Induction

Induction

We've stated several 'theorems', but how do we know they are true?

Intuition is often wrong – we need *proof*.

Use proof process also for strengthening our intuition about subtle language features, and for debugging definitions – it helps you examine all the various cases.

Most of our definitions are inductive – so to prove things about them, we need the corresponding *induction principles*.

Three forms of induction

Prove facts about all natural numbers by *mathematical induction*.

Prove facts about all terms of a grammar (e.g. the L1 expressions) by *structural induction*.

Prove facts about all elements of a relation defined by rules (e.g. the L1 transition relation, or the L1 typing relation) by *rule induction*.

We shall see that all three boil down to induction over certain *trees*.

Principle of Mathematical Induction

For any property $\Phi(x)$ of natural numbers $x \in \mathbb{N} = \{0, 1, 2, ...\}$, to prove

$\forall\, x \in \mathbb{N}.\Phi(x)$

it's enough to prove

$\Phi(0)$ and $\forall\, x \in \mathbb{N}.\Phi(x) \Rightarrow \Phi(x+1)$.

i.e.

$$\boxed{\big(\Phi(0) \wedge (\forall\, x \in \mathbb{N}.\Phi(x) \Rightarrow \Phi(x+1))\big) \Rightarrow \forall\, x \in \mathbb{N}.\Phi(x)}$$

$$\boxed{\big(\Phi(0) \wedge (\forall\, x \in \mathbb{N}.\Phi(x) \Rightarrow \Phi(x+1))\big) \Rightarrow \forall\, x \in \mathbb{N}.\Phi(x)}$$

For example, to prove

Theorem 8 $1 + 2 + ... + x = 1/2 * x * (x + 1)$

use mathematical induction for

$\Phi(x) = (1 + 2 + ... + x = 1/2 * x * (x + 1))$

There's a model proof in the notes, *(annotated to say what's going on)*, as an example of good style. Writing a clear proof structure like this becomes essential when things get more complex – you have to *use* the formalism to help you get things right. Emulate it! *(but without the annotations!)*

(NB, the natural numbers include 0)

Theorem 8 $1 + 2 + ... + x = 1/2 * x * (x + 1)$.

Proof We prove $\forall\, x.\Phi(x)$, where *(state Φ explicitly)*

$$\Phi(x) \stackrel{\text{def}}{=} (1 + 2 + ... + x = 1/2 * x * (x + 1))$$

by mathematical induction. *(state the induction principle you're using)*

(Now show each conjunct of the premise of the induction principle)

Base case: *(conjunct $\Phi(0)$)*

$\Phi(0)$ is *(instantiate Φ)* $(1 + ... + 0 = 1/2 * 0 * (0 + 1))$, which holds as both sides are equal to 0.

Inductive step: *(conjunct $\forall\, x \in \mathbb{N}.\Phi(x) \Rightarrow \Phi(x + 1)$)*

Consider an arbitrary $k \in \mathbb{N}$ *(it's a universal ($\forall$), so consider an arbitrary one).*

Suppose $\Phi(k)$ *(to show the implication $\Phi(k) \Rightarrow \Phi(k + 1)$, assume the premise and try to show the conclusion).*

We have to show $\Phi(k + 1)$, i.e. *(state what we have to show explicitly)*

$$(1 + 2 + ... + (k + 1)) = 1/2 * (k + 1) * ((k + 1) + 1)$$

Now, the left hand side is

$$\begin{array}{lcll} (1 + 2 + ... + (k + 1)) & = & (1 + 2 + ... + k) + (k + 1) & \text{(rearranging)} \\ & = & (1/2 * k * (k + 1)) + (k + 1) & \text{(using } \Phi(k) \text{)} \end{array}$$

(say where you use the 'induction hypothesis' assumption $\Phi(k)$ made above)

and the right hand side is

$$\begin{array}{lcll} 1/2 * (k + 1) * ((k + 1) + 1) & = & 1/2 * (k * (k + 1) + (k + 1) * 1 + 1 * k + 1) & \text{(rearranging)} \\ & = & 1/2 * k * (k + 1) + 1/2 * ((k + 1) + k + 1) & \text{(rearranging)} \\ & = & 1/2 * k * (k + 1) + (k + 1) & \text{(rearranging)} \end{array}$$

which is equal to the LHS.

Complete Induction

For reference we recall here the principle of *complete induction*, which is equivalent to the principle of mathematical induction (anything you can prove with one, you could prove with the other) but is sometimes more convenient:

For any property $\Phi(k)$ of natural numbers $k \in \mathbb{N} = \{0, 1, 2, ...\}$, to prove

$\forall\, k \in \mathbb{N}.\Phi(k)$

it's enough to prove

$\forall\, k \in \mathbb{N}.(\forall\, y \in \mathbb{N}.y < k \Rightarrow \Phi(y)) \Rightarrow \Phi(k)$.

Abstract Syntax and Structural Induction

Abstract Syntax and Structural Induction

How to prove facts about all expressions, e.g. Determinacy for L1?

Theorem 1 (Determinacy) *If* $\langle e, s\rangle \longrightarrow \langle e_1, s_1\rangle$ *and* $\langle e, s\rangle \longrightarrow \langle e_2, s_2\rangle$ *then* $\langle e_1, s_1\rangle = \langle e_2, s_2\rangle$.

First, don't forget the elided universal quantifiers.

Theorem 1 (Determinacy) *For all* e, s, e_1, s_1, e_2, s_2*, if* $\langle e, s\rangle \longrightarrow \langle e_1, s_1\rangle$ *and* $\langle e, s\rangle \longrightarrow \langle e_2, s_2\rangle$ *then* $\langle e_1, s_1\rangle = \langle e_2, s_2\rangle$.

Abstract Syntax

Then, have to pay attention to what an expression *is*.

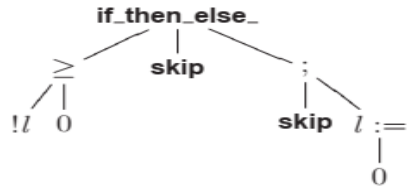
Recall we said:

$$\begin{aligned} e \;::=\; & n \mid b \mid e \;\; op \;\; e \mid \textbf{if}\;\; e \;\; \textbf{then} \;\; e \;\; \textbf{else} \;\; e \mid \\ & \ell := e \mid !\ell \mid \\ & \textbf{skip} \mid e; e \mid \\ & \textbf{while} \;\; e \;\; \textbf{do} \;\; e \end{aligned}$$

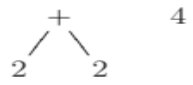
defining a set of expressions.

Q: Is an expression, e.g. **if** $!l \geq 0$ **then skip else** (**skip**; $l := 0$):

1. a list of characters `['i', 'f', '_', '!', 'l', ..]`;
2. a list of tokens `[ IF, DEREF, LOC "l", GTEQ, ..]`; or
3. an abstract syntax tree?

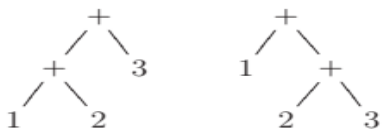


A: an abstract syntax tree. Hence: $2 + 2 \neq 4$



$1 + 2 + 3$ – ambiguous

$(1 + 2) + 3 \neq 1 + (2 + 3)$



Parentheses are only used for disambiguation – they are not part of the grammar. $1 + 2 = (1 + 2) = ((1 + 2)) = (((((1)))) + ((2)))$

All those are (sometimes) useful ways of looking at expressions (for lexing and parsing you start with (1) and (2)), but for semantics we don't want to be distracted by concrete syntax – it's easiest to work with abstract syntax trees, which for this grammar are finite trees, with ordered branches, labelled as follows:

- leaves (nullary nodes) labelled by $\mathbb{B} \cup \mathbb{Z} \cup (\{!\} * \mathbb{L}) \cup \{\textbf{skip}\} = \{\textbf{true}, \textbf{false}, \textbf{skip}\} \cup \{..., -1, 0, 1, ...\} \cup \{!l, !l_1, !l_2, ...\}$.
- unary nodes labelled by $\{l :=, l_1 :=, l_2 :=, ...\}$
- binary nodes labelled by $\{+, \geq, :=, ;, \textbf{while\_do\_}\}$
- ternary nodes labelled by $\{\textbf{if\_then\_else\_}\}$

Abstract grammar *suggests* a concrete syntax – we write expressions as strings just for convenience, using parentheses to disambiguate where required and infix/mixfix notation, but really mean trees. Arguments about exactly what concrete syntax a language should have – beloved amongst computer scientists everywhere – do not belong in a semantics course.

