Induction heuristics

Basic heuristics

Theorems about recursive functions are proved by induction

Basic heuristics

Theorems about recursive functions are proved by induction

Induction on argument number i of f
if f is defined by recursion on argument number i

A tail recursive reverse

primrec *itrev :: 'a list ⇒ 'a list ⇒ 'a list*

A tail recursive reverse

primrec *itrev :: 'a list* $\Rightarrow$ *'a list* $\Rightarrow$ *'a list* **where**

itrev [] ys = ys |

itrev (x#xs) ys =

A tail recursive reverse

primrec *itrev :: 'a list ⇒ 'a list ⇒ 'a list* **where**

itrev [] ys = ys |

itrev (x#xs) ys = itrev xs (x#ys)

A tail recursive reverse

primrec *itrev :: 'a list ⇒ 'a list ⇒ 'a list* **where**

itrev [] ys = ys |

itrev (x#xs) ys = itrev xs (x#ys)

lemma *itrev xs [] = rev xs*

A tail recursive reverse

primrec *itrev :: 'a list ⇒ 'a list ⇒ 'a list* **where**

itrev [] ys = ys |

itrev (x#xs) ys = itrev xs (x#ys)

lemma *itrev xs [] = rev xs*

Why in this direction?

A tail recursive reverse

primrec *itrev :: 'a list ⇒ 'a list ⇒ 'a list* **where**

itrev [] ys = ys |

itrev (x#xs) ys = itrev xs (x#ys)

lemma *itrev xs [] = rev xs*

Why in this direction?

Because the lhs is “more complex” than the rhs.

Demo

Generalisation

- Replace constants by variables

Generalisation

- Replace constants by variables
- Generalize free variables
  - by $\forall$ in formula
  - by *arbitrary* in induction proof