Proof Techniques in Operational Semantics

Lecture 3

Adapted in part from Winskel, and slides by George Necula.
Structural Induction

Let’s try an exercise:
- Prove that execution of IMP is deterministic, i.e.

\[ \forall \sigma . (\langle P, \sigma \rangle \rightarrow \sigma_1 \land \langle P, \sigma \rangle \rightarrow \sigma_2) \Rightarrow \sigma_1 = \sigma_2 \]

Where do we start?
- we know how to do induction on the structure of expressions
- we can start there
Expression Evaluation is Deterministic

We want to show that

$$\forall \sigma . ((e, \sigma) \rightarrow n_1 \land (e, \sigma) \rightarrow n_2) \Rightarrow n_1 = n_2$$

Base case: e=N
- Only one rule applies $\langle N, \sigma \rangle \rightarrow n$
- It’s clear that we will always get the same $n$

Inductive case $e = e_1 + e_2$

$$\frac{(e_1, \sigma) \rightarrow n_1^1 \quad (e_2, \sigma) \rightarrow n_2^2 \quad n_1 = n_1^1 + n_2^2}{(e_1 + e_2, \sigma) \rightarrow n_1}$$

$$\frac{(e_1, \sigma) \rightarrow n_1^1 \quad (e_2, \sigma) \rightarrow n_2^2 \quad n_2 = n_1^1 + n_2^2}{(e_1 + e_2, \sigma) \rightarrow n_2}$$

- By the inductive hypothesis, $n_1^1 = n_2^1$ \textit{and} $n_1^2 = n_2^2$ so it follows that $n_1 = n_2$
How about commands?

We can’t apply the same principle
- The problem is the while loop
  \[
  \langle e_1, \sigma \rangle \to \text{true} \quad \langle c, \sigma \rangle \to \sigma'' \quad \langle \text{while } e_1 \text{ then } c, \sigma'' \rangle \to \sigma' \quad \langle \text{while } e_1 \text{ then } c, \sigma \rangle \to \sigma'
  \]
- The problem is the same while is in both the premise and the conclusion,
- so we can’t use the inductive hypothesis to discharge the premise

We need a more powerful form of induction
Visually intuitive but can get a little clunky

\[ \lambda \]
\[ \frac{u \chi \ldots z \chi \ldots}{\ldots \ldots \ldots \ldots} \]

And derivation trees as

\[ \lambda \]
\[ \frac{u \chi \ldots z \chi \ldots}{\ldots \ldots \ldots \ldots} \]

We are used to seeing rule instances as

Some notation
Some notation

Let $X = \{x_1, x_2, \ldots x_n\}$ then,
- the rule $\frac{x_1 \ x_2 \ \ldots \ x_n}{y}$
- can be represented as $(X/y) = (\{x_1, x_2, \ldots x_n\} /y)$
- If the rule has no premises, $X$ will be empty

Definition: R-derivation of $y$
- It can be a rule instance with empty premises $(\emptyset /y)$
- Or, a pair $(\{d_1, \ldots, d_n\}/y)$ where there is a rule of the form $(\{x_1, \ldots x_n\} /y)$, and each $d_i$ is an R-derivation of $x_i$
- So an R-derivation is just a flat version of a derivation tree

Winskel uses the notation $d \vdash^-_R y$
Why new notation

This notation makes it clear that derivations are recursively generated from a set of rules.

Let $d$ be a derivation of a given judgement

$$d \vdash_K \langle c, \sigma \rangle \rightarrow \sigma'$$

By definition, the derivation $d$ will contain a finite number of sub-derivations.

- otherwise we can’t claim that the judgment holds

We can define a well founded order on the sub-derivations that make up $d$!
Induction on the Structure of Derivations

Let $d_1$ and $d_2$ be derivations such that
- $d_1 = (D/y)$ where $D=\{d_1, \ldots, d_n\}$ and
- $d_2 \in D$

Then we say that $d_2 \prec d_1$

Note that if $d \vdash_R (c, \sigma) \rightarrow \sigma'$, then $\prec$ is a well-founded order on the sub-derivations of $d$

So how do we use this to construct proofs?
Command evaluation is deterministic

We want to show that

\[ \forall \sigma. (\langle c, \sigma \rangle \rightarrow \sigma_1 \land \langle c, \sigma \rangle \rightarrow \sigma_2) \Rightarrow \sigma_1 = \sigma_2 \]

