @ \ l_2)$

Example:
$([1,2] \ @ \ [3]) \ = (1 \ \# \ 2 \ \# \ []) \ @ \ (3 \ \# \ [])$
$= 1 \ \# \ ((\ 2 \ \# \ []) \ @ \ (3 \ \# \ []))$

Append @

How to define @ ? Using only [] and #?

Case distinction whether first list is empty:

$[] \ @ \ l_2 = l_2$
$(x \ \# \ l_1) \ @ \ l_2 = x \ \# \ (l_1 \ @ \ l_2)$

Example:
$([1,2] \ @ \ [3]) \ = (1 \ \# \ 2 \ \# \ []) \ @ \ (3 \ \# \ [])$
$= 1 \ \# \ ((\ 2 \ \# \ []) \ @ \ (3 \ \# \ []))$
$= 1 \ \# \ 2 \ \# \ ([] \ @ \ (3 \ \# \ []))$

Append @

How to define @ ? Using only [] and #?

Case distinction whether first list is empty:

$[] \mathbin{@} l_2 = l_2$
$(x \# l_1) \mathbin{@} l_2 = x \# (l_1 \mathbin{@} l_2)$

Example:
$([1,2] \mathbin{@} [3]) = (1 \# 2 \# []) \mathbin{@} (3 \# [])$
$= 1 \# ((2 \# []) \mathbin{@} (3 \# []))$
$= 1 \# 2 \# ([] \mathbin{@} (3 \# []))$
$= 1 \# 2 \# 3 \# []$

Filter

Erase all elements not ≤ 4 from a list

Filter

Erase all elements not ≤ 4 from a list

*leq*4 [1,42,7,5,2,6,3]

Filter

Erase all elements not ≤ 4 from a list

$\mathit{leq}4$ [1,42,7,5,2,6,3] $=$ [1,2,3]

Filter

Erase all elements not ≤ 4 from a list

$\mathit{leq}4\ [1,42,7,5,2,6,3] = [1,2,3]$

$\mathit{leq}4\ [] =$

Filter

Erase all elements not ≤ 4 from a list

$\mathit{leq}4\ [1,42,7,5,2,6,3] = [1,2,3]$

$\mathit{leq}4\ [] = []$

Filter

Erase all elements $\text{not} \leq 4$ from a list

$\mathit{leq}4\ [1{,}42{,}7{,}5{,}2{,}6{,}3] = [1{,}2{,}3]$

$\mathit{leq}4\ [\,] = [\,]$
$\mathit{leq}4\ (x\#l) =$

Filter

Erase all elements not ≤ 4 from a list

$leq4\ [1,42,7,5,2,6,3] = [1,2,3]$

$leq4\ [] = []$
$leq4\ (x\#l) = if\ x \leq 4\ then\ x\ \#\ leq4\ l\ else\ leq4\ l$

Filter

Erase all elements not ≤ 4 from a list

$leq4$ [1,42,7,5,2,6,3] = [1,2,3]

$leq4$ [] = []
$leq4$ $(x\#l)$ = *if* $x \leq 4$ *then* x # $leq4$ l *else* $leq4$ l

$leq4$ [1, 42, 7, 5, 2, 6, 3]

Filter

Erase all elements not ≤ 4 from a list

$leq4$ [1,42,7,5,2,6,3] = [1,2,3]

$leq4\ [\,] = [\,]$
$leq4\ (x\#l) = if\ x{\leq}4\ then\ x\ \#\ leq4\ l\ else\ leq4\ l$

$leq4$ [1, 42, 7, 5, 2, 6, 3]
= 1 # $leq4$ [42, 7, 5, 2, 6, 3]

Filter

Erase all elements not ≤ 4 from a list

$\mathit{leq}4\ [1,42,7,5,2,6,3] = [1,2,3]$

$\mathit{leq}4\ [] = []$
$\mathit{leq}4\ (x\#l) = \mathit{if}\ x \leq 4\ \mathit{then}\ x \# \mathit{leq}4\ l\ \mathit{else}\ \mathit{leq}4\ l$

$\mathit{leq}4\ [1, 42, 7, 5, 2, 6, 3]$
$= 1 \# \mathit{leq}4\ [42, 7, 5, 2, 6, 3]$
$= 1 \# \mathit{leq}4\ [7, 5, 2, 6, 3]$

Filter

Erase all elements not ≤ 4 from a list

$leq4\ [1,42,7,5,2,6,3] = [1,2,3]$

$leq4\ [] = []$
$leq4\ (x\#l) = if\ x{\leq}4\ then\ x \# leq4\ l\ else\ leq4\ l$

$leq4\ [1, 42, 7, 5, 2, 6, 3]$
$= 1 \# leq4\ [42, 7, 5, 2, 6, 3]$
$= 1 \# leq4\ [7, 5, 2, 6, 3]$
$= 1 \# leq4\ [5, 2, 6, 3]$

Filter

Erase all elements not ≤ 4 from a list

$leq4\ [1,42,7,5,2,6,3] = [1,2,3]$

$leq4\ [] = []$
$leq4\ (x\#l) = \mathit{if}\ x \leq 4\ \mathit{then}\ x\ \#\ leq4\ l\ \mathit{else}\ leq4\ l$

