Claims + Evidence

This unit, we are interested in proving things about programming languages (PL).

Since we are doing proofs, of course we are going to be **logical**, but since we are working in PL, we are only interested in **computable/decidable** logic.

PL Proofs
- logical
- computable / decidable

=> Constructive (Intuitionistic logic)

This means we will be working with Constructive or Intuitionistic* logic.

In this setting, a **claim** is only true if we have **evidence** for it.

For example, I can claim that two is a natural number with the following **judgement**:

Judgement (claim):

$$2 \ \text{nat}$$

But that does not make this true. If I want this judgement to be true, if I want this to be an **evident judgement** (a judgement that has been proven true via evidence) I will need to provide evidence.

We will provide evidence using rules and proofs on them.

Rules (evidence):

Now I mentioned that we were going to be constructive, so to provide evidence that something is a natural number or prove things about natural numbers, I am going to explicitly state what I mean by natural numbers by providing rules that construct them.

That's a lot of words, so I will show you what I mean in Haskell first:

```
-- Rules for constructing natural numbers
> data Nat = Z
>          | S Nat

> two :: Nat  -- judgement
> two = S(S Z)  -- evidence
-- evident judgement
```

Q what would 3 be?

The rules for natural numbers actually look like this:

$$Z\ \frac{}{\text{zero nat}}$$

The zero rule says that zero is a natural number.

$$S\ \frac{n \ \text{nat}}{\text{succ } n \ \text{nat}}$$

The successor rule says that if n is a nat, then its successor (n+1) is also one.

These are the only way that we can construct natural numbers. Only zero and successors of zero are natural numbers.

We can use these rules to construct evidence for our judgements:

$$2 = \text{succ (succ zero)}$$

$$\cfrac{\cfrac{\cfrac{}{\text{zero nat}}\ Z}{\text{succ zero nat}}\ S}{\text{succ (succ zero) nat}}\ S$$

Q what would change for 3?

Q Can we prove -1 nat? Justify

Terminology:

Rules:

$$Z\ \frac{}{\text{zero nat}}$$

axiom

$$S\ \frac{n \ \text{nat}}{\text{succ } n \ \text{nat}}$$

n nat → premise(s)

succ n nat → conclusion(s)

Derivation

$$\cfrac{\cfrac{\cfrac{}{\text{zero nat}}\ Z}{\text{succ zero nat}}\ S}{\text{succ (succ zero) nat}}\ S$$

We can also define judgements simultaneously using mutually defined rules.

The Haskell analogue of this is this equivalent formulation of nats as odd or even:

```
> data Even = Zero
>           | Even Odd

> data Odd = Odd Even

> one :: Odd
> one = Odd Zero
```

Q What is two?

Lets now make the corresponding rules:

$$\frac{}{\text{zero even}}\ \text{EVENZ}$$

This axiom says that zero is even

$$\frac{n \ \text{odd}}{\text{succ } n \ \text{even}}\ \text{EVEN}$$

This rule says that if n is odd, then its successor is even.

Q Can you help me with odd?

$$\frac{n \ \text{even}}{\text{succ } n \ \text{odd}}\ \text{ODD}$$

Here we define oddness as any successor of an even number.

Q Can you help me prove that 2 is even?

$$\cfrac{\cfrac{\cfrac{}{\text{zero even}}\ \text{EVENZ}}{\text{succ zero odd}}\ \text{ODD}}{\text{succ (succ zero) even}}\ \text{EVEN}$$

---

*This is a subset of classical logic that captures decidable things. I do not expect you to know about it for this unit, but I wanted to provide the name in case you are interested! Another interesting link here is the Curry-Howard (-Lambek) correspondence

Derivable + Admissible Rules

It is often convenient to have more rules than absolutely necessary.

For example, in Haskell we may have smart constructors

```
>succSucc :: Nat -> Nat
>succSucc n = S(S n)
```

This smart constructor is equivalent to this superfluous rule:

$$\frac{n\ \mathbf{nat}}{\mathrm{succ}(\mathrm{succ}\ n)\ \mathbf{nat}}\,SS$$

This rule is equivalent to applying the S rule twice.

Any rule that is not necessary is called an **admissible** rule.

This example is a particular type of admissible rule called a **derivable** rule.

A rule is **derivable** if we can use a derivation of its premise as a building block in deriving its conclusion.

