COMP30026 Models of Computation

Induction Principles

Lecture Week 5 Part 2 (Zo

Semester 2, 2021

This Lecture is Being Recorded

Mathematical Induction

"Mathematical" induction is always a proof about the natural numbers, $\mathbb{N}$.

We're us

We proce

1. In the basis step, we show $S(0)$.
2. In the inductive step, we take $S(n$ esis and use it to establish $S(n+1)$.

Proof by Induction

Theorem: For all $n \geq 0$,

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Proof: $n = 0$.

For the inductive step, assume the statement is tru fixed n, and we shall show that it also holds true with n.

So the statement to prove is

$$\sum_{i=1}^{n+1} i^2 = \frac{(n+1)(n+2)(2n+3)}{6}$$

Proof by Induction

But the claim

$$\sum_{i=1}^{n+1} i^2 = \frac{(n+1)(n+2)(2n+3)}{6}$$

is the same

$$\sum_{i=1} \quad \frac{}{6}$$

By the induction hypothesis, it suffices to show that

$$\frac{n(n+1)(2n+1)}{6} + (n+1)^2 = \frac{(n+1)(n+2)(2n+3)}{6}$$

This is done by simple polynomial algebra.

Sometimes more base cases may be needed.

Sometimes we need to use several statements $S(i), \ldots, S(n)$ to establish $S(n+1)$.

Theore

Proof: (10) are true.

For the inductive step, assume that n are true. Since $S(n-2)$ is true, also $n+1$ ca and 5s – just add 3 to the sum we had for $n-2$. Hence we have established $S(n+1)$.

We conclude that $S(n)$ holds for all $n \geq 8$.

Course-of-Values Induction

We can take the generality of "general induction" all the way:

To prove some claim $P(n)$, we are allowed to take the entire conjunction

as our indu

This variant is called course-of-values i

At first it looks like performing induction without a b

But the base case is implicitly included in the inductive step, because we have to prove $P(0)$ from nothing, that is, from *true*, the empty conjunction.

Recursive Structure and Induction

We often deal with recursively defined objects. Lists and trees are examples.

The set of w her example

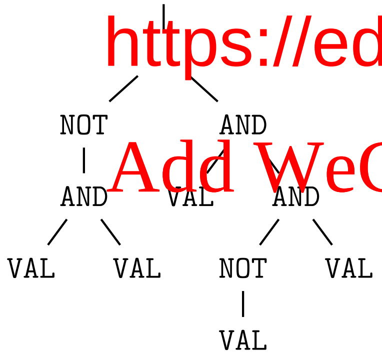
We will later meet context-free grammars; the language defined by such a grammar is a third example.

Induction is the natural way of proving assertions a jects.

In many cases we then rely on structural induction.

Consider the Haskell type Exp defined like so:


```
data Exp = AND Exp Exp | NOT Exp | VAL
```

- a b … odes,
- n b … odes,
- v be the number of VAL nodes.

I claim that $v = a + 1$, always.

The claim $v = a + 1$ applies to all trees that are inhabitants of `Exp`.

The definition of `Exp` told us that there are only three possible forms we need to deal with:

1. the t
2. a tre $\ldots_1 \ldots_2 \ldots_1 \ldots_2$
3. a tree of form (`NOT` t), where t is a tr

The first is a base case for induction.

It is straight-forward to prove $v = a + 1$ for the base case, since for `VAL`, a is 0 and v is 1.

Structural Induction: An Inductive Case

For the inductive case AND t_1 t_2, we proceed by assuming that the inductive hypothesis holds for t_1 and t_2.

That is, if the number of AND nodes in t_1 and t_2 is a_1 and a_2, respecti ... ectively, then we m

To get the number a of AND nodes in A t t number of AND nodes in t_1 and t_2, and the ... $_2 + 1$.

To get the number of VAL nodes, we just ha ... VAL nodes in t_1 and t_2: $v = v_1 + v_2$.

So $v = v_1 + v_2 = a_1 + 1 + a_2 + 1 = a + 1$. Just as we claimed!

Structural Induction: A Second Inductive Case

The case of $\texttt{NOT}\ t$ is even simpler.

Clearly th ber

in t, and si

So again w

Since we have established that $v = a +$

we conclude that it really is an invariant; it must hold fo sible

Exp trees.

Structural induction is a natural generalisation of course-of-values mathematical induction.

In Haskel tion:

```
da
```


Then structural induction over this type correspo course-of-values induction.

Conversely,if you prefer mathematical inducti hown $v = a + 1$ for the Exp trees, by doing induction on the height of the trees.

Next Week

We shall t large
dish of sets

This will b rse,
namely automata, formal language theory, and c