Australian National University

COMP3610/6361 Principles of Programming Languages

Peter Höfner

Aug 1, 2023

Section 4

Proofs (Structural Induction)

Why Proofs

- how do we know that the stated theorems are actually true? intuition is often wrong – we need proof
- proofs strengthen intuition about language features
- examines all the various cases
- can guarantee items such as type safety
- most of our definitions are **inductive**; we use *structural* induction

(Mathematical) Induction

Mathematical induction proves that we can climb as high as we like on a ladder, by proving that we can climb onto the bottom rung (the basis) and that from each rung we can climb up to the next one (the step).

[Concrete Mathematics (1994), R. Graham]

Natural Induction I

A proof by (natural) induction consists of **two cases**.

The **base case** proves the statement for $n = 0$ without assuming any knowledge of other cases.
The **induction step** proves that if the statement holds for any given case $n = k$, then it must also hold for the next case $n = k + 1$.

Natural Induction II

Theorem

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

Proof.

Base case: show $\Phi(0)$
Induction step: $\forall k.\ \Phi(k) \Longrightarrow \Phi(k+1)$
For that we fix an arbitrary k.
Assume $\Phi(k)$ derive $\Phi(k+1)$. $\square$

Example: $0 + 1 + 2 + \cdots + n = \frac{n \cdot (n+1)}{2}$.

Australian National University

Natural Induction III

Theorem

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

Proof.

Base case: show $\Phi(0)$

Induction step: $\forall i, k . 0 \leq i \leq k. \ \Phi(i) \Longrightarrow \Phi(k+1)$

For that we fix an arbitrary k.

Assume $\phi(i)$ *for all* $i \leq k$ derive $\phi(k+1)$. $\square$

Example: $F_n = \frac{\varphi^n - \psi^n}{\varphi - \psi}$,

with F_n is the n-th Fibonacci number, $\varphi = \frac{1+\sqrt{5}}{2}$ (the golden ratio) and $\psi = \frac{1-\sqrt{5}}{2}$.

Structural Induction I

- generalisation of natural induction
- prove that some proposition $\Phi(x)$ holds for all x of some sort of recursively defined structure
- requires well-founded partial order

Examples: lists, formulas, trees

Structural Induction II

Determinacy	structural induction for E
Progress	rule induction for $\Gamma \vdash E : T$
	(induction over the height of derivation tree)
Type Preservation	rule induction for $\langle E\, ,\, s\rangle \longrightarrow \langle E'\, ,\, s'\rangle$
Safety	mathematical induction on $\longrightarrow^n$
Uniqueness of typing	$\ldots$
Decidability of typability	exhibiting an algorithm
Decidability of type checking	corollary of other results

Structural Induction over Expressions

Prove facts about all expressions, e.g. Determinacy?

Theorem (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 $\langle E_1\,,\,s_1\rangle = \langle E_2\,,\,s_2\rangle$.

Do not forget the elided universal quantifiers.

Theorem (Determinacy)

For all E***,*** s***,*** E_1***,*** s_1***,*** E_2 ***and*** 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

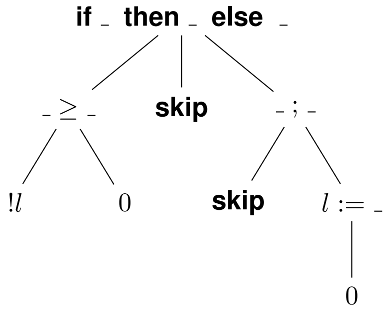
Remember the definition of expressions:

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

This defines an (infinite) set of expressions.

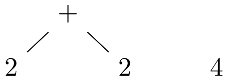
Abstract Syntax Tree I

Example: **if** $!l \geq 0$ **then skip else** ($\textbf{skip}\ ;\ l := 0$)

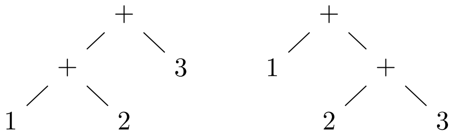


Abstract Syntax Tree II

- equivalent expressions are not the same, e.g., $2 + 2 \neq 4$



- ambiguity, e.g., $(1 + 2) + 3 \neq 1 + (2 + 3)$



Parentheses are only used for disambiguation – they are not part of the grammar

Australian National University

Structural Induction (for abstract syntax)

Theorem

$\forall E \in$ *IMP*. $\Phi(E)$

Proof.