$leq4\ [1, 42, 7, 5, 2, 6, 3]$
$= 1\ \#\ leq4\ [42, 7, 5, 2, 6, 3]$
$= 1\ \#\ leq4\ [7, 5, 2, 6, 3]$
$= 1\ \#\ leq4\ [5, 2, 6, 3]$
$= 1\ \#\ leq4\ [2, 6, 3]$

Filter

Erase all elements not ≤ 4 from a list

$\mathit{leq}4\ [1,42,7,5,2,6,3] = [1,2,3]$

$\mathit{leq}4\ [] = []$
$\mathit{leq}4\ (x\#l) = \mathit{if}\ x \leq 4\ \mathit{then}\ x\ \#\ \mathit{leq}4\ l\ \mathit{else}\ \mathit{leq}4\ l$

$\mathit{leq}4\ [1, 42, 7, 5, 2, 6, 3]$
$= 1\ \#\ \mathit{leq}4\ [42, 7, 5, 2, 6, 3]$
$= 1\ \#\ \mathit{leq}4\ [7, 5, 2, 6, 3]$
$= 1\ \#\ \mathit{leq}4\ [5, 2, 6, 3]$
$= 1\ \#\ \mathit{leq}4\ [2, 6, 3]$
$= 1\ \#\ 2\ \#\ \mathit{leq}4\ [6, 3]$

Filter

Erase all elements not ≤ 4 from a list

$leq4$ [1,42,7,5,2,6,3] = [1,2,3]

$leq4$ [] = []
$leq4\ (x\#l)$ = $if\ x \leq 4\ then\ x\ \#\ leq4\ l\ else\ leq4\ l$

$leq4$ [1, 42, 7, 5, 2, 6, 3]
= 1 # $leq4$ [42, 7, 5, 2, 6, 3]
= 1 # $leq4$ [7, 5, 2, 6, 3]
= 1 # $leq4$ [5, 2, 6, 3]
= 1 # $leq4$ [2, 6, 3]
= 1 # 2 # $leq4$ [6, 3]
= 1 # 2 # $leq4$ [3]

Filter

Erase all elements not ≤ 4 from a list

$leq4\ [1,42,7,5,2,6,3] \ = [1,2,3]$

$leq4\ [] \ = []$
$leq4\ (x\#l) \ = \mathit{if}\ x \leq 4\ \mathit{then}\ x \# leq4\ l\ \mathit{else}\ leq4\ l$

$leq4\ [1, 42, 7, 5, 2, 6, 3]$
$= 1 \# leq4\ [42, 7, 5, 2, 6, 3]$
$= 1 \# leq4\ [7, 5, 2, 6, 3]$
$= 1 \# leq4\ [5, 2, 6, 3]$
$= 1 \# leq4\ [2, 6, 3]$
$= 1 \# 2 \# leq4\ [6, 3]$
$= 1 \# 2 \# leq4\ [3]$
$= 1 \# 2 \# 3 \# leq4\ []$

Filter

Erase all elements not ≤ 4 from a list

$leq4\ [1,42,7,5,2,6,3] = [1,2,3]$

$leq4\ [\,] = [\,]$
$leq4\ (x\#l) = if\ x{\leq}4\ then\ x \# leq4\ l\ else\ leq4\ l$

$leq4\ [1, 42, 7, 5, 2, 6, 3]$
$= 1 \# leq4\ [42, 7, 5, 2, 6, 3]$
$= 1 \# leq4\ [7, 5, 2, 6, 3]$
$= 1 \# leq4\ [5, 2, 6, 3]$
$= 1 \# leq4\ [2, 6, 3]$
$= 1 \# 2 \# leq4\ [6, 3]$
$= 1 \# 2 \# leq4\ [3]$
$= 1 \# 2 \# 3 \# leq4\ [\,]$
$= 1 \# 2 \# 3 \# [\,]$

Filter

Condition as parameter to function

$\mathit{filter}\ P\ [] = []$
$\mathit{filter}\ P\ (x \# l) = (\textbf{if}\ P\ x\ \textbf{then}\ x \# \mathit{filter}\ P\ l\ \textbf{else}\ \mathit{filter}\ P\ l)$

Filter

Condition as parameter to function

$\mathit{filter}\ P\ [] = []$
$\mathit{filter}\ P\ (x \# l) = (\mathbf{if}\ P\ x\ \mathbf{then}\ x \# \mathit{filter}\ P\ l\ \mathbf{else}\ \mathit{filter}\ P\ l)$

$\mathit{filter}\ (\lambda x.\ x \leq 4)\ l$: Same as $\mathit{leq}4$

Filter

Condition as parameter to function

$\textit{filter}\ P\ [] = []$
$\textit{filter}\ P\ (x \# l) = (\textbf{if}\ P\ x\ \textbf{then}\ x \# \textit{filter}\ P\ l\ \textbf{else}\ \textit{filter}\ P\ l)$

$\textit{filter}\ (\lambda x.\ x \leq 4)\ l$: Same as $\textit{leq}4$

$\lambda x.\ x \leq 4$ is anonymous function, with parameter x .