Just as for natural numbers to prove $\forall\ x\ \in\ \mathbb{N}.\Phi(x)$ it was enough to prove $\Phi(0)$ and all the implications $\Phi(x) \Rightarrow \Phi(x+1)$ (for arbitrary $x\ \in\ \mathbb{N}$), here to prove $\forall\ e\ \in\ L_1.\Phi(e)$ it is enough to prove $\Phi(c)$ for each nullary tree constructor c and all the implications $(\Phi(e_1) \wedge ... \wedge \Phi(e_k)) \Rightarrow \Phi(c(e_1, .., e_k))$ for each tree constructor of arity $k \geq 1$ (and for arbitrary $e_1\ \in L_1, .., e_k\ \in\ L_1$).

Principle of Structural Induction (for abstract syntax)

For any property $\Phi(e)$ of expressions e, to prove

$\forall\ e\ \in\ L_1.\Phi(e)$

it's enough to prove for each tree constructor c (taking $k \geq 0$ arguments) that if Φ holds for the subtrees $e_1, .., e_k$ then Φ holds for the tree $c(e_1, .., e_k)$. i.e.

$$\big(\forall\ c.\forall\ e_1, .., e_k.(\Phi(e_1) \wedge ... \wedge \Phi(e_k)) \Rightarrow \Phi(c(e_1, .., e_k))\big) \Rightarrow \forall\ e.\Phi(e)$$

where the tree constructors (or node labels) c are n, **true**, **false**, $!l$, **skip**, $l :=$, **while_do_**, **if_then_else_**, etc.

In particular, for L1: to show $\forall\ e\ \in\ L_1.\Phi(e)$ it's enough to show:

nullary: $\Phi(\textbf{skip})$

$\forall\ b\ \in \{\textbf{true}, \textbf{false}\}.\Phi(b)$

$\forall\ n\ \in\ \mathbb{Z}.\Phi(n)$

$\forall\ \ell\ \in\ \mathbb{L}.\Phi(!\ell)$

unary: $\forall\ \ell\ \in\ \mathbb{L}.\forall\ e.\Phi(e) \Rightarrow \Phi(\ell := e)$

binary: $\forall\ op\ .\forall\ e_1, e_2.(\Phi(e_1) \wedge \Phi(e_2)) \Rightarrow \Phi(e_1\ \ op\ \ e_2)$

$\forall\ e_1, e_2.(\Phi(e_1) \wedge \Phi(e_2)) \Rightarrow \Phi(e_1; e_2)$

$\forall\ e_1, e_2.(\Phi(e_1) \wedge \Phi(e_2)) \Rightarrow \Phi(\textbf{while}\ \ e_1\ \ \textbf{do}\ \ e_2)$

ternary: $\forall\ e_1, e_2, e_3.(\Phi(e_1) \wedge \Phi(e_2) \wedge \Phi(e_3)) \Rightarrow \Phi(\textbf{if}\ \ e_1\ \ \textbf{then}\ \ e_2\ \ \textbf{else}\ \ e_3)$

(See how this comes directly from the grammar)

If you think of the natural numbers as the abstract syntax trees of the grammar $n ::= \textbf{zero} \mid \textbf{succ}\ (n)$ then Structural Induction for that grammar is exactly the same as the Principal of Mathematical Induction.

Proving Determinacy (Outline)

Theorem 1 (Determinacy) *If* $\langle e, s\rangle \longrightarrow \langle e_1, s_1\rangle$ *and* $\langle e, s\rangle \longrightarrow \langle e_2, s_2\rangle$ *then* $\langle e_1, s_1\rangle = \langle e_2, s_2\rangle$.

Take

$$\Phi(e) \stackrel{\text{def}}{=} \forall\, s, e', s', e'', s''.\ (\langle e, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle e, s\rangle \longrightarrow \langle e'', s''\rangle) \Rightarrow \langle e', s'\rangle = \langle e'', s''\rangle$$

and show $\forall\ e\ \in\ L_1.\Phi(e)$ by structural induction.

To do that we need to verify all the premises of the principle of structural induction – the formulae in the second box below – for this Φ.

$$\Phi(e) \stackrel{\text{def}}{=} \forall\, s, e', s', e'', s''.\ (\langle e, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle e, s\rangle \longrightarrow \langle e'', s''\rangle) \Rightarrow \langle e', s'\rangle = \langle e'', s''\rangle$$

nullary:	$\Phi(\textbf{skip})$
	$\forall\ b\ \in \{\textbf{true}, \textbf{false}\}.\Phi(b)$
	$\forall\ n\ \in\ \mathbb{Z}.\Phi(n)$
	$\forall\ \ell\ \in\ \mathbb{L}.\Phi(!\ell)$
unary:	$\forall\ \ell\ \in\ \mathbb{L}.\forall\ e.\Phi(e) \Rightarrow \Phi(\ell := e)$
binary:	$\forall\ \ op\ .\forall\ e_1, e_2.(\Phi(e_1) \wedge \Phi(e_2)) \Rightarrow \Phi(e_1\ \ op\ \ e_2)$
	$\forall\ e_1, e_2.(\Phi(e_1) \wedge \Phi(e_2)) \Rightarrow \Phi(e_1; e_2)$
	$\forall\ e_1, e_2.(\Phi(e_1) \wedge \Phi(e_2)) \Rightarrow \Phi(\textbf{while}\ \ e_1\ \ \textbf{do}\ \ e_2)$
ternary:	$\forall\ e_1, e_2, e_3.(\Phi(e_1) \wedge \Phi(e_2) \wedge \Phi(e_3)) \Rightarrow \Phi(\textbf{if}\ \ e_1\ \ \textbf{then}\ \ e_2\ \ \textbf{else}\ \ e_3)$

We will come back later to look at some of these details.

Inductive Definitions and Rule Induction

Inductive Definitions and Rule Induction

How to prove facts about all elements of the L1 typing relation or the L1 reduction relation, e.g. Progress or Type Preservation?

Theorem 2 (Progress) *If* $\Gamma \vdash e\colon T$ *and* $dom(\Gamma) \subseteq dom(s)$ *then either* e *is a value or there exist* e', s' *such that* $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$.

Theorem 3 (Type Preservation) *If* $\Gamma \vdash e\colon T$ *and* $dom(\Gamma) \subseteq dom(s)$ *and* $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ *then* $\Gamma \vdash e'\colon T$ *and* $dom(\Gamma) \subseteq dom(s')$.

Have to pay attention to what the elements of these relations really are...

Inductive Definitions

We defined the transition relation $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ and the typing relation $\Gamma \vdash e{:}T$ by giving some rules, eg

$$\text{(op +)}\quad \langle n_1 + n_2, s\rangle \longrightarrow \langle n, s\rangle \quad \text{if } n = n_1 + n_2$$

$$\text{(op1)}\quad \frac{\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle}{\langle e_1 \;\; op \;\; e_2, s\rangle \longrightarrow \langle e_1' \;\; op \;\; e_2, s'\rangle}$$

$$\text{(op +)}\quad \frac{\Gamma \vdash e_1{:}\mathsf{int} \quad \Gamma \vdash e_2{:}\mathsf{int}}{\Gamma \vdash e_1 + e_2{:}\mathsf{int}}$$

What did we actually mean?