Base case(s): show $\Phi(E)$ for each unary tree constructor (leaf)
Induction step(s): show it for the remaining constructors

$$\forall c.\ \forall E_1, \ldots E_k.\ (\Phi(E_1) \wedge \cdots \wedge \Phi(E_k)) \Longrightarrow \Phi(c(E_1, \ldots, E_k))$$

$\square$

Structural Induction (syntax IMP)

To show $\forall E \in$ IMP. $\Phi(E)$.

base cases

nullary:	$\Phi(\textbf{skip})$
	$\forall b \in \mathbb{B}.\ \Phi(b)$
	$\forall n \in \mathbb{Z}.\ \Phi(n)$
	$\forall l \in \mathbb{L}.\ \Phi(\ !l)$

steps

unary:	$\forall l \in \mathbb{L}.\ \forall E.\ \Phi(E) \Longrightarrow \Phi(l := E)$
binary:	$\forall op.\ \forall E_1, E_2.\ (\Phi(E_1) \wedge \Phi(E_2)) \Longrightarrow \Phi(E_1\ op\ E_2)$
	$\forall E_1, E_2.\ (\Phi(E_1) \wedge \Phi(E_2)) \Longrightarrow \Phi(E_1\ ;\ E_2)$
	$\forall E_1, E_2.\ (\Phi(E_1) \wedge \Phi(E_2)) \Longrightarrow \Phi(\textbf{while}\ E_1\ \textbf{do}\ E_2)$
ternary:	$\forall E_1, E_2, E_3.\ (\Phi(E_1) \wedge \Phi(E_2) \wedge \Phi(E_3))$ $\Longrightarrow \Phi(\textbf{if}\ E_1\ \textbf{then}\ E_2\ \textbf{else}\ E_3)$

Proving Determinacy – Outline

Theorem (Determinacy)

For all** E**,** s**,** E_1**,** s_1**,** E_2 **and** 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 $\langle E_1\,,\,s_1\rangle = \langle E_2\,,\,s_2\rangle$.

Proof.

Choose

$$\Phi(E) \overset{\text{def}}{=} \forall s, E', s', E'', s''. \quad (\langle E\,,\,s\rangle \longrightarrow \langle E'\,,\,s'\rangle \,\wedge\, \langle E\,,\,s\rangle \longrightarrow \langle E''\,,\,s''\rangle) \Longrightarrow \langle E'\,,\,s'\rangle = \langle E''\,,\,s''\rangle$$

and show $\Phi(E)$ by structural induction over E. □

Proving Determinacy – Sketch

Some cases on whiteboard

Proving Determinacy – auxiliary lemma

Values do not reduce.

Lemma

For all $E \in$ IMP, if E is a value then
$\forall s.\ \neg(\exists E', s'.\ \langle E\ ,\ s\rangle \longrightarrow \langle E'\ ,\ s'\rangle)$.

Proof.

- E is a value iff it is of the form n, b, **skip**
- By examination of the rules . . .
  there is no rule with conclusion of the form $\langle E\ ,\ s\rangle \longrightarrow \langle E'\ ,\ s'\rangle$ for E a value

$\square$

Inversion I

In proofs involving inductive definitions. one often needs an *inversion property*.
Given a tuple in one inductively defined relation, gives you a case analysis of the possible "last rule" used.

Lemma (Inversion for $\longrightarrow$)

If $\langle E\ ,\ s\rangle \longrightarrow \langle \hat{E}\ ,\ \hat{s}\rangle$ *then either*

1. *(op$+$): there exists* n_1, n_2 *and* n *such that* $E = n_1\ op\ n_2$, $\hat{E} = n$, $\hat{s} = s$ *and* $n = n_1 + n_2$,
   (Note: $+$s have different meanings in this statements), or
2. *(op1): there exists* E_1, E_2, op *and* E_1' *such that* $E = E_1\ op\ E_2$, $\hat{E} = E_1'\ op\ E_2$ *and* $\langle E_1\ ,\ s\rangle \longrightarrow \langle E_1'\ ,\ s'\rangle$, *or*
3. *...*

Inversion II

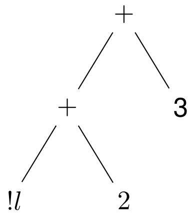
Lemma (Inversion for $\vdash$)

If $\Gamma \vdash E : T$ *then either*

- $\ldots$

Determinacy – Intuition

The intuition behind structural induction over expressions. Consider $(\,!l + 2) + 3$. How can we see that $\Phi((\,!l + 2) + 3)$ holds?