`Demo.thy`

Functions

Count

$count\ l\ x$ How often does element x occur in list l

Count

$count\ l\ x$ How often does element x occur in list l

Example: $count\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 =$

Count

$count\ l\ x$ How often does element x occur in list l

Example: $count\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

Count

$\mathit{count}\ l\ x$ How often does element x occur in list l

Example: $\mathit{count}\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$\mathit{count}\ []\ x\ =$

Count

$count\ l\ x$ How often does element x occur in list l

Example: $count\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$count\ []\ x\ = 0$

Count

$count\ l\ x$ How often does element x occur in list l

Example: $count\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$count\ []\ x\ = 0$
$count\ (y \# l)\ x\ =$

Count

$count\ l\ x$ How often does element x occur in list l

Example: $count\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$count\ []\ x\ = 0$
$count\ (y \# l)\ x\ = \mathbf{if}\ x{=}y\ \mathbf{then}\ 1 + count\ l\ x\ \mathbf{else}\ count\ l\ x$

Count

count l x How often does element *x* occur in list *l*

Example: $count\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$count\ []\ x\ = 0$
$count\ (y \# l)\ x\ = \textbf{if}\ x{=}y\ \textbf{then}\ 1 + count\ l\ x\ \textbf{else}\ count\ l\ x$

$count\ [2, 2, 1, 2]\ 2$
$=$

Count

$count\ l\ x$ How often does element x occur in list l

Example: $count\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$count\ []\ x\ = 0$
$count\ (y \# l)\ x\ = \textbf{if}\ x{=}y\ \textbf{then}\ 1 + count\ l\ x\ \textbf{else}\ count\ l\ x$

$count\ [2, 2, 1, 2]\ 2$
$= 1 + count\ [2, 1, 2]\ 2$
$=$

Count

$\mathit{count}\ l\ x$ How often does element x occur in list l

Example: $\mathit{count}\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$\mathit{count}\ []\ x\ = 0$
$\mathit{count}\ (y \# l)\ x\ = \textbf{if}\ x{=}y\ \textbf{then}\ 1 + \mathit{count}\ l\ x\ \textbf{else}\ \mathit{count}\ l\ x$

$\mathit{count}\ [2, 2, 1, 2]\ 2$
$= 1 + \mathit{count}\ [2, 1, 2]\ 2$
$= 1 + 1 + \mathit{count}\ [1, 2]\ 2$
$=$

Count

$count\ l\ x$ How often does element x occur in list l

Example: $count\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$count\ []\ x\ = 0$
$count\ (y \# l)\ x\ = \textbf{if}\ x{=}y\ \textbf{then}\ 1 + count\ l\ x\ \textbf{else}\ count\ l\ x$

$count\ [2, 2, 1, 2]\ 2$
$= 1 + count\ [2, 1, 2]\ 2$
$= 1 + 1 + count\ [1, 2]\ 2$
$= 1 + 1 + count\ [2]\ 2$
$=$

Count

$\mathit{count}\ l\ x$ How often does element x occur in list l

Example: $\mathit{count}\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$\mathit{count}\ []\ x\ = 0$
$\mathit{count}\ (y \# l)\ x\ = \textbf{if}\ x{=}y\ \textbf{then}\ 1 + \mathit{count}\ l\ x\ \textbf{else}\ \mathit{count}\ l\ x$

$\mathit{count}\ [2, 2, 1, 2]\ 2$
$= 1 + \mathit{count}\ [2, 1, 2]\ 2$
$= 1 + 1 + \mathit{count}\ [1, 2]\ 2$
$= 1 + 1 + \mathit{count}\ [2]\ 2$
$= 1 + 1 + 1 + \mathit{count}\ []\ 2$
$=$

Count

$count\ l\ x$ How often does element x occur in list l

Example: $count\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$count\ []\ x\ = 0$
$count\ (y \# l)\ x\ = \textbf{if}\ x{=}y\ \textbf{then}\ 1 + count\ l\ x\ \textbf{else}\ count\ l\ x$

$count\ [2, 2, 1, 2]\ 2$
$= 1 + count\ [2, 1, 2]\ 2$
$= 1 + 1 + count\ [1, 2]\ 2$
$= 1 + 1 + count\ [2]\ 2$
$= 1 + 1 + 1 + count\ []\ 2$
$= 1 + 1 + 1 + 0$
$=$

Count

$count\ l\ x$ How often does element x occur in list l

Example: $count\ [1, 2, 3, 1, 2, 3, 2, 2]\ 2 = 4$

$count\ []\ x\ = 0$
$count\ (y \# l)\ x\ = \textbf{if}\ x{=}y\ \textbf{then}\ 1 + count\ l\ x\ \textbf{else}\ count\ l\ x$

$count\ [2, 2, 1, 2]\ 2$
$= 1 + count\ [2, 1, 2]\ 2$
$= 1 + 1 + count\ [1, 2]\ 2$
$= 1 + 1 + count\ [2]\ 2$
$= 1 + 1 + 1 + count\ []\ 2$
$= 1 + 1 + 1 + 0$
$= 3$

Sortedness

Is a list sorted? E.g. $\textit{sorted}\ [1, 2, 4] = \textit{True}$, $\textit{sorted}\ [1, 4, 3] = \textit{False}$