These relations are just normal set-theoretic relations, written in infix or mixfix notation.

For the transition relation:

- Start with $A = L_1 * \text{store} * L_1 * \text{store}$.
- Write $\longrightarrow \subseteq A$ infix, e.g. $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ instead of $(e, s, e', s') \in \longrightarrow$.

For the typing relation:

- Start with $A = \text{TypeEnv} * L_1 * \text{types}$.
- Write $\vdash \subseteq A$ mixfix, e.g. $\Gamma \vdash e{:}T$ instead of $(\Gamma, e, T) \in \vdash$.

For each rule we can construct the set of all concrete *rule instances*, taking all values of the metavariables that satisfy the side condition. For example, for (op +) and (op1) we take all values of n_1, n_2, s, n (satisfying $n = n_1 + n_2$) and of e_1, e_2, s, e_1', s'.

$$\text{(op+)}\ \frac{}{\langle 2+2, \{\}\rangle \longrightarrow \langle 4, \{\}\rangle}\ , \quad \text{(op +)}\ \frac{}{\langle 2+3, \{\}\rangle \longrightarrow \langle 5, \{\}\rangle}\ , \ldots$$

$$\text{(op1)}\ \frac{\langle 2+2, \{\}\rangle \longrightarrow \langle 4, \{\}\rangle}{\langle (2+2)+3, \{\}\rangle \longrightarrow \langle 4+3, \{\}\rangle}\ , \quad \text{(op1)}\ \frac{\langle 2+2, \{\}\rangle \longrightarrow \langle \mathsf{false}, \{\}\rangle}{\langle (2+2)+3, \{\}\rangle \longrightarrow \langle \mathsf{false}+3, \{\}\rangle}$$

Note the last has a premise that is not itself derivable, but nonetheless this is a legitimate instance of (op1).

Now a *derivation* of a transition $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ or typing judgment $\Gamma \vdash e{:}T$ is a finite tree such that each step *is* a concrete rule instance.

$$\cfrac{\cfrac{\cfrac{}{\langle 2+2, \{\}\rangle \longrightarrow \langle 4, \{\}\rangle}\,\text{(op+)}}{\langle (2+2)+3, \{\}\rangle \longrightarrow \langle 4+3, \{\}\rangle}\,\text{(op1)}}{\langle (2+2)+3 \geq 5, \{\}\rangle \longrightarrow \langle 4+3 \geq 5, \{\}\rangle}\,\text{(op1)}$$

$$\cfrac{\cfrac{\cfrac{}{\Gamma \vdash !l{:}\mathsf{int}}\,\text{(deref)} \quad \cfrac{}{\Gamma \vdash 2{:}\mathsf{int}}\,\text{(int)}}{\Gamma \vdash (!l + 2){:}\mathsf{int}}\,\text{(op +)} \quad \cfrac{}{\Gamma \vdash 3{:}\mathsf{int}}\,\text{(int)}}{\Gamma \vdash (!l + 2) + 3{:}\mathsf{int}}\,\text{(op +)}$$

and $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ is an element of the reduction relation (resp. $\Gamma \vdash e{:}T$ is an element of the transition relation) iff there is a derivation with that as the root node.

Now, to prove something about an inductively-defined set...

Principle of Rule Induction

For any property $\Phi(a)$ of elements a of A, and any set of rules which define a subset S_R of A, to prove

$$\forall\ a\ \in\ S_R.\Phi(a)$$

it's enough to prove that $\{a \mid \Phi(a)\}$ is closed under the rules, ie for each concrete rule instance

$$\frac{h_1 \quad .. \quad h_k}{c}$$

if $\Phi(h_1) \wedge ... \wedge \Phi(h_k)$ then $\Phi(c)$.

For some proofs a slightly different principle is useful – this variant allows you to assume each of the h_i are themselves members of S_R.

Principle of rule induction (a slight variant)

For any property $\Phi(a)$ of elements a of A, and any set of rules which inductively define the set S_R, to prove

$$\forall\ a\ \in\ S_R.\Phi(a)$$

it's enough to prove that

for each concrete rule instance

$$\frac{h_1 \quad .. \quad h_k}{c}$$

if $\Phi(h_1) \wedge ... \wedge \Phi(h_k) \wedge h_1\ \in\ S_R \wedge .. \wedge h_k\ \in\ S_R$ then $\Phi(c)$.

Proving Progress (Outline)

Theorem 2 (Progress) *If* $\Gamma \vdash e{:}T$ *and* $dom(\Gamma) \subseteq dom(s)$ *then either* e *is a value or there exist* e', s' *such that* $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$.

Proof Take

$$\Phi(\Gamma, e, T) \stackrel{\text{def}}{=} \forall\ s.\ \text{dom}(\Gamma) \subseteq \text{dom}(s) \Rightarrow \text{value}(e) \vee (\exists\ e', s'.\langle e, s\rangle \longrightarrow \langle e', s'\rangle)$$

We show that for all Γ, e, T, if $\Gamma \vdash e{:}T$ then $\Phi(\Gamma, e, T)$, by rule induction on the definition of $\vdash$.

Principle of Rule Induction (variant form): to prove $\Phi(a)$ for all a in the set S_R, it's enough to prove that for each concrete rule instance

$$\frac{h_1 \quad .. \quad h_k}{c}$$

if $\Phi(h_1) \wedge ... \wedge \Phi(h_k) \wedge h_1 \in S_R \wedge .. \wedge h_k \in S_R$ then $\Phi(c)$.

Instantiating to the L1 typing rules, have to show:

(int) $\forall \Gamma, n.\Phi(\Gamma, n, \mathsf{int})$

(deref) $\forall \Gamma, \ell.\Gamma(\ell) = \mathsf{intref} \Rightarrow \Phi(\Gamma, !\ell, \mathsf{int})$

(op +) $\forall \Gamma, e_1, e_2.(\Phi(\Gamma, e_1, \mathsf{int}) \wedge \Phi(\Gamma, e_2, \mathsf{int}) \wedge \Gamma \vdash e_1{:}\mathsf{int} \wedge \Gamma \vdash e_2{:}\mathsf{int})$
$\Rightarrow \Phi(\Gamma, e_1 + e_2, \mathsf{int})$

(seq) $\forall \Gamma, e_1, e_2, T.(\Phi(\Gamma, e_1, \mathsf{unit}) \wedge \Phi(\Gamma, e_2, T) \wedge \Gamma \vdash e_1{:}\mathsf{unit} \wedge \Gamma \vdash e_2{:}T)$
$\Rightarrow \Phi(\Gamma, e_1; e_2, T)$

etc.

Having proved those 10 things, consider an example $\Gamma \vdash (!l + 2) + 3{:}\mathsf{int}$. To see why $\Phi(\Gamma,\ (!l + 2) + 3,\ \mathsf{int})$ holds:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash !l{:}\mathsf{int}}\ (\text{deref}) \quad \dfrac{}{\Gamma \vdash 2{:}\mathsf{int}}\ (\text{int})}{\Gamma \vdash (!l + 2){:}\mathsf{int}}\ (\text{op} +) \quad \dfrac{}{\Gamma \vdash 3{:}\mathsf{int}}\ (\text{int})}{\Gamma \vdash (!l + 2) + 3{:}\mathsf{int}}\ (\text{op} +)$$

Which Induction Principle to Use?

Which of these induction principles to use is a matter of convenience – you want to use an induction principle that matches the definitions you're working with.

For completeness, observe the following:

Mathematical induction over $\mathbb{N}$ is equivalent to complete induction over $\mathbb{N}$.

Mathematical induction over $\mathbb{N}$ is essentially the same as structural induction over $n ::= \mathbf{zero} \mid \mathbf{succ}\ (n)$.

Instead of using structural induction (for an arbitrary grammar), you could use complete induction on the *size* of terms.