In this case, we can do that by applying the S rule twice:

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ n\ \mathbf{nat}\end{matrix}}{\mathrm{succ}\ n\ \mathbf{nat}}\,S}{\mathrm{succ}\ (\mathrm{succ}\ n)\ \mathbf{nat}}\,S$$

Not every admissible rule is derivable.
Admissibility has the following more general definition:

An admissible rule is a rule where if we have a derivation of the premise, we know that we can construct a derivation of the conclusion.

not necessarily by derivation

For example, the following rule is admissible but not derivable.

$$\frac{\mathrm{succ}\ n\ \mathbf{nat}}{n\ \mathbf{nat}}$$

The Haskell analogue of this is pattern matching: If we match on succ n :: Nat, we can extract n : Nat

```
> match :: Nat -> Nat
> --match Z can't happen by the premise
> match (S n) = n
```

Because, given a derivation for (succ n nat), it is easy to produce a derivation for (n nat):

eg. n = succ zero

$$\dfrac{\dfrac{\dfrac{}{\mathrm{zero}\ \mathbf{nat}}\,Z}{\mathrm{succ}\ \mathrm{zero}\ \mathbf{nat}}\,S}{\mathrm{succ}\ (\mathrm{succ}\ \mathrm{zero})\ \mathbf{nat}}\,S \qquad \dfrac{\dfrac{}{\mathrm{zero}\ \mathbf{nat}}\,Z}{\mathrm{succ}\ \mathrm{zero}\ \mathbf{nat}}\,S$$

We can just remove the bottom line from the derivation of the premise.

(It is the removal of the bottom line of the premise that makes this not derivable)

Induction

This is one of the key skills of this unit.

I think every sheet will contain a proof by induction. So please ensure you grasp this, and if you don't please chat to us ASAP.

Q Who has done proof by induction before?
At school? — should be all
At uni? (TiC / PLC) ↓ getting less
On rules?

Proof by induction is a huge benefit that we get from specifying what it means to be something via rules.

The idea is that to prove something for all things specified by the rules, you just need to prove that the rules preserve that property.

Consider our natural number rules:

$$\frac{}{\text{zero nat}}\ Z \qquad \frac{n\ \text{nat}}{\text{succ}\ n\ \text{nat}}\ S$$

They specify the only way to build nats, thus to prove something for all nats, we just need to show that it holds for Z, and is preserved by S.

Formally:

P is a property that holds for all nats if:

- It holds for zero (P(zero)), and
- whenever it holds for n (P(n)), it also holds for succ n (P(succ n)).

Just like induction you learned at school:
- prove for 0
- assume for n, and thus prove for n+1

We call this the principle of induction

Each set of defining rules has its own associated induction principle.

This includes simultaneously defined judgements.

For example, let's derive the induction principle for our odd/even numbers:

Recall the rules:

$$\frac{}{\text{zero even}}\ \text{EVENZ} \qquad \frac{n\ \text{odd}}{\text{succ}\ n\ \text{even}}\ \text{EVEN}$$

$$\frac{n\ \text{even}}{\text{succ}\ n\ \text{odd}}\ \text{ODD}$$

Induction principle:

Let P be a property of even numbers, Let Q be a property of odd numbers

P/Q will hold for all even/odd numbers if:

and now we have one point per rule.
Axioms are easy, we just assert that they hold:

- P(zero)

Other rules, will be if statements involving the premise

- whenever n even and P(n), we have Q(succ n).

Q Can you help with the final part?

- whenever n odd and Q(n), we have P(succ n).

We use these induction principles to perform proof by induction. I'll give you a little recipe and go through some examples.

Recipe for proof by induction:

1. Decide who to induct on
   - You must induct on a premise
   - Pick the "most interesting" premise
   - (Don't worry too much. If you get it wrong it will become apparent quickly and you can switch)
2. State that you are doing proof by induction and on which premise (readability)
3. Complete your proof by cases
   - One case per rule / point of induction principle
   - Remember to assume your premises
   - Write down your induction hypothesis so you remember to use it.

(see proofs note for more tips)

Example

Claim: If succ(n) nat then n nat

premise: succ(n) nat; conclusion: n nat

Proof:

Step One: we must pick who to induct on. Since this must be a premise, this is easy in our case, because we only have one premise: succ(n) nat

Step Two: we state that we are doing a proof by induction, and that it is on succ n nat.

Proof by induction on succ n nat.

We do this for readability. This is vital because the point of a proof is to convince the reader that the claim is true. The more accessible the proof the more likely it is to be convincing.