In order to prove this by induction on the structure of derivations, our inductive hypothesis will be as follows

\[ P(d) := \forall c, \sigma, \sigma_1, \sigma_2 . (d \vdash_R \langle c, \sigma \rangle \rightarrow \sigma_1 \land \langle c, \sigma \rangle \rightarrow \sigma_2) \Rightarrow \sigma_1 = \sigma_2 \]

Note that if \( \langle c, \sigma \rangle \rightarrow \sigma_2 \) then there must be a \( d' \) such that

\[ d' \vdash_R \langle c, \sigma \rangle \rightarrow \sigma_2 \]
Command evaluation is deterministic

Base case

\[ d = d' = \frac{\langle \text{skip}, \sigma \rangle \rightarrow \sigma}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma} \]

Inductive cases

X:=e

\[ d = \frac{\langle e, \sigma \rangle \rightarrow n_1}{\langle X := e, \sigma \rangle \rightarrow \sigma[X = n_1]} \quad d' = \frac{\langle e, \sigma \rangle \rightarrow n_2}{\langle X := e, \sigma \rangle \rightarrow \sigma[X = n_2]} \]

- we already showed n1 = n2, so we don’t even need the inductive hypothesis
Command evaluation is deterministic

Inductive cases
c1; c2

\[ d = \frac{d_1 = \langle c_1, \sigma \rangle \rightarrow \sigma_1''}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma_1} \quad d_2 = \frac{\langle c_2, \sigma_1'' \rangle \rightarrow \sigma_1}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma_1} \quad d' = \frac{\langle c_1, \sigma \rangle \rightarrow \sigma_2''}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma_2} \quad \frac{\langle c_2, \sigma_2'' \rangle \rightarrow \sigma_2}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma_2} \]

- Note that \( d_1 < d \) and \( d_2 < d \), so we can assume our IH for both of them

\[ P(d) := \forall c, \sigma, \sigma_1, \sigma_2 . (d \models_R \langle c, \sigma \rangle \rightarrow \sigma_1 \land \langle c, \sigma \rangle \rightarrow \sigma_2) \Rightarrow \sigma_1 = \sigma_2 \]

- Applying the IH to \( d_1 \) tells us that \( \sigma_1'' = \sigma_2'' \)
- That, together with the IH on \( d_2 \) tells us that \( \sigma_1 = \sigma_2 \)
Command evaluation is deterministic

The same strategy applies to all other cases

But there are some caveats

What we proved:

$$\forall \sigma . (\langle P, \sigma \rangle \rightarrow \sigma_1 \land \langle P, \sigma \rangle \rightarrow \sigma_2) \Rightarrow \sigma_1 = \sigma_2$$

- Notice one KEY assumption is that the judgments $\langle P, \sigma \rangle \rightarrow \sigma_1$ and $\langle P, \sigma \rangle \rightarrow \sigma_2$ are valid.

- This is only true if the program terminates!
  - Otherwise, all bets are off

- So what we really proved is that IF the program terminates, its execution is deterministic
Rule Induction

Think of it as a generalization of structural induction

Let \( R = \{ \{x_0, \ldots, x_i\}/y\}_i \) be a set of rules
- given a set of elements \( \{x_0, \ldots, x_i\} \), the rule produces an element \( y \)
- rules of the form \( (\emptyset / y) \) are allowed too

Examples
- \( R \) can be the set of derivation rules in a Big Step OS
  - in that case, the elements \( x_i \) and \( y \) will be judgments
  - rules of the form \( (\emptyset / y) \) correspond to derivation rules with no premises
- \( R \) can also be the global rules in a Small Step OS
  - in that case, the elements \( x_i \) and \( y \) will be configurations
  - initial configurations \( y \) correspond to rules of the form \( (\emptyset / y) \)
- \( R \) could also be functions that produce outputs from an input
  - functions with no input parameters \( (\emptyset / y) \) correspond to constants
Rule Induction

Given a set of rules $R$, we can define the set $I_R$ as the set of elements that can be derived through the rules in $R$

$I_R$ is defined from $R$ as the smallest set such that:

- for all rules $(\emptyset/y) \in R$, $y \in I_R$
- if $\{x_0, \ldots, x_i\}/y) \in R$ and $\{x_0, \ldots, x_i\} \subseteq I_R$ then $y \in I_R$