Instead of using structural induction, you could use rule induction: supposing some fixed set of tree node labels (e.g. all the character strings), take A to be the set of all trees with those labels, and consider each clause of your grammar (e.g. $e ::= ... \mid e + e$) to be a rule

$$\frac{e \quad e}{e + e}$$

Example Proofs

Example Proofs

In the notes there are detailed example proofs for Determinacy (structural induction), Progress (rule induction on type derivations), and Type Preservation (rule induction on reduction derivations).

You should read them off-line, and do the exercises.

When is a proof a proof?

What's a proof?

Formal: a derivation in formal logic (e.g. a big natural deduction proof tree). Often far too verbose to deal with by hand (but can *machine-check* such things).

Informal but rigorous: an argument to persuade the reader that, if pushed, you could write a fully formal proof (the usual mathematical notion, e.g. those we just did). Have to learn by practice to see when they are rigorous.

Bogus: neither of the above.

Remember – the point is to use the mathematics to *help you think* about things that are too complex to keep in your head all at once: to keep track of all the cases etc. To do that, and to communicate with other people, it's important to *write down* the reasoning and proof structure as clearly as possible. After you've done a proof you should give it to someone (your supervision partner first, perhaps) to see if they (a) can understand what you've said, and (b) if they believe it.

Sometimes it seems hard or pointless to prove things because they seem 'too obvious'....

1. proof lets you see (and explain) ***Why*** they are obvious
2. sometimes the obvious facts are false...
3. sometimes the obvious facts are not obvious at all
4. sometimes a proof contains or suggests an algorithm that you need – eg, proofs that type inference is decidable (for fancier type systems)

Theorem 1 (Determinacy) *If* $\langle e, s\rangle \longrightarrow \langle e_1, s_1\rangle$ *and* $\langle e, s\rangle \longrightarrow \langle e_2, s_2\rangle$ *then* $\langle e_1, s_1\rangle = \langle e_2, s_2\rangle$.

Proof Take

$$\Phi(e) \stackrel{\text{def}}{=} \forall\, s, e', s', e'', s''.(\langle e, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle e, s\rangle \longrightarrow \langle e'', s''\rangle) \Rightarrow \langle e', s'\rangle = \langle e'', s''\rangle$$

We show $\forall\ e\ \in\ L_1.\Phi(e)$ by structural induction.

Cases skip, b, n**.** For e of these forms there are no rules with a conclusion of the form $\langle e, ...\rangle \longrightarrow \langle .., ..\rangle$ so the left hand side of the implication cannot hold, so the implication is true.

Case $!\ell$**.** Take arbitrary s, e', s', e'', s'' such that $\langle !\ell, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle !\ell, s\rangle \longrightarrow \langle e'', s''\rangle$.

The only rule which could be applicable is (deref), in which case, for those transitions to be instances of the rule we must have

$$\begin{array}{ll} \ell \in \mathrm{dom}(s) & \ell \in \mathrm{dom}(s) \\ e' = s(\ell) & e'' = s(\ell) \\ s' = s & s'' = s \end{array}$$

so $e' = e''$ and $s' = s''$.

Case $\ell := e$**.** Suppose $\Phi(e)$ (then we have to show $\Phi(\ell := e)$).

Take arbitrary s, e', s', e'', s'' such that $\langle \ell := e, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle \ell := e, s\rangle \longrightarrow \langle e'', s''\rangle$.

It's handy to have this lemma:

Lemma 1 *For all* $e \in L_1$*, if* e *is a value then* $\forall\ s.\neg\ \exists e', s'.\langle e, s\rangle \longrightarrow \langle e', s'\rangle$.

Proof By defn e is a value if it is of one of the forms $n, b,$ **skip**. By examination of the rules on slides ..., there is no rule with conclusion of the form $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ for e one of $n, b,$ **skip**.

The only rules which could be applicable, for each of the two transitions, are (assign1) and (assign2).

case $\langle \ell := e, s\rangle \longrightarrow \langle e', s'\rangle$ is an instance of (assign1). Then for some n we have $e = n$ and $\ell \in \mathrm{dom}(s)$ and $e' =$ **skip** and $s' = s + \{\ell \mapsto n\}$.

case $\langle \ell := n, s\rangle \longrightarrow \langle e'', s''\rangle$ is an instance of (assign1) (note we are using the fact that $e = n$ here). Then $e'' =$ **skip** and $s'' = s + \{\ell \mapsto n\}$ so $\langle e', s'\rangle = \langle e'', s''\rangle$ as required.

case $\langle \ell := e, s\rangle \longrightarrow \langle e'', s''\rangle$ is an instance of (assign2). Then $\langle n, s\rangle \longrightarrow \langle e'', s''\rangle$, which contradicts the lemma, so this case cannot arise.

case $\langle \ell := e, s\rangle \longrightarrow \langle e', s'\rangle$ is an instance of (assign2). Then for some e_1' we have $\langle e, s\rangle \longrightarrow \langle e_1', s'\rangle$ (*) and $e' = (\ell := e_1')$.

case $\langle \ell := e, s\rangle \longrightarrow \langle e'', s''\rangle$ is an instance of (assign1). Then for some n we have $e = n$, which contradicts the lemma, so this case cannot arise.

case $\langle \ell := e, s\rangle \longrightarrow \langle e'', s''\rangle$ is an instance of (assign2). Then for some e_1'' we have $\langle e, s\rangle \longrightarrow \langle e_1'', s''\rangle$(**) and $e'' = (\ell := e_1'')$. Now, by the induction hypothesis $\Phi(e)$, (*) and (**) we have $\langle e_1', s'\rangle = \langle e_1'', s''\rangle$, so $\langle e', s'\rangle = \langle \ell := e_1', s'\rangle = \langle \ell := e_1'', s''\rangle = \langle e'', s''\rangle$ as required.

Case $e_1\ \ op\ \ e_2$. Suppose $\Phi(e_1)$ and $\Phi(e_2)$.

Take arbitrary s, e', s', e'', s'' such that $\langle e_1\ \ op\ \ e_2, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle e_1\ \ op\ \ e_2, s\rangle \longrightarrow \langle e'', s''\rangle$.

By examining the expressions in the left-hand-sides of the conclusions of the rules, and using the lemma above, the only possibilities are those below (you should check why this is so for yourself).

case $op = +$ and $\langle e_1 + e_2, s\rangle \longrightarrow \langle e', s'\rangle$ is an instance of (op+) and $\langle e_1 + e_2, s\rangle \longrightarrow \langle e'', s''\rangle$ is an instance of (op+).

Then for some n_1, n_2 we have $e_1 = n_1$, $e_2 = n_2$, $e' = n_3 = e''$ for $n_3 = n_1 + n_2$, and $s' = s = s''$.

case $op = \geq$ and $\langle e_1 \geq e_2, s\rangle \longrightarrow \langle e', s'\rangle$ is an instance of (op$\geq$) and $\langle e_1 \geq e_2, s\rangle \longrightarrow \langle e'', s''\rangle$ is an instance of (op$\geq$).

Then for some n_1, n_2 we have $e_1 = n_1$, $e_2 = n_2$, $e' = b = e''$ for $b = (n_1 \geq n_2)$, and $s' = s = s''$.

case $\langle e_1 \ op \ e_2, s\rangle \longrightarrow \langle e', s'\rangle$ is an instance of (op1) and $\langle e_1 \ op \ e_2, s\rangle \longrightarrow \langle e'', s''\rangle$ is an instance of (op1).

Then for some e_1' and e_1'' we have $\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle$ (*), $\langle e_1, s\rangle \longrightarrow \langle e_1'', s''\rangle$ (**), $e' = e_1' \ op \ e_2$, and $e'' = e_1'' \ op \ e_2$. Now, by the induction hypothesis $\Phi(e_1)$, (*) and (**) we have $\langle e_1', s'\rangle = \langle e_1'', s''\rangle$, so $\langle e', s'\rangle = \langle e_1' \ op \ e_2, s'\rangle = \langle e_1'' \ op \ e_2, s''\rangle = \langle e'', s''\rangle$ as required.

case $\langle e_1 \ op \ e_2, s\rangle \longrightarrow \langle e', s'\rangle$ is an instance of (op2) and $\langle e_1 \ op \ e_2, s\rangle \longrightarrow \langle e'', s''\rangle$ is an instance of (op2).