This statement aids readability because
- it primes the reader to expect a proof by induction
- it informs the reader which premise is being inducted on (helpful if there is more than one), and thus importantly which induction principle is being used.

Step Three: Okay this step was sort of just ~~"do the rest"~~, but now I will show you how.

We split into cases, one for each rule.

[Case: Zero]

Within each case, I always write what I want to prove specialised:

GOAL: If succ(zero) nat then zero nat.

Then I assume my premises, updating my goal

P = succ(zero) nat

GOAL' = zero nat

At this point, I hope that I have enough to prove my goal!

In this case, I do. I know how to prove that zero is a nat — using the Z rule!

(If I didn't see the solution, I would have brainstormed more things I know that are related, in the hope that were'd help. At the very least, it might get me close or some marks).

zero nat by Z:

$$\frac{}{\text{zero nat}}\ Z \qquad \square_{\text{zero}}$$

Of course, I could have noticed this soon as I wrote the specialised goal, with no need to involve the premise, but I just wanted to show you all the steps you can take.

On to the next case!

[Case: Succ]

Same as last time, I write my specialised goal and assume my premise

GOAL = if succ(succ n) nat
then succ n nat

P = succ(succ n) nat

GOAL' = succ n nat

Now at this point, because I am in an inductive case, there is also something else I have at my desposal: the induction hypothesis!

So I write that too so I have everything in front of me.

Note that the IH is exactly the statement you are trying to prove, but for the smaller expression specified in the induction principle. In our case for n:

IH = if succ n nat
then n nat

At this point, I look at all the things at my desposal (P, IH) and try and figure out how to prove the goal.

In this case, we don't need our IH, because the statement we are trying to prove is an admissible rule. Just the premise is enough.

By P we have succ(succ n) nat, which must have had the following derivation:

And this ⋯ is exactly what we need!

$$\frac{\dfrac{\vdots}{\text{succ } n \text{ nat}}}{\text{succ}(\text{succ } n)\text{ nat}}\ S$$

The derivation of P contains a derivation of our goal $\square_{\text{succ}}$

We have now covered all the cases, so we are done!

$\square$

That was a very simple example of a proof by induction, but have a go at using the recipe yourself in the first worksheet

Here's another example, but for simultaneous induction:

Example

Claim: If n even
then either
n = zero, or
n = succ x where x odd

Proof:

Proof by simultaneous induction on n even

Because we are inducting on even, which is mutually defined with odd, we must use simultaneous induction.

This also means that we need a complementary property for odd numbers to "complete" the chain. You can think of this as just passing on the buck:

Complementary claim for odd numbers:

If n odd then n = succ(x) where x even

We will call our original claim "CLAIM-E" and this one "CLAIM-O"

[Case: Zero]

Following the induction principle, here we just need to show that CLAIM-O holds for zero

GOAL = If zero even
then either
n = zero or
n = succ(x) where x odd

This is immediate as n = zero. $\square_{\text{zero}}$

[Case: Even]

In the even case, following the sim induction principle, we will assume CLAIM-O for n odd and try use that to prove CLAIM-E

GOAL = if succ n even
then either
succ n = zero, or — obviously not true
succ n = succ x where x odd — we must show this

P = succ n even

GOAL' = succ n = succ x where x odd

(IH = CLAIM-O for n
= if n odd
then n = succ x where x even)

The first part of our goal (the shape of succ n being succ x is immediate.

We just need to verify that x/n is indeed odd.

We can do this via the derivation of succ n:

$$\frac{\dfrac{\vdots}{n\ \text{odd}}}{\text{succ } n\ \text{even}}\ \text{EVEN}$$

$\square_{\text{even}}$

[CASE: ODD]

Finally, we must show CLAIM-O for succ n, assuming CLAIM-E

GOAL = CLAIM-O @ succ n
= If succ n odd
then succ n = succ x where x even

P = succ n odd

GOAL' = succ n = succ x where x even

As before, we immediately have the shape of succ x, and just need to show that x has the correct parity. This time we do this using the ODD rule:

$$\frac{n\ \text{even}}{\text{succ } n\ \text{odd}}\ \text{ODD}$$

$\square_{\text{odd}}$

Note how methodical doing proofs is. This is why theorem provers are so popular. The way I update my goal is actually reminicent of the way they work. For those interested, when I release the answers for sheet 1, I will also give you a formalised version in Lean 4 that a PhD student in our group wrote.

For more proof tips, see the proof note I will release with these "slides"