Rule Induction

How to prove the following theorems?

Theorem (Progress)

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

Theorem (Type Preservation)

If $\Gamma \vdash E : T$*, 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')*.*

Inductive Definition of $\longrightarrow$

What does $\langle E \, , \, s \rangle \longrightarrow \langle E' \, , \, s' \rangle$ actually mean?

We defined the transition relation by providing some rules, such as

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

(op1) $$\frac{\langle E_1 \, , \, s \rangle \longrightarrow \langle E_1' \, , \, s' \rangle}{\langle E_1 \; op \; E_2 \, , \, s \rangle \longrightarrow \langle E_1' \; op \; E_2 \, , \, s' \rangle}$$

These rules (their concrete instances) inductively/recursively define a set of derivation trees. The last step in the derivation tree defines a step in the transition system.
We define the (infinite) set of all finite derivation trees

Derivation Tree (Transition Relation) – Example

$$\cfrac{\cfrac{\cfrac{}{\langle 2+2\,,\,\{\}\rangle \longrightarrow \langle 4\,,\,\{\}\rangle}\ (\textsc{op}+)}{\langle (2+2)+3\,,\,\{\}\rangle \longrightarrow \langle 4+3\,,\,\{\}\rangle}\ (\textsc{op}1)}{\langle (2+2)+3 \geq 5\,,\,\{\}\rangle \longrightarrow \langle 4+3 \geq 5\,,\,\{\}\rangle}\ (\textsc{op}1)$$

Derivation Tree (Typing Judgement) – Example

$$
\cfrac{
  \cfrac{
    \cfrac{\Gamma(l) = \textsf{intref}}{\Gamma \vdash !l : \textsf{int}}\ (\textsc{derref})
    \qquad
    \cfrac{}{\Gamma \vdash 2 : \textsf{int}}\ (\textsc{int})
  }{\Gamma \vdash\ !l + 2 : \textsf{int}}\ (\textsc{op+})
  \qquad
  \cfrac{}{\Gamma \vdash 3 : \textsf{int}}\ (\textsc{int})
}{\Gamma \vdash (!l + 2) + 3 : \textsf{int}}\ (\textsc{op+})
$$

Principle of Rule Induction I

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 is enough to prove that $\{a \mid \Phi(a)\}$ is closed under the rules, i.e., for each

$$\frac{h_1 \ \ \ldots h_k}{c}$$

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

Principle of Rule Induction II

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 is enough to prove that for each

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

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

Proving Progress I

Theorem (Progress)

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

Proof.

Choose

$$\begin{aligned}\Phi(\Gamma, E, T) = \ & \forall s.\ \mathsf{dom}(\Gamma) \subseteq \mathsf{dom}(s) \\ & \qquad \Longrightarrow \mathsf{value}(E) \vee (\exists E', s'.\ \langle E\,,\ s\rangle \longrightarrow \langle E'\,,\ s'\rangle)\end{aligned}$$

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 $\vdash$. □

Australian National University

Proving Progress II

Rule induction for our typing rules means:

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

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

(op+) $\forall \Gamma, E_1, E_2.\ \big(\Phi(\Gamma, E_1, \mathsf{int}) \wedge \Phi(\Gamma, E_2, \mathsf{int}) \wedge \Gamma \vdash E_1 : \mathsf{int} \wedge \Gamma \vdash E_2 : \mathsf{int}\big)$
$\Longrightarrow \Phi(\Gamma, E_1 + E_2, \mathsf{int})$

(seq) $\forall \Gamma, E_1, E_2.\ \big(\Phi(\Gamma, E_1, \mathsf{unit}) \wedge \Phi(\Gamma, E_2, T) \wedge \Gamma \vdash E_1 : \mathsf{unit} \wedge \Gamma \vdash E_2 : T\big)$
$\Longrightarrow \Phi(\Gamma, E_1; E_2, \mathsf{int})$