Similar, save that we use the induction hypothesis $\Phi(e_2)$.

Case $e_1; e_2$**.** Suppose $\Phi(e_1)$ and $\Phi(e_2)$.

Take arbitrary s, e', s', e'', s'' such that $\langle e_1; e_2, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle e_1; e_2, s\rangle \longrightarrow \langle e'', s''\rangle$.

By examining the expressions in the left-hand-sides of the conclusions of the rules, and using the lemma above, the only possibilities are those below.

case $e_1 =$ **skip** and both transitions are instances of (seq1).

Then $\langle e', s'\rangle = \langle e_2, s\rangle = \langle e'', s''\rangle$.

case e_1 is not a value and both transitions are instances of (seq2). Then for some e_1' and e_1'' we have $\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle$ (*), $\langle e_1, s\rangle \longrightarrow \langle e_1'', s''\rangle$ (**), $e' = e_1'; e_2$, and $e'' = e_1''; e_2$

Then by the induction hypothesis $\Phi(e_1)$ we have $\langle e_1', s'\rangle = \langle e_1'', s''\rangle$, so $\langle e', s'\rangle = \langle e_1'; e_2, s'\rangle = \langle e_1''; e_2, s''\rangle = \langle e'', s''\rangle$ as required.

Case while e_1 **do** e_2**.** Suppose $\Phi(e_1)$ and $\Phi(e_2)$.

Take arbitrary s, e', s', e'', s'' such that $\langle$**while** e_1 **do** $e_2, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle$**while** e_1 **do** $e_2, s\rangle \longrightarrow \langle e'', s''\rangle$.

By examining the expressions in the left-hand-sides of the conclusions of the rules both must be instances of (while), so $\langle e', s'\rangle = \langle$**if** e_1 **then** $(e_2;$ **while** e_1 **do** $e_2)$ **else skip**$, s\rangle = \langle e'', s''\rangle$.

Case if e_1 **then** e_2 **else** e_3**.** Suppose $\Phi(e_1)$, $\Phi(e_2)$ and $\Phi(e_3)$.

Take arbitrary s, e', s', e'', s'' such that $\langle$**if** e_1 **then** e_2 **else** $e_3, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle$**if** e_1 **then** e_2 **else** $e_3, s\rangle \longrightarrow \langle e'', s''\rangle$.

By examining the expressions in the left-hand-sides of the conclusions of the rules, and using the lemma above, the only possibilities are those below.

case $e_1 =$ **true** and both transitions are instances of (if1).

case $e_1 =$ **false** and both transitions are instances of (if2).

case e_1 is not a value and both transitions are instances of (if3).

The first two cases are immediate; the last uses $\Phi(e_1)$.

(check we've done all the cases!)

(note that the level of written detail can vary, as here – if you and the reader agree – but you must do all the steps in your head. If in any doubt, write it down, as an aid to thought...!)

Theorem 2 (Progress) *If* $\Gamma \vdash e{:}T$ *and* $dom(\Gamma) \subseteq dom(s)$ *then either* e *is a value or there exist* e', s' *such that* $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$.

Proof Take

$$\Phi(\Gamma, e, T) \stackrel{\text{def}}{=} \forall\, s.\mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(s) \Rightarrow \mathrm{value}(e) \vee (\exists\, e', s'.\langle e, s\rangle \longrightarrow \langle e', s'\rangle)$$

We show that for all Γ, e, T, if $\Gamma \vdash e{:}T$ then $\Phi(\Gamma, e, T)$, by rule induction on the definition of $\vdash$.

Case (int). Recall the rule scheme

$$\text{(int)} \quad \Gamma \vdash n{:}\mathsf{int} \quad \text{for } n \in \mathbb{Z}$$

It has no premises, so we have to show that for all instances Γ, e, T of the conclusion we have $\Phi(\Gamma, e, T)$.

For any such instance, there must be an $n \in \mathbb{Z}$ for which $e = n$.

Now Φ is of the form $\forall\, s.\mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(s) \Rightarrow ...$, so consider an arbitrary s and assume $\mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(s)$.

We have to show $\mathrm{value}(e) \vee (\exists\, e', s'.\langle e, s\rangle \longrightarrow \langle e', s'\rangle)$. But the first disjunct is true as integers are values (according to the definition).

Case (bool) similar.

Case (op+). Recall the rule

$$\text{(op +)} \quad \frac{\Gamma \vdash e_1{:}\mathsf{int} \quad \Gamma \vdash e_2{:}\mathsf{int}}{\Gamma \vdash e_1 + e_2{:}\mathsf{int}}$$

We have to show that for all Γ, e_1, e_2, if $\Phi(\Gamma, e_1, \mathsf{int})$ and $\Phi(\Gamma, e_2, \mathsf{int})$ then $\Phi(\Gamma, e_1 + e_2, \mathsf{int})$.

Suppose $\Phi(\Gamma, e_1, \mathsf{int})$ (*), $\Phi(\Gamma, e_2, \mathsf{int})$ (**), $\Gamma \vdash e_1{:}\mathsf{int}$ (***), and $\Gamma \vdash e_2{:}\mathsf{int}$ (****) (note that we're using the variant form of rule induction here).

Consider an arbitrary s. Assume $\mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(s)$.

We have to show $\mathrm{value}(e_1 + e_2) \vee (\exists\, e', s'.\langle e_1 + e_2, s\rangle \longrightarrow \langle e', s'\rangle)$.

Now the first disjunct is false ($e_1 + e_2$ is not a value), so we have to show the second, i.e.$\exists \langle e', s'\rangle.\langle e_1 + e_2, s\rangle \longrightarrow \langle e', s'\rangle$.

By (*) one of the following holds.

case $\exists\, e_1', s'.\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle$.

Then by (op1) we have $\langle e_1 + e_2, s\rangle \longrightarrow \langle e_1' + e_2, s'\rangle$, so we are done.

case e_1 is a value. By (**) one of the following holds.

case $\exists\, e_2', s'.\langle e_2, s\rangle \longrightarrow \langle e_2', s'\rangle$.

Then by (op2) $\langle e_1 + e_2, s\rangle \longrightarrow \langle e_1 + e_2', s'\rangle$, so we are done.

case e_2 is a value.

(Now want to use (op+), but need to know that e_1 and e_2 are really integers.)

Lemma 2 *for all* Γ, e, T, *if* $\Gamma \vdash e{:}T$, e *is a value and* $T = \mathsf{int}$ *then for some* $n \in \mathbb{Z}$ *we have* $e = n$.

Proof By rule induction. Take $\Phi'(\Gamma, e, T) = ((\mathrm{value}(e) \wedge T = \mathsf{int}) \Rightarrow \exists\, n \in \mathbb{Z}.e = n)$.

Case (int). ok

Case (bool),(skip). In instances of these rules the conclusion is a value but the type is not int, so ok.

Case otherwise. In instances of all other rules the conclusion is not a value, so ok.

(a rather trivial use of rule induction – we never needed to use the induction hypothesis, just to do case analysis of the last rule that might have been used in a derivation of $\Gamma \vdash e{:}T$).

Using the Lemma, (***) and (****) there exist $n_1 \in \mathbb{Z}$ and $n_2 \in \mathbb{Z}$ such that $e_1 = n_1$ and $e_2 = n_2$. Then by (op+) $\langle e_1 + e_2, s\rangle \longrightarrow \langle n, s\rangle$ where $n = n_1 + n_2$, so we are done.

Case (op $\geq$). Similar to (op +).

