Outline of Lecture 9

1. Converting loops into recursive functions and accumulator passing style.
2. More generalizing theorems for induction
Iterative Reverse Algorithm

- The reverse function is inefficient because it uses the append function over and over again.
- The following iterative algorithm reverses a list in linear time (textbook page 317).

```plaintext
procedure iterative_reverse(list)
    xs = list
    ys = []
    while xs != []
        ys = hd(xs) # ys
        xs = tl(xs)
    return ys
```
The following \texttt{itrev} function is a recursive version of the iterative algorithm.

The trick is to add an extra parameter for each variable that gets updated in the for loop of the iterative algorithm.

\texttt{primrec itrev :: "'a list ⇒ 'a list ⇒ 'a list" where}
\begin{verbatim}
  "itrev [] ys = ys" |
  "itrev (x#xs) ys = itrev xs (x#ys)"
\end{verbatim}

\texttt{lemma "itrev [1,2,3] [] = [3,2,1]"}
\texttt{proof -}
\begin{verbatim}
  have "itrev [1,2,3] [] = itrev [2,3] [1]" by simp
  also have "... = itrev [3] [2,1]" by simp
  also have "... = itrev [] [3,2,1]" by simp
  also have "... = [3,2,1]" by simp
  finally show ?thesis .
\end{verbatim}
\texttt{qed}
Correctness of \texttt{itrev}

Let's try to prove that \texttt{itrev} reverses a list.

\texttt{lemma } "\texttt{itrev xs [] = reverse xs}"
\texttt{ oops}
lemma "itrev xs [] = reverse xs"
proof (induct xs)
  show "itrev [] [] = reverse []" by simp
next
  fix x xs assume IH: "itrev xs [] = reverse xs"
  have "itrev (x#xs) [] = itrev xs [x]" by simp
  oops

▶ The induction hypothesis does not apply to itrev xs [x].
▶ We need to generalize the lemma, make it stronger, to give ourselves more to assume in the induction hypothesis.
lemma "∀ ys. itrev xs ys = app (reverse xs) ys"
proof (induct xs)
  show "∀ ys. itrev [] ys = app (reverse []) ys" by simp
next
  fix x xs assume IH: "∀ ys. itrev xs ys = app (reverse xs) ys"
  show "∀ ys. itrev (x#xs) ys = app (reverse (x # xs)) ys"
  proof
    fix ys
    have "itrev (x#xs) ys = itrev xs (x#ys)" by simp
    also from IH have "... = app (reverse xs) (x#ys)" by simp
    also have "... = app (reverse xs) (app [x] ys)" by simp
    also have "... = app (app (reverse xs) [x]) ys"
      by (simp only: app_assoc)
    also have "... = app (reverse (x # xs)) ys" by simp
    finally show "itrev (x#xs) ys = app (reverse (x # xs)) ys".
  qed
qed