Sortedness

Is a list sorted? E.g. $\mathit{sorted}\ [1, 2, 4] = \mathit{True}$, $\mathit{sorted}\ [1, 4, 3] = \mathit{False}$

$\mathit{sorted}\ [\,] \ =$

Sortedness

Is a list sorted? E.g. $\mathit{sorted}\ [1, 2, 4] = \mathit{True}$, $\mathit{sorted}\ [1, 4, 3] = \mathit{False}$

$\mathit{sorted}\ []\ = \mathit{True}$

Sortedness

Is a list sorted? E.g. $\mathit{sorted}\ [1, 2, 4] = \mathit{True}$, $\mathit{sorted}\ [1, 4, 3] = \mathit{False}$

$\mathit{sorted}\ [\,] \ = \mathit{True}$
$\mathit{sorted}\ [x] \ =$

Sortedness

Is a list sorted? E.g. $\mathit{sorted}\ [1, 2, 4] = \mathit{True}$, $\mathit{sorted}\ [1, 4, 3] = \mathit{False}$

$\mathit{sorted}\ [] \ = \mathit{True}$
$\mathit{sorted}\ [x] \ = \mathit{True}$

Sortedness

Is a list sorted? E.g. $\textit{sorted}\ [1, 2, 4] = \textit{True}$, $\textit{sorted}\ [1, 4, 3] = \textit{False}$

$\textit{sorted}\ [] \ = \textit{True}$
$\textit{sorted}\ [x] \ = \textit{True}$
$\textit{sorted}\ (x \# y \# l) \ =$

Sortedness

Is a list sorted? E.g. $\textit{sorted}\ [1, 2, 4] = \textit{True}$, $\textit{sorted}\ [1, 4, 3] = \textit{False}$

$$
\begin{aligned}
&\textit{sorted}\ [\,] = \textit{True}\\
&\textit{sorted}\ [x] = \textit{True}\\
&\textit{sorted}\ (x \mathbin{\#} y \mathbin{\#} l) = x \le y \wedge \textit{sorted}\ (y \mathbin{\#} l)
\end{aligned}
$$

Sortedness

Is a list sorted? E.g. $\textit{sorted}\ [1, 2, 4] = \textit{True}$, $\textit{sorted}\ [1, 4, 3] = \textit{False}$

$$
\begin{aligned}
&\textit{sorted}\ [] \ = \textit{True} \\
&\textit{sorted}\ [x] \ = \textit{True} \\
&\textit{sorted}\ (x \# y \# l) \ = x \leq y \wedge \textit{sorted}\ (y \# l)
\end{aligned}
$$

Note $\wedge$ means "and".

`Demo.thy`

Count and Sortedness

Quicksort

Algorithm to sort a list

Quicksort

Algorithm to sort a list

Idea:

$$qs\ (p \mathbin{\#} l) = qs\ (\text{elements} \le p)\ @\ [p]\ @\ qs\ (\text{elements} > p)$$

Quicksort

Algorithm to sort a list

Idea:

$$qs\ (p \# l) = qs\ (\text{elements} \leq p) \mathbin{@} [p] \mathbin{@} qs\ (\text{elements} > p)$$

$qs\ [3, 2, 5, 4, 7]$
$=$

Quicksort

Algorithm to sort a list

Idea:

$$qs\ (p \# l) = qs\ (\text{elements} \leq p)\ @\ [p]\ @\ qs\ (\text{elements} > p)$$

$qs\ [3, 2, 5, 4, 7]$
$=qs\ [2]\ @\ [3]\ @\ qs\ [5, 4, 7]$
$=$

Quicksort

Algorithm to sort a list

Idea:

$$qs\ (p \# l) = qs\ (\text{elements} \leq p) \mathbin{@} [p] \mathbin{@} qs\ (\text{elements} > p)$$

$\mathsf{qs}\ [3, 2, 5, 4, 7]$
$= \mathsf{qs}\ [2] \mathbin{@} [3] \mathbin{@} \mathsf{qs}\ [5, 4, 7]$
$= [2] \mathbin{@} [3] \mathbin{@} \mathsf{qs}\ [4] \mathbin{@} [5] \mathbin{@} \mathsf{qs}\ [7]$
$=$

Quicksort

Algorithm to sort a list

Idea:

$$qs\ (p \# l) = qs\ (\text{elements} \leq p)\ @\ [p]\ @\ qs\ (\text{elements} > p)$$

$\mathsf{qs}\ [3, 2, 5, 4, 7]$
$= \mathsf{qs}\ [2]\ @\ [3]\ @\ \mathsf{qs}\ [5, 4, 7]$
$= [2]\ @\ [3]\ @\ \mathsf{qs}\ [4]\ @\ [5]\ @\ \mathsf{qs}\ [7]$
$= [2]\ @\ [3]\ @\ [4]\ @\ [5]\ @\ [7]$
$=$

Quicksort

Algorithm to sort a list

Idea:

$$qs\ (p \# l) = qs\ (\text{elements} \leq p) \mathbin{@} [p] \mathbin{@} qs\ (\text{elements} > p)$$

