COSC 4780 - Principles of Programming Languages - Spring 2022

Lecture 12: Proof Trees

Finley McIlwaine

University of Wyoming

fmcilwai@uwyo.edu

March 3, 2022

Review

Last time we finished formalizing the type system of our simple imperative language. We defined the typing relations for expressions, statements, and programs.

Today, we'll look back at all of the inference we developed and cover what each one is saying again. Then, we'll see how we can actually use those inference rules to formally prove various judgements.

Recall that *judgements* are statements of the form:

$$\Gamma \vdash_e e : T$$

Expression Typing Rules

Here's our grammar for expressions:

```
EOr.   Exp   ::= Exp1 "||" Exp   ;
EAnd.  Exp   ::= Exp1 "&&" Exp   ;
EEq.   Exp1  ::= Exp1 "==" Exp2  ;
ELeq.  Exp1  ::= Exp1 "<=" Exp2  ;
ENot.  Exp2  ::=      "!"  Exp4  ;
ESub.  Exp2  ::= Exp2 "-"  Exp3  ;
EAdd.  Exp2  ::= Exp2 "+"  Exp3  ;
EMul.  Exp3  ::= Exp3 "*"  Exp4  ;
ETru.  Exp4  ::= "true"          ;
EFls.  Exp4  ::= "false"         ;
EVar.  Exp4  ::= Ident           ;
EInt.  Exp4  ::= Integer         ;
```

Expression Typing Rules

Literal value & identifier expression typing rules:

$$\frac{n \in \mathbb{Z}}{\Gamma \vdash_e n : \mathit{int}}\ \textsc{EInt} \qquad \frac{b \in \mathbb{B}}{\Gamma \vdash_e b : \mathit{bool}}\ \textsc{EBool} \qquad \frac{\mathit{id} : T \in \Gamma}{\Gamma \vdash_e \mathit{id} : T}\ \textsc{EIdent}$$

Expression Typing Rules

Literal value & identifier expression typing rules:

$$\frac{n \in \mathbb{Z}}{\Gamma \vdash_e n : int}\ \textsc{EInt} \qquad \frac{b \in \mathbb{B}}{\Gamma \vdash_e b : bool}\ \textsc{EBool} \qquad \frac{id : T \in \Gamma}{\Gamma \vdash_e id : T}\ \textsc{EIdent}$$

Numeric operator typing rules:

$$\frac{\Gamma \vdash_e e_1 : int \qquad \Gamma \vdash_e e_2 : int}{\Gamma \vdash_e e_1 \circledcirc e_2 : int}\ \textsc{ENumOp}$$

where $\circledcirc \in \{*, +, -\}$

Expression Typing Rules

Literal value & identifier expression typing rules:

$$\frac{n \in \mathbb{Z}}{\Gamma \vdash_e n : \mathit{int}}\ \textsc{EInt} \qquad \frac{b \in \mathbb{B}}{\Gamma \vdash_e b : \mathit{bool}}\ \textsc{EBool} \qquad \frac{\mathit{id} : T \in \Gamma}{\Gamma \vdash_e \mathit{id} : T}\ \textsc{EIdent}$$

Numeric operator typing rules:

$$\frac{\Gamma \vdash_e e_1 : \mathit{int} \qquad \Gamma \vdash_e e_2 : \mathit{int}}{\Gamma \vdash_e e_1 \circledcirc e_2 : \mathit{int}}\ \textsc{ENumOp}$$

where $\circledcirc \in \{*, +, -\}$

Comparison operator typing rules:

$$\frac{\Gamma \vdash_e e_1 : \mathit{int} \qquad \Gamma \vdash_e e_2 : \mathit{int}}{\Gamma \vdash_e e_1 \circledcirc e_2 : \mathit{bool}}\ \textsc{ECompOp}$$

where $\circledcirc \in \{==, <=\}$

Expression Typing Rules

Boolean negation and "and" and "or" typing rules:

$$\frac{\Gamma \vdash_e e : \mathit{bool}}{\Gamma \vdash_e \,!e : \mathit{bool}}\ \textsc{ENeg} \qquad \frac{\Gamma \vdash_e e_1 : \mathit{bool} \qquad \Gamma \vdash_e e_2 : \mathit{bool}}{\Gamma \vdash_e e_1 \circledcirc e_2 : \mathit{bool}}\ \textsc{EBoolOp}$$

$$\text{where } \circledcirc \in \{\&\&, ||\}$$

That concludes the rules for expressions!

