侵害

Content

Rough timeline

- Intro & motivation, getting started
  - [1]
- Foundations & Principles
  - Lambda Calculus, natural deduction
    - [2, 3, 4]
  - Higher Order Logic
    - [5, 6, 7]
  - Term rewriting
    - [8, 9, 10]
- Proof & Specification Techniques
  - Isar
    - [11, 12]
  - Inductively defined sets, rule induction
    - [13, 14]
  - Datatypes, recursion, induction
    - [15, 16, 17, 18]
  - Calculational reasoning, mathematics style proofs
    - [19]
  - Hoare logic, proofs about programs
    - [20]
  - [21, 22, 23]

a1 out; a1 due; a2 out; a2 due; session break; a3 out; a3 due

Datatypes in Isar

Datatype case distinction

proof (cases term)
  case Constructor1
    ...
    next
    ...
  next
  case (Constructor2, x)
    ... 
    next
  qed

  case (Constructor, x) =
  fix x assume Constructor : "term = Constructor, x"

Slide 1

Slide 2

Slide 3

Slide 4
Structural induction for type nat

show $P \ n$
proof (induct $n$)
case 0
  = let ?case = $P \ 0$
...
show ?case
next
case (Suc $n$)
  = fix $n$ assume Suc: $P \ n$
    let ?case = $P \ (Suc \ n)$
    ... $n$ ...
show ?case
qed

Structural induction with $\Rightarrow$ and $\land$

show $\forall x. A \ n \Rightarrow P \ n$
proof (induct $n$)
case 0
  = fix $x$ assume 0: "$A \ 0$"
    let ?case = "$P \ 0$"
...
show ?case
next
case (Suc $n$)
  = fix $n$ and $x$
    assume Suc: $\forall x. A \ n \Rightarrow P \ n$
    "$A \ (Suc \ n)$"
    ... $n$ ...
    let ?case = "$P \ (Suc \ n)"$
  ... $n$ ...
show ?case
qed
We have seen today ...

- Datatypes in Isar
- Defining regular expressions as a data type
- Playing with recursion and induction