Examples

- If $R$ are derivation rules in Big Step OS, $I_R$ is the set of valid judgments
- If $R$ are rules in small step OS, $I_R$ is the set of valid intermediate configurations that can be derived from the initial configurations
General Principle of Rule Induction

Given a property $P(x)$, we want to prove that it holds for all elements $x \in I_R$

$$\forall x \in I_R \, P(x)$$

Then, it is sufficient to prove that

$$\forall (\{x_0, \ldots, x_i\}/y) \in R \left( \forall x_j \in \{x_0, \ldots, x_i\}, \, x_j \in I_R \land P(x_j) \right) \Rightarrow P(y)$$
Example

Applying rule induction with Big Step OS
- For notational convenience, we are going to use \((c, \sigma, \sigma')\) to refer to the judgement \(<c, \sigma \to \sigma'\) and \((e, \sigma, n)\) to refer to the judgement \(<e, \sigma \to n\)

Then, the following is a complete set of rules for IMP

<table>
<thead>
<tr>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset / (N, \sigma, n))</td>
</tr>
<tr>
<td>(\emptyset / (x, \sigma, \sigma(x)))</td>
</tr>
<tr>
<td>({(e_1, \sigma, n_1), (e_1, \sigma, n_2)} / (e_1 \text{ op } e_2, \sigma, n_1 \text{ op } n_2)) \text{ where op is some operator}</td>
</tr>
<tr>
<td>({(e, \sigma, n)} / (x := e, \sigma, \sigma[x = n]))</td>
</tr>
<tr>
<td>(\emptyset / (\text{skip}, \sigma, \sigma))</td>
</tr>
<tr>
<td>({(e, \sigma, \text{true}), (c_t, \sigma, \sigma')} / (\text{if } e \text{ then } c_t \text{ else } c_f, \sigma, \sigma'))</td>
</tr>
<tr>
<td>({(e, \sigma, \text{false}), (c_f, \sigma, \sigma')} / (\text{if } e \text{ then } c_t \text{ else } c_f, \sigma, \sigma'))</td>
</tr>
<tr>
<td>({(c_1, \sigma, \sigma''), (c_2, \sigma'', \sigma')} / (c_1; c_2, \sigma, \sigma'))</td>
</tr>
<tr>
<td>({(e, \sigma, \text{false})} / (\text{while } e \text{ do } c, \sigma, \sigma))</td>
</tr>
<tr>
<td>({(e, \sigma, \text{true}), (c, \sigma, \sigma''), (\text{while } e \text{ do } c, \sigma'', \sigma')} / (\text{while } e \text{ do } c, \sigma, \sigma'))</td>
</tr>
</tbody>
</table>
Example

Consider the following program we’ll call W:

\[
W ::= \text{while } x \neq 5 \text{ do } x := x + 1
\]

We want to show that the loop will only terminate when \( x \leq 5 \), and in that case, \( x \) will end up with the value 5

\[
\forall \sigma, \sigma' \ (W, \sigma) \rightarrow \sigma' \Rightarrow \sigma(x) \leq 5 \land \sigma' = \sigma[x = 5]
\]

To prove this using rule induction, we’ll prove that the following proposition holds for all judgments in \( I_R \)

\[
P(s) = \begin{cases} 
  s = (W, \sigma, \sigma') & \sigma(x) \leq 5 \land \sigma' = \sigma[x = 5] \\
  \text{else} & \\
  \text{true} & 
\end{cases}
\]
Example

So by rule induction, what we have to prove is

\[ \forall ( (x_0, \ldots, x_l)/y) \in R \left( \forall x_j \in \{x_0, \ldots, x_l\} \ x_j \in I_R \land P(x_j) \right) \Rightarrow P(y) \]

Where R is the set of rules shown below

- Note that the only rules that matter are those that involve the while loop; P(y) will be trivially satisfied for all other rules.