Statement Typing Rules

Here was our grammar for statements:

```
SInit.  Stm ::= Ident ":=" Exp ";"                  ;
SAss.   Stm ::= Ident "=" Exp ";"                   ;
SBlock. Stm ::= "{" [Stm] "}"                       ;
SIf.    Stm ::= "if" Exp "then" Stm "else" Stm      ;
SWhile. Stm ::= "while" Exp "do" Stm                ;
SPrint. Stm ::= "print" Exp ";"                     ;
```

Statement Typing Rules

Initialization and assignment typing rules:

$$\frac{\Gamma \vdash_e e : T}{\Gamma \vdash_s id := e; \Rightarrow \Gamma, id : T}\ \textsc{SInit} \qquad \frac{\Gamma \vdash_e e : T \qquad id : T \in \Gamma}{\Gamma \vdash_s id = e; \Rightarrow \Gamma}\ \textsc{SAss}$$

Statement Typing Rules

Initialization and assignment typing rules:

$$\frac{\Gamma \vdash_e e : T}{\Gamma \vdash_s id := e; \Rightarrow \Gamma, id : T}\ \textsc{SInit} \qquad \frac{\Gamma \vdash_e e : T \qquad id : T \in \Gamma}{\Gamma \vdash_s id = e; \Rightarrow \Gamma}\ \textsc{SAss}$$

Print and block statement typing rules:

$$\frac{\Gamma \vdash_e e : T}{\Gamma \vdash_s print\ \ e; \Rightarrow \Gamma}\ \textsc{SPrint} \qquad \frac{\Gamma \triangleright \varnothing \vdash_s s_0 \ldots s_n \Rightarrow \Gamma \triangleright \Gamma'}{\Gamma \vdash_s \{s_0 \ldots s_n\} \Rightarrow \Gamma}\ \textsc{SBlock}$$

Statement Typing Rules

If and while statement typing rules:

$$\frac{\Gamma \vdash_e e : bool \qquad \Gamma \triangleright \varnothing \vdash_s stm_1 \Rightarrow \Gamma \triangleright \Gamma' \qquad \Gamma \triangleright \varnothing \vdash_s stm_2 \Rightarrow \Gamma \triangleright \Gamma''}{\Gamma \vdash_s if \;\; e \;\; then \;\; stm_1 \;\; else \;\; stm_2 \Rightarrow \Gamma} \; \textsc{SIf}$$

$$\frac{\Gamma \vdash_e e : bool \qquad \Gamma \triangleright \varnothing \vdash_s stm \Rightarrow \Gamma \triangleright \Gamma'}{\Gamma \vdash_s while \;\; e \;\; do \;\; stm \Rightarrow \Gamma} \; \textsc{SWhile}$$

Statement Typing Rules

If and while statement typing rules:

$$\frac{\Gamma \vdash_e e : bool \qquad \Gamma \triangleright \varnothing \vdash_s stm_1 \Rightarrow \Gamma \triangleright \Gamma' \qquad \Gamma \triangleright \varnothing \vdash_s stm_2 \Rightarrow \Gamma \triangleright \Gamma''}{\Gamma \vdash_s if\ \ e\ \ then\ \ stm_1\ \ else\ \ stm_2 \Rightarrow \Gamma}\ \textsc{SIf}$$

$$\frac{\Gamma \vdash_e e : bool \qquad \Gamma \triangleright \varnothing \vdash_s stm \Rightarrow \Gamma \triangleright \Gamma'}{\Gamma \vdash_s while\ \ e\ \ do\ \ stm \Rightarrow \Gamma}\ \textsc{SWhile}$$

And the rules for type checking lists of statements:

$$\frac{\Gamma \vdash_s s_0 \Rightarrow \Gamma' \qquad \Gamma' \vdash_s s_1 \ldots s_n \Rightarrow \Gamma''}{\Gamma \vdash_s s_0 \ldots s_n \Rightarrow \Gamma''}\ \textsc{SNEList} \qquad \frac{}{\Gamma \vdash_s \oslash \Rightarrow \Gamma}\ \textsc{SEList}$$

Program Typing Rule

Since programs in our language are only lists of statements:

$$\frac{\varnothing \vdash_s s_0 \ldots s_n \Rightarrow \Gamma}{\varnothing \vdash_p s_0 \ldots s_n} \textsc{Prog}$$

And we're done! That is the entire type system for the language.