$\mathsf{qs}\ [3, 2, 5, 4, 7]$
$= \mathsf{qs}\ [2] \mathbin{@} [3] \mathbin{@} \mathsf{qs}\ [5, 4, 7]$
$= [2] \mathbin{@} [3] \mathbin{@} \mathsf{qs}\ [4] \mathbin{@} [5] \mathbin{@} \mathsf{qs}\ [7]$
$= [2] \mathbin{@} [3] \mathbin{@} [4] \mathbin{@} [5] \mathbin{@} [7]$
$= [2, 3, 4, 5, 7]$

Quicksort

Algorithm to sort a list

Idea:

$$qs\ (p \# l) = qs\ (\text{elements} \leq p)\ @\ [p]\ @\ qs\ (\text{elements} > p)$$

$\mathsf{qs}\ [3, 2, 5, 4, 7]$
$= \mathsf{qs}\ [2]\ @\ [3]\ @\ \mathsf{qs}\ [5, 4, 7]$
$= [2]\ @\ [3]\ @\ \mathsf{qs}\ [4]\ @\ [5]\ @\ \mathsf{qs}\ [7]$
$= [2]\ @\ [3]\ @\ [4]\ @\ [5]\ @\ [7]$
$= [2, 3, 4, 5, 7]$

$\mathsf{qs}\ [] = []$
$\mathsf{qs}\ (p \# l) = \mathsf{qs}\ (\mathsf{filter}\ (\lambda x.\ x \leq p)\ l)\ @\ [p]\ @\ \mathsf{qs}\ (\mathsf{filter}\ (\lambda x.\ x > p)\ l)$

`Demo.thy`

Quicksort

Correct Sorting

What does it mean that quicksort is correct?

Correct Sorting

What does it mean that quicksort is correct?

1. The resulting list is sorted: $\textit{sorted}\ (\textit{qs}\ l)$

Correct Sorting

What does it mean that quicksort is correct?

1. The resulting list is sorted: $\mathit{sorted}\ (\mathit{qs}\ l)$
2. and contains the same elements: $\forall x.\ \mathit{count}\ (\mathit{qs}\ l)\ x = \mathit{count}\ l\ x$

   $\forall x. \ldots$ means "for all x"

`Demo.thy`

Correctness of Sorting

Useful Properties

$\mathit{count}\ (l_1\ @\ l_2)\ x =$

Useful Properties

$count\ (l_1\ @\ l_2)\ x = count\ l_1\ x + count\ l_2\ x$

Useful Properties

$count\ (l_1\ @\ l_2)\ x = count\ l_1\ x + count\ l_2\ x$

$count\ (filter\ P\ l)\ x\ =$

Useful Properties

$$count\ (l_1\ @\ l_2)\ x = count\ l_1\ x + count\ l_2\ x$$

$$count\ (filter\ P\ l)\ x\ = if\ P\ x\ then\ count\ l\ x\ else\ 0$$

Useful Properties

$count\ (l_1\ @\ l_2)\ x = count\ l_1\ x + count\ l_2\ x$

$count\ (filter\ P\ l)\ x\ = if\ P\ x\ then\ count\ l\ x\ else\ 0$

$count\ (filter\ (\lambda x.\ x \leq p)\ l)\ x + count\ (filter\ (\lambda x.\ x > p)\ l)\ x\ =$

Useful Properties

$$count\ (l_1\ @\ l_2)\ x = count\ l_1\ x + count\ l_2\ x$$

$$count\ (filter\ P\ l)\ x\ = if\ P\ x\ then\ count\ l\ x\ else\ 0$$

$$count\ (filter\ (\lambda x.\ x \leq p)\ l)\ x + count\ (filter\ (\lambda x.\ x > p)\ l)\ x\ = count\ l\ x$$

Useful Properties

$$\mathit{count}\ (l_1 \ @\ l_2)\ x = \mathit{count}\ l_1\ x + \mathit{count}\ l_2\ x$$

$$\mathit{count}\ (\mathit{filter}\ P\ l)\ x\ = \mathit{if}\ P\ x\ \mathit{then}\ \mathit{count}\ l\ x\ \mathit{else}\ 0$$

$$\mathit{count}\ (\mathit{filter}\ (\lambda x.\ x \leq p)\ l)\ x + \mathit{count}\ (\mathit{filter}\ (\lambda x.\ x > p)\ l)\ x\ = \mathit{count}\ l\ x$$

partitioning preserves elements

`Demo.thy`

Useful Properties

Induction

To prove correctness of $qs\ l$ for all l:

- **base** show that $qs\ []$ is correct
- **step** show that $qs\ (p\#l)$ is correct,
  assuming recursive calls are already correct (IH)

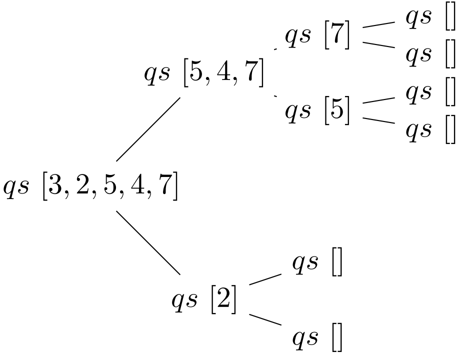
Induction

To prove correctness of *qs l* for all *l*:

base show that *qs* [] is correct

step show that *qs* (*p*#*l*) is correct,
assuming recursive calls are already correct (IH)