Case (if). Recall the rule

$$(\mathrm{if})\quad \frac{\begin{array}{c}\Gamma \vdash e_1{:}\mathsf{bool}\\ \Gamma \vdash e_2{:}T\\ \Gamma \vdash e_3{:}T\end{array}}{\Gamma \vdash \mathbf{if}\ \ e_1\ \ \mathbf{then}\ \ e_2\ \ \mathbf{else}\ \ e_3{:}T}$$

Suppose $\Phi(\Gamma, e_1, \mathsf{bool})$ (*1), $\Phi(\Gamma, e_2, T)$ (*2), $\Phi(\Gamma, e_3, T)$ (*3), $\Gamma \vdash e_1{:}\mathsf{bool}$ (*4), $\Gamma \vdash e_2{:}T$ (*5) and $\Gamma \vdash e_3{:}T$ (*6).

Consider an arbitrary s. Assume $\mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(s)$. Write e for **if** e_1 **then** e_2 **else** e_3.

This e is not a value, so we have to show $\langle e, s\rangle$ has a transition.

case $\exists\, e_1', s'.\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle$.

Then by (if3) $\langle e, s\rangle \longrightarrow \langle \mathbf{if}\ \ e_1'\ \ \mathbf{then}\ \ e_2\ \ \mathbf{else}\ \ e_3, s\rangle$, so we are done.

case e_1 is a value.

(Now want to use (if1) or (if2), but need to know that $e_1 \in \{\mathbf{true}, \mathbf{false}\}$. Realise should have proved a stronger Lemma above).

Lemma 3 *For all* Γ, e, T. *if* $\Gamma \vdash e{:}T$ *and* e *is a value, then* $T = \mathsf{int} \Rightarrow \exists\, n \in \mathbb{Z}.e = n$, $T = \mathsf{bool} \Rightarrow \exists\, b \in \{\mathbf{true}, \mathbf{false}\}.e = b$, *and* $T = \mathsf{unit} \Rightarrow e = \mathbf{skip}$.

Proof By rule induction – details omitted.

Using the Lemma and (*4) we have $\exists\, b \in \{\mathbf{true}, \mathbf{false}\}.e_1 = b$.

case $b = \mathbf{true}$. Use (if1).

case $b = \mathbf{false}$. Use (if2).

Case (deref). Recall the rule

$$(\mathrm{deref})\quad \frac{\Gamma(\ell) = \mathsf{intref}}{\Gamma \vdash !\ell{:}\mathsf{int}}$$

(This is a leaf – it has no $\Gamma \vdash e{:}T$ premises - so no Φs to assume).

Consider an arbitrary s with $\mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(s)$.

By the condition $\Gamma(\ell) = \mathsf{intref}$ we have $\ell \in \mathrm{dom}(\Gamma)$, so $\ell \in \mathrm{dom}(s)$, so there is some n with $s(\ell) = n$, so there is an instance of (deref) $\langle !\ell, s\rangle \longrightarrow \langle n, s\rangle$.

Cases (assign), (skip), (seq), (while). Left as an exercise.

Theorem 3 (Type Preservation) *If* $\Gamma \vdash e{:}T$ *and* $dom(\Gamma) \subseteq dom(s)$ *and* $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ *then* $\Gamma \vdash e'{:}T$ *and* $dom(\Gamma) \subseteq dom(s')$.

Proof First show the second part, using the following lemma.

Lemma 4 *If* $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ *then* $dom(s') = dom(s)$.

Proof Rule induction on derivations of $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$. Take $\Phi(e, s, e', s') = (\mathrm{dom}(s) = \mathrm{dom}(s'))$.

All rules are immediate uses of the induction hypothesis except (assign1), for which we note that if $\ell \in \mathrm{dom}(s)$ then $\mathrm{dom}(s + (\ell \mapsto n)) = \mathrm{dom}(s)$.

Now prove the first part, ie If $\Gamma \vdash e{:}T$ and $\mathrm{dom}(\Gamma) \supseteq \mathrm{dom}(s)$ and $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ then $\Gamma \vdash e'{:}T$.

Prove by rule induction on derivations of $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$.

Take $\Phi(e, s, e', s') = \forall\, \Gamma, T.(\Gamma \vdash e{:}T \wedge \mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(s)) \Rightarrow \Gamma \vdash e'{:}T$.

Case (op+). Recall

$$(\mathrm{op}+) \quad \langle n_1 + n_2, s\rangle \longrightarrow \langle n, s\rangle \quad \text{if } n = n_1 + n_2$$

Take arbitrary Γ, T. Suppose $\Gamma \vdash n_1 + n_2{:}T$ (*) and $\mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(s)$. The last rule in the derivation of (*) must have been (op+), so must have $T = \mathsf{int}$. Then can use (int) to derive $\Gamma \vdash n{:}T$.

Case (op $\geq$). Similar.

Case (op1). Recall

$$(\mathrm{op1}) \quad \frac{\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle}{\langle e_1 \;\; op \;\; e_2, s\rangle \longrightarrow \langle e_1' \;\; op \;\; e_2, s'\rangle}$$

Suppose $\Phi(e_1, s, e_1', s')$ (*) and $\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle$. Have to show $\Phi(e_1 \;\; op \;\; e_2, s, e_1' \;\; op \;\; e_2,$ Take arbitrary Γ, T. Suppose $\Gamma \vdash e_1 \;\; op \;\; e_2{:}T$ and $\mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(\Gamma)$ (**).

case $op = +$. The last rule in the derivation of $\Gamma \vdash e_1 + e_2{:}T$ must have been (op+), so must have $T = \mathsf{int}$, $\Gamma \vdash e_1{:}\mathsf{int}$ (***) and $\Gamma \vdash e_2{:}\mathsf{int}$ (****). By the induction hypothesis (*), (**), and (***) we have $\Gamma \vdash e_1'{:}\mathsf{int}$. By the (op+) rule $\Gamma \vdash e_1' + e_2{:}T$.

case $op = \geq$. Similar.

Case s (op2) (deref), (assign1), (assign2), (seq1), (seq2), (if1), (if2), (if3), (while). Left as exercises.

Theorem 4 (Safety) *If* $\Gamma \vdash e{:}T$, $dom(\Gamma) \subseteq dom(s)$, *and* $\langle e, s\rangle \longrightarrow^* \langle e', s'\rangle$ *then either* e' *is a value or there exist* e'', s'' *such that* $\langle e', s'\rangle \longrightarrow \langle e'', s''\rangle$.

Proof Hint: induction along $\longrightarrow^*$ using the previous results.

Theorem 7 (Uniqueness of typing) *If* $\Gamma \vdash e{:}T$ *and* $\Gamma \vdash e{:}T'$ *then* $T = T'$. The proof is left as Exercise 19.

Theorem 5 (Decidability of typeability) *Given* Γ, e, *one can decide* $\exists\, T.\Gamma \vdash e{:}T$.

Theorem 6 (Decidability of type checking) *Given* Γ, e, T, *one can decide* $\Gamma \vdash e{:}T$.

Proof The implementation gives a type inference algorithm, which, *if correct*, and together with Uniqueness, implies both of these results.

Proving Progress

Theorem 2 (Progress) *If* $\Gamma \vdash e{:}T$ *and* $dom(\Gamma) \subseteq dom(s)$ *then either* e *is a value or there exist* e', s' *such that* $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$.

Proof Take

$$\Phi(\Gamma, e, T) \stackrel{\text{def}}{=} \forall\, s.\ \mathsf{dom}(\Gamma) \subseteq \mathsf{dom}(s) \Rightarrow \mathsf{value}(e) \vee (\exists\, e', s'.\langle e, s\rangle \longrightarrow \langle e', s'\rangle)$$

We show that for all Γ, e, T, if $\Gamma \vdash e{:}T$ then $\Phi(\Gamma, e, T)$, by rule induction on the definition of $\vdash$.