Proof Trees

We would like to be able to prove judgements. For example, how could we know if the following judgement is true or false?

$$\varnothing \vdash_p x := 5;\ \ \textit{print}\ 5 + x;$$

Proof Trees

We would like to be able to prove judgements. For example, how could we know if the following judgement is true or false?

$$\varnothing \vdash_p x := 5;\ \textit{print}\ 5 + x;$$

We could run the type checker on it, but we have a mathematical type system so we want a mathematical proof that uses the system we've built!

Proof Trees

We use *proof trees*, which are trees of repeated applications of inference rules. By construction, the root and internal nodes of a proof tree are true judgements. The leaves of a proof tree are axioms. Proof trees look like stacked inference rules, with the root occurring at the bottom and the leaves occurring up top:

$$\cfrac{\cfrac{\cfrac{}{2 \in \mathbb{Z}}\;\text{Obv}}{\varnothing \vdash_e 2 : \mathit{int}}\;\text{EInt} \qquad \cfrac{\cfrac{}{3 \in \mathbb{Z}}\;\text{Obv}}{\varnothing \vdash_e 2 : \mathit{int}}\;\text{EInt}}{\varnothing \vdash_e 2 <= 3 : \mathit{bool}}\;\text{ECompOp}$$

Every node of a proof tree is labelled with the name of the inference rule being applied.

Proof Trees

Proof trees are effectively traces of the steps taken by a type checker! If a program p is not well-typed, there should be no proof tree concluding with the judgement $\Gamma \vdash_p p$. Equivelently, it should be impossible to construct such a proof tree!

Additionally, if a program is not well-typed, our type checker should reject the program.

Proof Trees

Let's build a proof tree for the judgement $\varnothing \vdash_p x := 5;\ \textit{print}\ 3 + x;$.

$$\frac{}{\varnothing \vdash_p x := 5;\ \textit{print}\ 3 + x;}\ ???$$

What rule applies?

Proof Trees

Let's build a proof tree for the judgement $\varnothing \vdash_p x := 5;\ \textit{print}\ 3 + x;$.

$$\dfrac{\dfrac{}{\varnothing \vdash_s x := 5;\ \textit{print}\ 5 + x; \Rightarrow x : \textit{int}}\ ???}{\varnothing \vdash_p x := 5;\ \textit{print}\ 3 + x;}\ \text{Prog}$$

Prog! Now what rule applies?

Proof Trees

Let's build a proof tree for the judgement $\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;$.

$$\dfrac{\dfrac{}{\varnothing \vdash_s x := 5; \Rightarrow x : \mathit{int}}\ ??? \qquad \dfrac{}{x : \mathit{int} \vdash_s \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}\ ???}{\dfrac{\varnothing \vdash_s x := 5;\ \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}{\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;}\ \text{Prog}}\ \text{SNEList}$$

SNEList! Let's prove the left premise first! What rule applies?

Proof Trees

Let's build a proof tree for the judgement $\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;$.

$$\dfrac{\dfrac{\dfrac{}{\varnothing \vdash_e 5 : \mathit{int}}\ ???}{\varnothing \vdash_s x := 5; \Rightarrow x : \mathit{int}}\ \text{SInit} \qquad \dfrac{}{x : \mathit{int} \vdash_s \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}\ ???}{\dfrac{\varnothing \vdash_s x := 5;\ \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}{\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;}\ \text{Prog}}\ \text{SNEList}$$

SInit! Now what rule applies?

Proof Trees

Let's build a proof tree for the judgement $\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{5 \in \mathbb{Z}}\ \text{Obv}}{\varnothing \vdash_e 5 : \mathit{int}}\ \text{EInt}}{\varnothing \vdash_s x := 5; \Rightarrow x : \mathit{int}}\ \text{SInit} \qquad \dfrac{}{x : \mathit{int} \vdash_s \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}\ ???}{\varnothing \vdash_s x := 5;\ \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}\ \text{SNEList}}{\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;}\ \text{Prog}$$

EInt! We've proven the left premise! What rule applies to the right premise? This one is a little tricky.

Proof Trees

Let's build a proof tree for the judgement $\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{5 \in \mathbb{Z}}\,\text{Obv}}{\varnothing \vdash_e 5 : \mathit{int}}\,\text{EInt}}{\varnothing \vdash_s x := 5; \Rightarrow x : \mathit{int}}\,\text{SInit} \qquad \dfrac{\dfrac{}{x : \mathit{int} \vdash_s \mathit{print}\ 3 + x; \Rightarrow x : \mathit{int}}\,\text{???} \qquad \dfrac{}{x : \mathit{int} \vdash_s \varnothing \Rightarrow x : \mathit{int}}\,\text{SEList}}{x : \mathit{int} \vdash_s \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}\,\text{SNEList}}{\dfrac{\varnothing \vdash_s x := 5;\ \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}{\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;}\,\text{Prog}}\,\text{SNEList}$$

SNEList! The right premise of this goal is now just the empty statement list case. What rule applies to the left premise?