<table>
<thead>
<tr>
<th>( \emptyset ) / (N, ( \sigma ), n)</th>
<th>( \emptyset ) / (x, ( \sigma ), ( \sigma(x) ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>{e_1, ( \sigma ), n_1}, {e_1, ( \sigma ), n_2} / (e_1 \ op \ e_2, ( \sigma ), n_1 \ op \ n_2) \ where \ op \ is \ some \ operation</td>
<td></td>
</tr>
<tr>
<td>{e, ( \sigma ), n} / (x:=e, ( \sigma ), ( \sigma[x=n] ))</td>
<td>( \emptyset ) / (skip, ( \sigma ), ( \sigma ))</td>
</tr>
<tr>
<td>{e, ( \sigma ), true}, {ct, ( \sigma ), ( \sigma' }} / (if \ e \ then \ ct \ else \ cf, ( \sigma ), ( \sigma' ))</td>
<td>{e, ( \sigma ), false}, {cf, ( \sigma ), ( \sigma' }} / (if \ e \ then \ ct \ else \ cf, ( \sigma ), ( \sigma' ))</td>
</tr>
<tr>
<td>{c_1, ( \sigma'' ), c_2, ( \sigma'' ), ( \sigma' )} / (c_1;c_2, ( \sigma ), ( \sigma' ))</td>
<td>{e, ( \sigma ), false} / (while \ e \ do \ c, ( \sigma ), ( \sigma ))</td>
</tr>
<tr>
<td>{e, ( \sigma ), true}, {c, ( \sigma ), ( \sigma'' )}, (while \ e \ do \ c, ( \sigma'' ), ( \sigma' )} / (while \ e \ do \ c, ( \sigma ), ( \sigma' ))</td>
<td></td>
</tr>
</tbody>
</table>
Example

In order to complete the proof, we need to prove

\[ W ::= \text{while } x \neq 5 \text{ do } x := x + 1 \]

- \((x \neq 5, \sigma, \text{false}) \implies P(W, \sigma, \sigma)\)
- \((x \neq 5, \sigma, \text{true}), (x=x+1, \sigma, \sigma'), (W, \sigma'', \sigma'), P(W, \sigma'', \sigma') \implies P(W, \sigma, \sigma')\)

Proving \((x \neq 5, \sigma, \text{false}) \implies P(W, \sigma, \sigma)\) is easy

- \((x \neq 5, \sigma, \text{false})\) is only valid if \(\sigma(x) = 5\)
- In that case, \(P(W, \sigma, \sigma) = \sigma(x) \leq 5 \& \sigma = \sigma[x=5]\)
- which is true, so the implication is clearly valid
Example

In order to complete the proof, we need to prove

\[ W ::= \text{while } x \neq 5 \text{ do } x := x + 1 \]

- \((x \neq 5, \sigma, \text{false}) \Rightarrow P(W, \sigma, \sigma)\)
- \((x \neq 5, \sigma, \text{true}), (x=x+1, \sigma, \sigma\'), (W, \sigma'', \sigma'), P(W, \sigma'', \sigma') \Rightarrow P(W, \sigma, \sigma')\)

And for the second implication, the proof is as follows

- First, we are told that \((x := x + 1, \sigma, \sigma''')\) which implies that

\[ \sigma''' = \sigma[x = \sigma(x) + 1] \]

- Then, we know \(P(W, \sigma'', \sigma')\) so that tells us that

\[ \sigma''(x) \leq 5 \land \sigma' = \sigma''[x = 5] \]

- Consequently \(\sigma''(x) = \sigma(x) + 1 \leq 5 \Rightarrow \sigma(x) \leq 5\)
- and \(\sigma' = \sigma'''[x = 5] = (\sigma[x = \sigma(x) + 1])[x = 5] = \sigma[x = 5]\)
- Which proves \(P(W, \sigma, \sigma') = \sigma(x) \leq 5 \land \sigma' = \sigma[x = 5]\)
What we have proven

By structural induction we have proven that:
- \( P(s) \) holds for all valid judgments \( s \in I_R \), where
\[
P(s) = \begin{cases} 
  s = (W, \sigma, \sigma') & \sigma(x) \leq 5 \land \sigma' = \sigma[x = 5] \\
  \text{else} & \text{true}
\end{cases}
\]

So what does this mean?
- It means the judgment \( \langle W, \sigma \rangle \rightarrow \sigma' \) can only be valid when \( \sigma(x) \leq 5 \)
- i.e. \( W \) will never terminate when \( \sigma(x) > 5 \)
- and finally, it means that if \( W \) does terminate, \( \sigma' = \sigma[x = 5] \)

Pretty neat!