Principle of Rule Induction (variant form): to prove $\Phi(a)$ for all a in the set S_R defined by the rules, it's enough to prove that for each rule instance

$$\frac{h_1 \quad .. \quad h_k}{c}$$

if $\Phi(h_1) \wedge ... \wedge \Phi(h_k) \wedge h_1 \in S_R \wedge .. \wedge h_k \in S_R$ then $\Phi(c)$.

Instantiating to the L1 typing rules, have to show:

(int) $\forall\, \Gamma, n.\Phi(\Gamma, n, \mathsf{int})$

(deref) $\forall\, \Gamma, \ell.\Gamma(\ell) = \mathsf{intref} \Rightarrow \Phi(\Gamma, !\ell, \mathsf{int})$

(op +) $\forall\, \Gamma, e_1, e_2.(\Phi(\Gamma, e_1, \mathsf{int}) \wedge \Phi(\Gamma, e_2, \mathsf{int}) \wedge \Gamma \vdash e_1{:}\mathsf{int} \wedge \Gamma \vdash e_2{:}\mathsf{int}) \Rightarrow \Phi(\Gamma, e_1 + e_2, \mathsf{int})$

(seq) $\forall\, \Gamma, e_1, e_2, T.(\Phi(\Gamma, e_1, \mathsf{unit}) \wedge \Phi(\Gamma, e_2, T) \wedge \Gamma \vdash e_1{:}\mathsf{unit} \wedge \Gamma \vdash e_2{:}T) \Rightarrow \Phi(\Gamma, e_1; e_2, T)$

etc.

$$\Phi(\Gamma, e, T) \stackrel{\text{def}}{=} \forall\, s.\ \mathsf{dom}(\Gamma) \subseteq \mathsf{dom}(s) \Rightarrow \mathsf{value}(e) \vee (\exists\, e', s'.\langle e, s\rangle \longrightarrow \langle e', s'\rangle)$$

Case (op+). Recall the rule

$$(\text{op} +)\quad \frac{\Gamma \vdash e_1{:}\mathsf{int} \qquad \Gamma \vdash e_2{:}\mathsf{int}}{\Gamma \vdash e_1 + e_2{:}\mathsf{int}}$$

Suppose $\Phi(\Gamma, e_1, \mathsf{int})$, $\Phi(\Gamma, e_2, \mathsf{int})$, $\Gamma \vdash e_1{:}\mathsf{int}$, and $\Gamma \vdash e_2{:}\mathsf{int}$. We have to show $\Phi(\Gamma, e_1 + e_2, \mathsf{int})$.

Consider an arbitrary s. Assume $\mathsf{dom}(\Gamma) \subseteq \mathsf{dom}(s)$.

Now $e_1 + e_2$ is not a value, so we have to show $\exists\langle e', s'\rangle.\langle e_1 + e_2, s\rangle \longrightarrow \langle e', s'\rangle$.

Using $\Phi(\Gamma, e_1, \mathsf{int})$ and $\Phi(\Gamma, e_2, \mathsf{int})$ we have:

case e_1 reduces. Then $e_1 + e_2$ does, using (op1).

case e_1 is a value but e_2 reduces. Then $e_1 + e_2$ does, using (op2).

case Both e_1 and e_2 are values. Want to use:

$$\boxed{(\text{op}+) \quad \langle n_1 + n_2, s\rangle \longrightarrow \langle n, s\rangle \quad \text{if } n = n_1 + n_2}$$

Lemma 5 *for all* Γ, e, T, *if* $\Gamma \vdash e{:}T$, e *is a value and* $T = \mathsf{int}$ *then for some* $n \in \mathbb{Z}$ *We have* $e = n$.

We assumed (the variant rule induction principle) that $\Gamma \vdash e_1{:}\mathsf{int}$ and $\Gamma \vdash e_2{:}\mathsf{int}$, so using this Lemma have $e_1 = n_1$ and $e_2 = n_2$.

Then $e_1 + e_2$ reduces, using rule (op+).

All the other cases are in the notes.

Having proved those 10 things, consider an example $\Gamma \vdash (!l + 2) + 3{:}\mathsf{int}$. To see why $\Phi(\Gamma,\ (!l + 2) + 3,\ \mathsf{int})$ holds:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash !l{:}\mathsf{int}}\ (\text{deref}) \quad \dfrac{}{\Gamma \vdash 2{:}\mathsf{int}}\ (\text{int})}{\Gamma \vdash (!l + 2){:}\mathsf{int}}\ (\text{op}+) \quad \dfrac{}{\Gamma \vdash 3{:}\mathsf{int}}\ (\text{int})}{\Gamma \vdash (!l + 2) + 3{:}\mathsf{int}}\ (\text{op}+)$$

Proving Determinacy

Theorem 1 (Determinacy) *If* $\langle e, s\rangle \longrightarrow \langle e_1, s_1\rangle$ *and* $\langle e, s\rangle \longrightarrow \langle e_2, s_2\rangle$ *then* $\langle e_1, s_1\rangle = \langle e_2, s_2\rangle$.

Take

$$\begin{aligned}\Phi(e) \quad \stackrel{\text{def}}{=} \quad & \forall\, s, e', s', e'', s''.\\ & (\langle e, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle e, s\rangle \longrightarrow \langle e'', s''\rangle)\\ & \Rightarrow \langle e', s'\rangle = \langle e'', s''\rangle\end{aligned}$$

We show $\forall\ e\ \in\ L_1.\Phi(e)$ by structural induction.

Principle of Structural Induction: to prove $\Phi(e)$ for all expressions e of L1, it's enough to prove for each tree constructor c that if Φ holds for the subtrees $e_1, .., e_k$ then Φ holds for the tree $c(e_1, .., e_k)$.

Instantiating to the L1 grammar, have to show:

nullary:	$\Phi(\textbf{skip})$
	$\forall\ b\ \in \{\textbf{true}, \textbf{false}\}.\Phi(b)$
	$\forall\ n\ \in\ \mathbb{Z}.\Phi(n)$
	$\forall\ \ell\ \in\ \mathbb{L}.\Phi(!\ell)$
unary:	$\forall\ \ell\ \in\ \mathbb{L}.\forall\ e.\Phi(e) \Rightarrow \Phi(\ell := e)$
binary:	$\forall\ op\ .\forall\ e_1, e_2.(\Phi(e_1) \wedge \Phi(e_2)) \Rightarrow \Phi(e_1\ \ op\ \ e_2)$
	$\forall\ e_1, e_2.(\Phi(e_1) \wedge \Phi(e_2)) \Rightarrow \Phi(e_1; e_2)$
	$\forall\ e_1, e_2.(\Phi(e_1) \wedge \Phi(e_2)) \Rightarrow \Phi(\textbf{while}\ \ e_1\ \ \textbf{do}\ \ e_2)$
ternary:	$\forall\ e_1, e_2, e_3.(\Phi(e_1) \wedge \Phi(e_2) \wedge \Phi(e_3)) \Rightarrow \Phi(\textbf{if}\ \ e_1\ \ \textbf{then}\ \ e_2\ \ \textbf{else}\ \ e_3)$

(op +) $\langle n_1 + n_2, s\rangle \longrightarrow \langle n, s\rangle$ if $n = n_1 + n_2$

(op ≥) $\langle n_1 \geq n_2, s\rangle \longrightarrow \langle b, s\rangle$ if $b = (n_1 \geq n_2)$

(op1) $$\frac{\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle}{\langle e_1 \;\; op \;\; e_2, s\rangle \longrightarrow \langle e_1' \;\; op \;\; e_2, s'\rangle}$$

(op2) $$\frac{\langle e_2, s\rangle \longrightarrow \langle e_2', s'\rangle}{\langle v \;\; op \;\; e_2, s\rangle \longrightarrow \langle v \;\; op \;\; e_2', s'\rangle}$$

(deref) $\langle !\ell, s\rangle \longrightarrow \langle n, s\rangle$ if $\ell \in \mathrm{dom}(s)$ and $s(\ell) = n$