Idea: Justify corrrectness, starting at leaves of call tree:

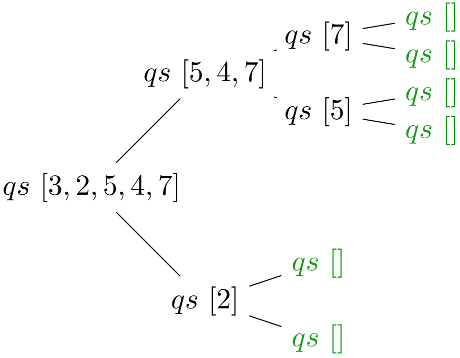


Induction

To prove correctness of *qs l* for all *l*:

base show that *qs* [] is correct
step show that *qs* (*p*#*l*) is correct,
assuming recursive calls are already correct (IH)

Idea: Justify corrrectness, starting at leaves of call tree:



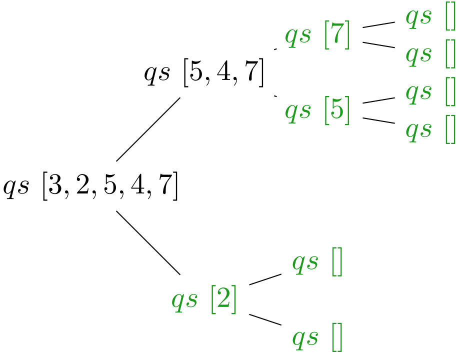
Induction

To prove correctness of *qs l* for all *l*:

base show that *qs* [] is correct

step show that *qs* (*p*#*l*) is correct,
assuming recursive calls are already correct (IH)

Idea: Justify corrrectness, starting at leaves of call tree:

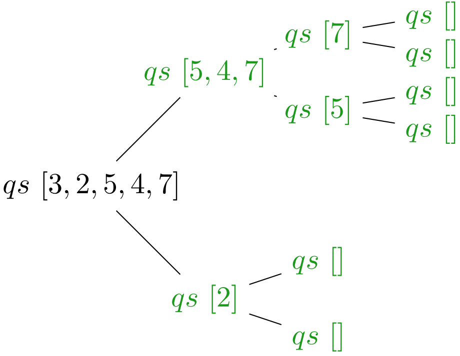


Induction

To prove correctness of *qs l* for all *l*:

- **base** show that *qs* [] is correct
- **step** show that *qs* (*p*#*l*) is correct,
  assuming recursive calls are already correct (IH)

Idea: Justify corrrectness, starting at leaves of call tree:

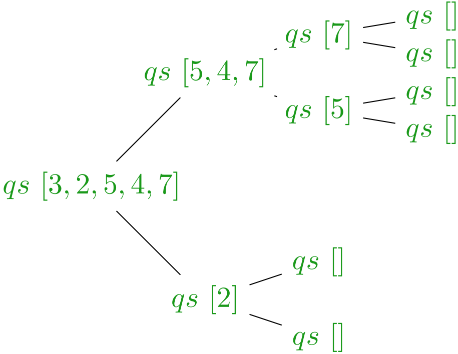


Induction

To prove correctness of *qs l* for all *l*:

- **base** show that *qs* [] is correct
- **step** show that *qs* (*p*#*l*) is correct,
  assuming recursive calls are already correct (IH)

Idea: Justify corrrectness, starting at leaves of call tree:



Element Preservation

$\mathit{count}\ (\mathit{qs}\ l)\ x = \mathit{count}\ l\ x$

Base: $\mathit{count}\ (\mathit{qs}\ [])\ x = \mathit{count}\ []\ x$

Element Preservation

$count\ (qs\ l)\ x = count\ l\ x$

Base: $count\ (qs\ [])\ x = count\ []\ x$

Step:

Element Preservation

$\mathit{count}\ (\mathit{qs}\ l)\ x = \mathit{count}\ l\ x$

Base: $\mathit{count}\ (\mathit{qs}\ [])\ x = \mathit{count}\ []\ x$

Step:

Let $l_1 = \mathit{filter}\ (\lambda x.\ x \leq p)\ l$ and $l_2 = \mathit{filter}\ (\lambda x.\ x > p)\ l$

Element Preservation

$count\ (qs\ l)\ x = count\ l\ x$

Base: $count\ (qs\ [])\ x = count\ []\ x$

Step:
Let $l_1 = filter\ (\lambda x.\ x \leq p)\ l$ and $l_2 = filter\ (\lambda x.\ x > p)\ l$
IH: $count\ (qs\ l_1)\ x = count\ l_1\ x$ and $count\ (qs\ l_2)\ x = count\ l_2\ x$

Element Preservation

$count\ (qs\ l)\ x = count\ l\ x$

Base: $count\ (qs\ [])\ x = count\ []\ x$

Step:
Let $l_1 = filter\ (\lambda x.\ x \leq p)\ l$ and $l_2 = filter\ (\lambda x.\ x > p)\ l$
IH: $count\ (qs\ l_1)\ x = count\ l_1\ x$ and $count\ (qs\ l_2)\ x = count\ l_2\ x$
Show: $count\ (qs\ (p\#l))\ x = count\ (p\#l)\ x$