Proof Trees

Let's build a proof tree for the judgement $\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{5 \in \mathbb{Z}}\ \text{Obv}}{\varnothing \vdash_e 5 : \mathit{int}}\ \text{EInt}}{\varnothing \vdash_s x := 5; \Rightarrow x : \mathit{int}}\ \text{SInit} \qquad \dfrac{\dfrac{\dfrac{}{x : \mathit{int} \vdash_e 3 + x : \mathit{int}}\ ???}{x : \mathit{int} \vdash_s \mathit{print}\ 3 + x; \Rightarrow x : \mathit{int}}\ \text{SPrint} \qquad \dfrac{}{x : \mathit{int} \vdash_s \varnothing \Rightarrow x : \mathit{int}}\ \text{SEList}}{x : \mathit{int} \vdash_s \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}\ \text{SNEList}}{\dfrac{\varnothing \vdash_s x := 5;\ \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}{\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;}\ \text{Prog}}\ \text{SNEList}$$

SPrint! Now what rule applies now?

Proof Trees

Let's build a proof tree for the judgement $\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;$.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{}{5 \in \mathbb{Z}}\ \text{Obv}
      }{\varnothing \vdash_e 5 : \mathit{int}}\ \text{EInt}
    }{\varnothing \vdash_s x := 5; \Rightarrow x : \mathit{int}}\ \text{SInit}
    \quad
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{}{x : \mathit{int} \vdash_e 3 : \mathit{int}}\ ???
          \quad
          \dfrac{}{x : \mathit{int} \vdash_e x : \mathit{int}}\ ???
        }{x : \mathit{int} \vdash_e 3 + x : \mathit{int}}\ \text{EAdd}
      }{x : \mathit{int} \vdash_s \mathit{print}\ 3 + x; \Rightarrow x : \mathit{int}}\ \text{SPrint}
      \quad
      \dfrac{}{x : \mathit{int} \vdash_s \varnothing \Rightarrow x : \mathit{int}}\ \text{SEList}
    }{x : \mathit{int} \vdash_s \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}\ \text{SNEList}
  }{\varnothing \vdash_s x := 5;\ \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}\ \text{SNEList}
}{\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;}\ \text{Prog}
$$

EAdd! Let's prove the left premise first! What rule applies?

Proof Trees

Let's build a proof tree for the judgement $\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;$.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{5 \in \mathbb{Z}}\ \text{Obv}
    }{\varnothing \vdash_e 5 : \mathit{int}}\ \text{EInt}
  }{\varnothing \vdash_s x := 5; \Rightarrow x : \mathit{int}}\ \text{SInit}
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{}{3 \in \mathbb{Z}}\ \text{Obv}
        }{x : \mathit{int} \vdash_e 3 : \mathit{int}}\ \text{EInt}
        \quad
        \dfrac{
          \dfrac{}{x : \mathit{int} \in \{x : \mathit{int}\}}\ \text{Obv}
        }{x : \mathit{int} \vdash_e x : \mathit{int}}\ \text{EIdent}
      }{x : \mathit{int} \vdash_e 3 + x : \mathit{int}}\ \text{EAdd}
    }{x : \mathit{int} \vdash_s \mathit{print}\ 3 + x; \Rightarrow x : \mathit{int}}\ \text{SPrint}
    \qquad
    \dfrac{}{x : \mathit{int} \vdash_s \varnothing \Rightarrow x : \mathit{int}}\ \text{SEList}
  }{x : \mathit{int} \vdash_s \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}\ \text{SNEList}
}{
  \dfrac{\varnothing \vdash_s x := 5;\ \mathit{print}\ 5 + x; \Rightarrow x : \mathit{int}}{\varnothing \vdash_p x := 5;\ \mathit{print}\ 3 + x;}\ \text{Prog}
}\ \text{SNEList}
$$

EIdent! We've officially proven all judgements up to our axioms!

Midterm Discussion

For the rest of this class, we are going to discuss the midterm, and maybe mess around with our type checker from last time a little bit to examine the parallels to proof trees.

The End