…[10 rules in total]

Proving Progress III

$$\begin{aligned}\Phi(\Gamma, E, T) = \ & \forall s.\ \mathsf{dom}(\Gamma) \subseteq \mathsf{dom}(s) \\ & \qquad \Longrightarrow \mathsf{value}(E) \vee (\exists E', s'.\ \langle E\ ,\ s\rangle \longrightarrow \langle E'\ ,\ s'\rangle)\end{aligned}$$

Case (op$+$):

$$\text{(op+)} \quad \frac{\Gamma \vdash E_1 : \mathsf{int} \qquad \Gamma \vdash E_2 : \mathsf{int}}{\Gamma \vdash E_1 + E_2 : \mathsf{int}}$$

- assume $\Phi(\Gamma, E_1, \mathsf{int})$, $\Phi(\Gamma, E_2, \mathsf{int})$, $\Gamma \vdash E_1 : \mathsf{int}$ and $\Gamma \vdash E_2 : \mathsf{int}$
- we have to show $\Phi(\Gamma, E_1 + E_2, \mathsf{int})$
- assume an arbitrary but fixed s, and $\mathsf{dom}(\Gamma) \subseteq \mathsf{dom}(s)$
- $E_1 + E_2$ is not a value; hence we have to show

$$\exists E', s'.\ \langle E_1 + E_2\ ,\ s\rangle \longrightarrow \langle E'\ ,\ s'\rangle$$

Proving Progress IV

Case (op$+$) (cont'd)**:**

- we have to show

$$\exists E', s'.\ \langle E_1 + E_2\ ,\ s\rangle \longrightarrow \langle E'\ ,\ s'\rangle$$

- 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, by (op1).
  case E_1 is a value and E_2 reduces. Then $E_1 + E_2$ does, by (op2).
  case E_1 and E_2 are values; we want to use

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

  we assumed $\Gamma \vdash E_1 : \mathsf{int}$ and $\Gamma \vdash E_2 : \mathsf{int}$ we need $E_1 = n_1$ and $E_2 = n_2$; then $E_1 + E_2$ reduces, by (op$+$).

Australian National University

Proving Progress V

Lemma

For all Γ, E, T, *if* $\Gamma \vdash E : T$ *is a value with* $T =$ *int* *then there exists* $n \in \mathbb{Z}$ *with* $E = n$.

Derivation Tree (Typing Judgement) – Example

$$\dfrac{\dfrac{\dfrac{\Gamma(l) = \mathsf{intref}}{\Gamma \vdash\ !l : \mathsf{int}}\ (\textsc{deref}) \qquad \dfrac{}{\Gamma \vdash 2 : \mathsf{int}}\ (\textsc{int})}{\Gamma \vdash\ !l + 2 : \mathsf{int}}\ (\textsc{op+}) \qquad \dfrac{}{\Gamma \vdash 3 : \mathsf{int}}\ (\textsc{int})}{\Gamma \vdash (\ !l + 2) + 3 : \mathsf{int}}\ (\textsc{op+})$$

Which Induction Principle to Use?

- matter of convenience (all equivalent)
- use an induction principle that matches the definitions

Structural Induction (Repetition)

Determinacy	structural induction for E
Progress	rule induction for $\Gamma \vdash E : T$
	(induction over the height of derivation tree)
Type Preservation	rule induction for $\langle E\ ,\ s\rangle \longrightarrow \langle E'\ ,\ s'\rangle$
Safety	mathematical induction on $\longrightarrow^n$
Uniqueness of typing	$\ldots$
Decidability of typability	exhibiting an algorithm
Decidability of type checking	corollary of other results

Australian National University

Why care about Proofs?

1. sometimes it seems hard or pointless to prove things because they are ‘obvious’, . . .
   (in particular with our language)
2. proofs illustrate (and explain) why ‘things are obvious’
3. sometimes the obvious facts are false . . .
4. sometimes the obvious facts are not obvious at all
   (in particular for ‘real’ languages)
5. sometimes a proof contains or suggests an algorithm that you need
   (proofs that type inference is decidable (for fancier type systems))
6. force a clean language design