Proving with Computer Assistance
Lecture 12

Inversion

Herman Geuvers

The natural number type

Recall the definition of natural numbers:

```
Inductive nat : Set := O : nat | S : nat -> nat.
```

Meaning of this definition:

- Every number has one of two forms:
  - it is the constructor `O` **or**
  - it is built by applying the constructor `S` to another number.
- But there is more to say, which is implicit in the definition:
  - The constructor `S` is injective: If `S n = S m`, then `n = m`
  - The constructors `O` and `S` are distinct: `O` is not equal to `S n` for any `n`.

General inductive types

Principles similar to `nat` apply to all inductively defined types:

- injectivity: the constructors are injective
- no overlap: the values built from distinct constructors are never equal.

For lists:

- `cons` is injective
- `nil` $\neq$ `cons a l` for every `a`, `l`

For booleans: `true` $\neq$ `false`

Inversion tactic

The inversion tactic is used to exploit injectivity and no overlap
Suppose

$$H : c\ a_1\ a_2 \ldots a_n = d\ b_1\ b_2 \ldots b_m$$

for constructors c and d and arguments $a_1 a_2 \ldots a_n$ and $b_1 b_2 \ldots b_m$. Then

$$\texttt{inversion}\ H$$

looks at the possible ways that this equation can arise:

- If c and d are the same constructor, then (by the injectivity) $a_1 = b_1$, $a_2 = b_2, \ldots$. These facts are added to the context, and can be use to rewrite the goal.
- If c and d are different constructors, then H is contradictory (the equality is false). So, the goal is provable! `inversion` H completes the goal.

See the examples in the Rocq files.

Curry-Howard-de Bruijn isomorphism

predicate is true

$\Updownarrow$

inductive type is inhabited

even natural numbers

```
Inductive even : nat -> Prop :=
  | even_O : even O
  | even_S_S : forall n, even n -> even (S (S n)).
```

even and odd natural numbers

```
Inductive even : nat -> Prop :=
  | even_O : even O
  | even_S : forall n, odd n -> even (S n)
with odd : nat -> Prop :=
    odd_S : forall n, even n -> odd (S n).
```

$n \leq m$

```
Inductive le (n:nat) : nat -> Prop :=
  | le_n : le n n
  | le_S : forall m:nat, le n m -> le n (S m).
```

sorted lists of natural numbers

```
Inductive Sorted : natlist -> Prop :=
  | Sorted_nil : Sorted nil
  | Sorted_one : forall n : nat, Sorted (cons n nil)
  | Sorted_cons :
      forall (l : natlist) (n m : nat),
        Sorted (cons m l) -> le n m ->
          Sorted (cons n (cons m l)).
```

Leibniz equality

```
Inductive eq (A : Set) (x : A) : A -> Prop :=
    refl_equal : eq A x x.
```

applying the induction principle

- `elim`
- `induction`

inversion

```
Lemma not_even_1 : ~(even 1).
```

- `inversion`

what does inversion do?

abstract explanation

elimination proves universally quantified properties

$$\forall x_1, \ldots, x_l.\, \mathtt{I}(x_1, \ldots, x_l) \Rightarrow P(x_1, \ldots, x_l)$$

- what to do with $\mathtt{I}(t_1, \ldots, t_l) \Rightarrow P(t_1, \ldots, t_l)$ when t_i are not variables?
- generalize to $\forall x_1, \ldots, x_l.\, \mathtt{I}(x_1, \ldots, x_l) \Rightarrow P(x_1, \ldots, x_l)$
- what to do if the generalization does not hold?
- find another one which works:

$$\forall x_1, \ldots, x_l.\, \mathtt{I}(x_1, \ldots, x_l) \Rightarrow \underbrace{(x_1, \ldots, x_l) = (t_1, \ldots, t_l) \Rightarrow P(t_1, \ldots, t_l)}$$

example

constructors

$$\texttt{even\_O} : \texttt{even}\,(0) \qquad \texttt{even\_S\_S} : \forall n.\,\texttt{even}\,(n) \Rightarrow \texttt{even}\,(n+2)$$

induction principe

$$\frac{P(0) \qquad \forall n.\,\texttt{even}\,(n) \Rightarrow P(n) \Rightarrow P(n+2)}{\forall n.\,\texttt{even}\,(n) \Rightarrow P(n)}$$

how to prove $\texttt{even}\,(1) \Rightarrow \bot$?

take: $P(n) = n = 1 \Rightarrow \bot$