(assign1) $\langle \ell := n, s\rangle \longrightarrow \langle \mathbf{skip}, s + \{\ell \mapsto n\}\rangle$ if $\ell \in \mathrm{dom}(s)$

(assign2) $$\frac{\langle e, s\rangle \longrightarrow \langle e', s'\rangle}{\langle \ell := e, s\rangle \longrightarrow \langle \ell := e', s'\rangle}$$

(seq1) $\langle \mathbf{skip}; e_2, s\rangle \longrightarrow \langle e_2, s\rangle$

(seq2) $$\frac{\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle}{\langle e_1; e_2, s\rangle \longrightarrow \langle e_1'; e_2, s'\rangle}$$

(if1) $\langle \mathbf{if}\ \mathbf{true}\ \mathbf{then}\ e_2\ \mathbf{else}\ e_3, s\rangle \longrightarrow \langle e_2, s\rangle$

(if2) $\langle \mathbf{if}\ \mathbf{false}\ \mathbf{then}\ e_2\ \mathbf{else}\ e_3, s\rangle \longrightarrow \langle e_3, s\rangle$

(if3) $$\frac{\langle e_1, s\rangle \longrightarrow \langle e_1', s'\rangle}{\langle \mathbf{if}\ e_1\ \mathbf{then}\ e_2\ \mathbf{else}\ e_3, s\rangle \longrightarrow \langle \mathbf{if}\ e_1'\ \mathbf{then}\ e_2\ \mathbf{else}\ e_3, s'\rangle}$$

(while) $\langle \mathbf{while}\ e_1\ \mathbf{do}\ e_2, s\rangle \longrightarrow \langle \mathbf{if}\ e_1\ \mathbf{then}\ (e_2; \mathbf{while}\ e_1\ \mathbf{do}\ e_2)\ \mathbf{else}\ \mathbf{skip}, s\rangle$

$$\Phi(e) \stackrel{\text{def}}{=} \forall\, s, e', s', e'', s''.\ (\langle e, s\rangle \longrightarrow \langle e', s'\rangle \wedge \langle e, s\rangle \longrightarrow \langle e'', s''\rangle) \Rightarrow \langle e', s'\rangle = \langle e'', s''\rangle$$

(assign1) $\langle \ell := n, s\rangle \longrightarrow \langle \mathbf{skip}, s + \{\ell \mapsto n\}\rangle$ if $\ell \in \mathrm{dom}(s)$

(assign2) $$\frac{\langle e, s\rangle \longrightarrow \langle e', s'\rangle}{\langle \ell := e, s\rangle \longrightarrow \langle \ell := e', s'\rangle}$$

Lemma: Values don't reduce

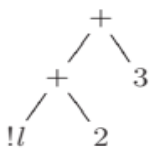
It's handy to have this lemma:

Lemma 6 *For all* $e \in L_1$, *if* e *is a value then* $\forall\, s. \neg\, \exists e', s'. \langle e, s\rangle \longrightarrow \langle e', s'\rangle$.

Proof By defn e is a value if it is of one of the forms n, b, **skip**. By examination of the rules on slides ..., there is no rule with conclusion of the form $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ for e one of n, b, **skip**.

All the other cases are in the notes.

Having proved those 9 things, consider an example $(!l + 2) + 3$. To see why $\Phi((!l + 2) + 3)$ holds:



Summarising Proof Techniques	
Determinacy	structural induction for e
Progress	rule induction for $\Gamma \vdash e : T$
Type Preservation	rule induction for $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$
Safety	mathematical induction on $\longrightarrow^k$
Uniqueness of typing	...
Decidability of typability	exhibiting an algorithm
Decidability of checking	corollary of other results

Inductive Definitions, More Formally (optional)

Here we will be more precise about inductive definitions and rule induction. Following this may give you a sharper understanding, but it is not itself examinable. To make an *inductive definition* of a particular subset of a set A, take a set R of some concrete rule instances, each of which is a pair (H, c) where H is a finite subset of A (the hypotheses) and c is an element of A (the conclusion).

Consider finite trees labelled by elements of A for which every step is in R, eg

$$\dfrac{\dfrac{}{a_1} \quad \dfrac{\dfrac{}{a_3}}{a_2}}{a_0}$$

where $(\{\}, a_1)$, $(\{\}, a_3)$, $(\{a_3\}, a_2)$, and $(\{a_1, a_2\}, a_0)$ all elements of R.

The subset S_R of A *inductively defined* by the rule intances R is the set of $a \in A$ such that there is such a proof with root node labelled by a.

For the definition of the transition relation:

- Start with $A = \text{expr} * \text{store} * \text{expr} * \text{store}$
- We define $\longrightarrow \subseteq A$ (write infix, e.g.$\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ instead of $(e, s, e', s') \in \longrightarrow$).
- The rule instances R are the concrete rule instances of the transition rules.

For the definition of the typing relation:

- Start with $A = \text{TypeEnv} * \text{expr} * \text{types}$.
- We define $\vdash \subseteq A$ (write mixfix, e.g.$\Gamma \vdash e : T$ instead of $(\Gamma, e, T) \in \vdash$).
- The rule instances are the concrete rule instances of the typing rules.

Instead of talking informally about derivations as finite trees, we can regard S_R as a *least fixed point*. Given rules R, define $F_R : P\ A \to P\ A$ by

$$F_R(S) = \{c \mid \exists\, H.(H, c) \in R \wedge H \subseteq S\}$$

($F_R(S)$ is the set of all things you can derive in exactly one step from things in S)

$$\begin{aligned} S_R^0 &= \{\} \\ S_R^{k+1} &= F_R(S_R^k) \\ S_R^\omega &= \textstyle\bigcap_{k \in \mathbb{N}} {S_R}^k \end{aligned}$$

Theorem 9 $S_R = S_R^\omega$.

Say a subset $S \subseteq A$ is *closed under rules* R if $\forall (H, c) \in R.(H \subseteq S) \Rightarrow c \in S$, ie, if $F_R(S) \subseteq S$.

Theorem 10 $S_R = \bigcap\{S \mid S \subseteq A \wedge F_R(S) \subseteq S\}$

This says 'the subset S_R of A inductively defined by R is the smallest set closed under the rules R'. It is the intersection of all of them, so smaller than (or equal to) any of them.

Now, to prove something about an inductively-defined set...

To see why rule induction is sound, using this definition: Saying $\{a \mid \Phi(a)\}$ closed under the rules means exactly $F_R(\{a \mid \Phi(a)\}) \subseteq \{a \mid \Phi(a)\}$, so by Theorem 10 we have $S_R \subseteq \{a \mid \Phi(a)\}$, i.e.$\forall\, a \in S_R.a \in \{a' \mid \Phi(a')\}$, i.e.$\forall\, a \in S_R.\Phi(a)$.

summary

Inductive definitions and proofs by induction are all-pervasive in the structural approach to operational semantics. The familiar (one hopes!) principle of Mathematical Induction and the equivalent Least Number Principle are recalled on Slide 10. Most of the induction techniques we will use can be justified by appealing to Mathematical Induction. Nevertheless, it is convenient to derive from it a number of induction principles more readily applicable to the structures with which we have to deal. This section briefly reviews some of the ideas and techniques; many examples of their use will occur throughout the rest of the course. Apart from the importance of these techniques for the subject, they should be important to you too, for **examination questions on this course assume an ability to give proofs using the various induction techniques**.

Mathematical Induction

For any property $\Phi(x)$ of natural numbers $x \in \mathbb{N} \stackrel{\text{def}}{=} \{0, 1, 2, \ldots\}$, to prove

$$\forall x \in \mathbb{N}.\Phi(x)$$

it suffices to prove

$$\Phi(0) \quad \& \quad \forall x \in \mathbb{N}.\ \Phi(x) \Rightarrow \Phi(x+1).$$

Equivalently:

Least Number Principle: any non-empty subset of $\mathbb{N}$ possesses a least element.