Element Preservation

$count\ (qs\ l)\ x = count\ l\ x$

Base: $count\ (qs\ [])\ x = count\ []\ x$

Step:
Let $l_1 = filter\ (\lambda x.\ x \leq p)\ l$ and $l_2 = filter\ (\lambda x.\ x > p)\ l$
IH: $count\ (qs\ l_1)\ x = count\ l_1\ x$ and $count\ (qs\ l_2)\ x = count\ l_2\ x$
Show: $count\ (qs\ (p\#l))\ x = count\ (p\#l)\ x$

$$count\ (qs\ (p\#l))\ x$$

Element Preservation

$count\ (qs\ l)\ x = count\ l\ x$

Base: $count\ (qs\ [])\ x = count\ []\ x$

Step:
Let $l_1 = filter\ (\lambda x.\ x \le p)\ l$ and $l_2 = filter\ (\lambda x.\ x > p)\ l$
IH: $count\ (qs\ l_1)\ x = count\ l_1\ x$ and $count\ (qs\ l_2)\ x = count\ l_2\ x$
Show: $count\ (qs\ (p \# l))\ x = count\ (p \# l)\ x$

$$
\begin{aligned}
& count\ (qs\ (p \# l))\ x \\
= \ & count\ (qs\ l_1\ @\ [p]\ @\ qs\ l_2)\ x
\end{aligned}
$$

Element Preservation

$count\ (qs\ l)\ x = count\ l\ x$

Base: $count\ (qs\ [])\ x = count\ []\ x$

Step:
Let $l_1 = filter\ (\lambda x.\ x \leq p)\ l$ and $l_2 = filter\ (\lambda x.\ x > p)\ l$
IH: $count\ (qs\ l_1)\ x = count\ l_1\ x$ and $count\ (qs\ l_2)\ x = count\ l_2\ x$
Show: $count\ (qs\ (p\#l))\ x = count\ (p\#l)\ x$

$$
\begin{aligned}
& count\ (qs\ (p\#l))\ x \\
= \ & count\ (qs\ l_1\ @\ [p]\ @\ qs\ l_2)\ x \\
= \ & count\ [p]\ x + count\ (qs\ l_1)\ x + count\ (qs\ l_2)\ x
\end{aligned}
$$

Element Preservation

$count\ (qs\ l)\ x = count\ l\ x$

Base: $count\ (qs\ [])\ x = count\ []\ x$

Step:
Let $l_1 = filter\ (\lambda x.\ x \leq p)\ l$ and $l_2 = filter\ (\lambda x.\ x > p)\ l$
IH: $count\ (qs\ l_1)\ x = count\ l_1\ x$ and $count\ (qs\ l_2)\ x = count\ l_2\ x$
Show: $count\ (qs\ (p\#l))\ x = count\ (p\#l)\ x$

$$\begin{aligned} & count\ (qs\ (p\#l))\ x \\ = \ & count\ (qs\ l_1\ @\ [p]\ @\ qs\ l_2)\ x \\ = \ & count\ [p]\ x + count\ (qs\ l_1)\ x + count\ (qs\ l_2)\ x \\ = \ & count\ [p]\ x + count\ l_1\ x + count\ l_2\ x \quad \text{(IH)} \end{aligned}$$

Element Preservation

$count\ (qs\ l)\ x = count\ l\ x$

Base: $count\ (qs\ [])\ x = count\ []\ x$

Step:
Let $l_1 = filter\ (\lambda x.\ x \leq p)\ l$ and $l_2 = filter\ (\lambda x.\ x > p)\ l$
IH: $count\ (qs\ l_1)\ x = count\ l_1\ x$ and $count\ (qs\ l_2)\ x = count\ l_2\ x$
Show: $count\ (qs\ (p\#l))\ x = count\ (p\#l)\ x$

$$
\begin{aligned}
& count\ (qs\ (p\#l))\ x \\
= \;& count\ (qs\ l_1\ @\ [p]\ @\ qs\ l_2)\ x \\
= \;& count\ [p]\ x + count\ (qs\ l_1)\ x + count\ (qs\ l_2)\ x \\
= \;& count\ [p]\ x + count\ l_1\ x + count\ l_2\ x \quad \text{(IH)} \\
= \;& count\ [p]\ x + count\ l\ x
\end{aligned}
$$

Element Preservation

$count\ (qs\ l)\ x = count\ l\ x$

Base: $count\ (qs\ [])\ x = count\ []\ x$

Step:

Let $l_1 = filter\ (\lambda x.\ x \leq p)\ l$ and $l_2 = filter\ (\lambda x.\ x > p)\ l$

IH: $count\ (qs\ l_1)\ x = count\ l_1\ x$ and $count\ (qs\ l_2)\ x = count\ l_2\ x$

