CIS 575. Introduction to Algorithm Analysis
Material for February 16, 2024

Induction for Correctness of Recursive Functions

1 Correctness Proofs for Recursive Algorithms

So far, we have given several examples of how to prove the correctness of *iterative* algorithms. We shall now address how to prove the correctness of algorithms that are built around **recursive** functions and thus typically of the form

```
f(x)
  if G
    ...
  else
    ...
    f(y_1)
    ...
    f(y_n)
```

which reflects that there must be some input on which no recursive call is made (when G is true); observe that while the body could contain several recursive calls, for a given input not all of them may be called (they could be in different sub-branches).

When we reason about the correctness of a recursive function, we certainly do not want to "unfold" the recursive calls as this could go on arbitrarily long; rather, we want to be able to **assume** that each recursive call meets its specification. For this assumption to be justified, we need each y_i to be in some sense "smaller" than x, as we can then use **mathematical induction**. We shall illustrate this technique through an example (and in the next note apply it to prove the correctness of recursive insertion sort).

2 Example Correctness Proof

Let us again consider the problem of finding the *last* occurrence in A of an element x that is known to exist in A, as specified by (with r the result)

Precondition $x \in A[1..n]$ (thus $n \geq 1$)

Postcondition $1 \leq r \leq n$, $A[r] = x$, $x \notin A[r+1..n]$

We have already proved the correctness of an iterative implementation; this time, let us look at a recursive implementation:

```
FindLast(A, n, x)
  if A[n] = x
    return n
  else
    return FindLast(A, n − 1, x)
```

We shall prove, by induction in n, that

$$\text{this implementation satisfies its specification for all } \mathbf{n \geq 1}. \tag{1}$$

We therefore assume the precondition, that $x \in A[1..n]$. To show the postcondition, we do a case analysis on the guard of the function body:

- if $A[n] = x$ we return n which will satisfy our postcondition since $1 \leq n \leq n$, $A[n] = x$, and $x \notin A[n+1 \ldots n]$ (as the latter segment is empty).
- if $A[n] \neq x$, a recursive call is made with $n - 1$. Observe that from $A[n] \neq x$ and $x \in A[1..n]$ we can infer

  $$x \in A[1..n-1]. \tag{2}$$

  We want to apply the induction hypothesis on that recursive call, and must therefore check:

  - that $n-1 < n$ (as the induction hypothesis can be applied only to smaller values) but that is obvious, and
  - that $n - 1 \geq 1$ (since (1) refers to values ≥ 1) which follows from (2).

  We can thus inductively assume that the recursive call satisfies its specification. Since (2) tells us that the precondition is true, we infer that the postcondition is true. That is, with r the value returned by the recursive call, we have

  $$1 \leq r \leq n-1,\ A[r] = x,\ x \notin A[r+1..n-1]$$

  which clearly (since $A[n] \neq x$) implies the desired postcondition

  $$1 \leq r \leq n,\ A[r] = x,\ x \notin A[r+1..n].$$

This completes the proof of (1). Observe that the "base case" (where no use is made of the induction hypothesis) is $A[n] = x$, which will hold when $n = 1$ (the "expected" base case) but may also hold for $n > 1$.