An introduction to recursion and induction

A recursive datatype: toy lists

datatype *'a list = Nil | Cons 'a ('a list)*

A recursive datatype: toy lists

datatype *’a list* = *Nil* | *Cons ’a (’a list)*

Nil**:** empty list

Cons x xs**:** head *x :: ’a*, tail *xs :: ’a list*

A recursive datatype: toy lists

datatype *'a list = Nil | Cons 'a ('a list)*

Nil**:** empty list

Cons x xs**:** head *x :: 'a*, tail *xs :: 'a list*

A toy list: *Cons False (Cons True Nil)*

A recursive datatype: toy lists

datatype *'a list = Nil | Cons 'a ('a list)*

***Nil*:** empty list

Cons x xs: head *x :: 'a*, tail *xs :: 'a list*

A toy list: *Cons False (Cons True Nil)*

Predefined lists: *[False, True]*

Structural induction on lists

P xs holds for all lists *xs* if

Structural induction on lists

P xs holds for all lists *xs* if

- *P Nil*

Structural induction on lists

$P\ xs$ holds for all lists xs if

- $P\ \mathit{Nil}$
- and for arbitrary x and xs, $P\ xs$ implies $P\ (\mathit{Cons}\ x\ xs)$

A recursive function: append

Definition by *primitive recursion*:

```
primrec app :: 'a list ⇒ 'a list ⇒ 'a list where
app Nil ys = ? |
app (Cons x xs) ys = ??
```

A recursive function: append

Definition by *primitive recursion*:

primrec *app :: 'a list ⇒ 'a list ⇒ 'a list* **where**
app Nil ys = ? |
app (Cons x xs) ys = ??

1 rule per constructor
Recursive calls must drop the constructor ⟹ Termination

Concrete syntax

In `.thy` files:

Types and formulas need to be inclosed in "

Concrete syntax

In `.thy` files:
Types and formulas need to be inclosed in "

Except for single identifiers, e.g. *'a*

Concrete syntax

In `.thy` files:

Types and formulas need to be inclosed in "

Except for single identifiers, e.g. *'a*

" normally not shown on slides

Demo: append and reverse

Proofs

General schema:

```
lemma name: "..."
apply (...)
apply (...)
⋮
done
```

If the lemma is suitable as a simplification rule:

```
lemma name[simp]: "..."
```

Proof methods

- Structural induction
  - Format: *(induct x)*
    x must be a free variable in the first subgoal.
    The type of *x* must be a datatype.
  - Effect: generates 1 new subgoal per constructor
- Simplification and a bit of logic
  - Format: *auto*
  - Effect: tries to solve as many subgoals as possible using simplification and basic logical reasoning.

Top down proofs

Command

sorry

“completes” any proof.

Top down proofs

Command

sorry

“completes” any proof.

Allows top down development:

Assume lemma first, prove it later.

Some useful tools

Disproving tools

Automatic counterexample search by random testing:

quickcheck

Disproving tools

Automatic counterexample search by random testing:

quickcheck

Counterexample search via SAT solver:

nitpick

Finding theorems

1. Click on Find button
2. Input search pattern (e.g. *"_ & True"*)

Demo: Disproving and Finding