Show: $count\ (qs\ (p\#l))\ x = count\ (p\#l)\ x$

$$\begin{aligned}
& count\ (qs\ (p\#l))\ x \\
= \ & count\ (qs\ l_1\ @\ [p]\ @\ qs\ l_2)\ x \\
= \ & count\ [p]\ x + count\ (qs\ l_1)\ x + count\ (qs\ l_2)\ x \\
= \ & count\ [p]\ x + count\ l_1\ x + count\ l_2\ x \quad \text{(IH)} \\
= \ & count\ [p]\ x + count\ l\ x \\
= \ & count\ (p\#l)\ x
\end{aligned}$$

`Demo.thy`

Quicksort preserves Elements

Quicksort Sorts

Set of elements in list $(x \in \mathit{set}\ l) = (0 < \mathit{count}\ l\ x)$

Quicksort Sorts

Set of elements in list $(\mathit{x} \in \mathit{set}\ \mathit{l}) = (0 < \mathit{count}\ \mathit{l}\ \mathit{x})$

Obviously: $\mathit{set}\ (\mathit{qs}\ \mathit{l}) = \mathit{set}\ \mathit{l}$

Quicksort Sorts

Set of elements in list $(x \in set\ l) = (0 < count\ l\ x)$

Obviously: $set\ (qs\ l) = set\ l$

When is list l_1 @ $[p]$ @ l_2 sorted?

Quicksort Sorts

Set of elements in list $(x \in set\ l) = (0 < count\ l\ x)$

Obviously: $set\ (qs\ l) = set\ l$

When is list $l_1 \ @\ [p]\ @\ l_2$ sorted?

$sorted\ (l_1 \ @\ [p]\ @\ l_2)$ iff

Quicksort Sorts

Set of elements in list $(x \in \mathit{set}\ l) = (0 < \mathit{count}\ l\ x)$

Obviously: $\mathit{set}\ (\mathit{qs}\ l) = \mathit{set}\ l$

When is list $l_1\ @\ [p]\ @\ l_2$ sorted?

$\mathit{sorted}\ (l_1\ @\ [p]\ @\ l_2)$ iff
$\mathit{sorted}\ l_1 \wedge \mathit{sorted}\ l_2$

Quicksort Sorts

Set of elements in list $(x \in set\ l) = (0 < count\ l\ x)$

Obviously: $set\ (qs\ l) = set\ l$

When is list $l_1\ @\ [p]\ @\ l_2$ sorted?

$sorted\ (l_1\ @\ [p]\ @\ l_2)$ iff
$sorted\ l_1 \wedge sorted\ l_2$
and

Quicksort Sorts

Set of elements in list $(x \in set\ l) = (0 < count\ l\ x)$

Obviously: $set\ (qs\ l) = set\ l$

When is list $l_1\ @\ [p]\ @\ l_2$ sorted?

$sorted\ (l_1\ @\ [p]\ @\ l_2)$ iff
$sorted\ l_1 \wedge sorted\ l_2$
and $(\forall x{\in}set\ l_1.\ x \leq p) \wedge (\forall x{\in}set\ l_2.\ p \leq x)$

Quicksort Sorts

Set of elements in list $(x \in \mathit{set}\ l) = (0 < \mathit{count}\ l\ x)$

Obviously: $\mathit{set}\ (\mathit{qs}\ l) = \mathit{set}\ l$

When is list $l_1\ @\ [p]\ @\ l_2$ sorted?

$\mathit{sorted}\ (l_1\ @\ [p]\ @\ l_2)$ iff
$\mathit{sorted}\ l_1 \wedge \mathit{sorted}\ l_2$
and $(\forall x{\in}\mathit{set}\ l_1.\ x \leq p) \wedge (\forall x{\in}\mathit{set}\ l_2.\ p \leq x)$

What do we know about element x if $x \in \mathit{set}\ (\mathit{filter}\ P\ l)$?

Quicksort Sorts

Set of elements in list $(x \in set\ l) = (0 < count\ l\ x)$

Obviously: $set\ (qs\ l) = set\ l$

When is list $l_1\ @\ [p]\ @\ l_2$ sorted?

$sorted\ (l_1\ @\ [p]\ @\ l_2)$ iff
$sorted\ l_1 \wedge sorted\ l_2$
and $(\forall x \in set\ l_1.\ x \leq p) \wedge (\forall x \in set\ l_2.\ p \leq x)$

What do we know about element x if $x \in set\ (filter\ P\ l)$?

$x \in set\ (filter\ P\ l) \Longrightarrow P\ x$

`Demo.thy`

More useful Properties and Quicksort Sorts

Conclusions

Proved correct functional implementation of quicksort.

Conclusions

Proved correct functional implementation of quicksort.
Proof machine checked, using Isabelle/HOL.

Conclusions

Proved correct functional implementation of quicksort.
Proof machine checked, using Isabelle/HOL.

Further material:

Book: Concrete Semantics `http://www.concrete-semantics.org/`
Lecture: Semantics of PL
`http://www21.in.tum.de/teaching/semantik/WS1819/`

Conclusions

Proved correct functional implementation of quicksort.
Proof machine checked, using Isabelle/HOL.

Further material:

Book: Concrete Semantics `http://www.concrete-semantics.org/`
Lecture: Semantics of PL
`http://www21.in.tum.de/teaching/semantik/WS1819/`

Thanks!