CS-320

Computer Language Processing

Exercise Session 4

November 1, 2017

Overview

Today you will get some more practice in understanding and designing type systems:

- Exploring a typing derivation in Amyrli
- Amy's pattern matching rule
- A type system for physical units

Recap: Type-checking a simple program

Consider the Amy-like language of arithmetic, logical connectives and if expressions from the lecture:

$$t := true \mid false \mid c_I \mid f(t_1, \ldots, t_n) \mid \textbf{if } (t)\ t_1 \textbf{ else } t_2$$

where c_I denotes integer literals.

We also saw some of its typing rules, for instance:

$$\textsc{If-Then-Else}\ \frac{\Gamma \vdash b : Bool \qquad \Gamma \vdash t_1 : \tau \qquad \Gamma \vdash t_2 : \tau}{\Gamma \vdash (\textbf{if } (b)\ t_1 \textbf{ else } t_2) : \tau}$$

$$\vdots$$

Finding a typing-derivation for a simple program

Exercise 1

▷ Given the type system we saw for this language, type-check and show the typing derivations for the following program:

$p_{fun} = (e, fun(2))$
where $e(fun) = (n,\ Int,\ \textbf{if}\ (n \leq 1)\ 1\ \textbf{else}\ n * fun(n),\ Int)$

Recap: When do we say a program type-checks?

Given initial program (e, t) define

$$\Gamma_0 = \{(f, \tau_1 \times \cdots \times \tau_n \to \tau_0) \mid (f, \_, (\tau_1, \ldots, \tau_n), t_f, \tau_0) \in e\}$$

We say program type checks iff:
(1) the top-level expression type checks:

$$\Gamma_0 \vdash t : \tau$$

and
(2) each function body type checks:

$$\Gamma_0 \oplus \{(x_1, \tau_1), \ldots, (x_n, \tau_n)\} \vdash t_f : \tau_0$$

for each $(f, (x_1, \ldots, x_n), (\tau_1, \ldots, \tau_n), t_f, \tau_0) \in e$.

Finding a typing-derivation for a simple program

Exercise 1 (solution)

$\Rightarrow$ We have to check whether
a) $\Gamma_0 \vdash \mathit{fun}(2) : T$ for some type T, and
b) $\Gamma_0' \vdash \textbf{if } (n \leq 1)\ 1 \textbf{ else } n * \mathit{fun}(n) : \mathit{Int}$
where $\Gamma_0 = \{\ldots(\mathit{builtins}), (\mathit{fun}, \mathit{Int} \Rightarrow \mathit{Int})\}$ and
$\Gamma_0' = \Gamma_0 \oplus \{(n, \mathit{Int})\}$.

Typing derivation for $\Gamma_0 \vdash \mathit{fun}(2) : T$:

$$\dfrac{\dfrac{(\mathit{fun}, \mathit{Int} \Rightarrow \mathit{Int}) \in \Gamma_0}{\Gamma_0 \vdash \mathit{fun} : \mathit{Int} \Rightarrow \mathit{Int}} \qquad \dfrac{}{\Gamma_0 \vdash 2 : \mathit{Int}}}{\Gamma_0 \vdash \mathit{fun}(2) : \mathit{Int}}$$

Finding a typing-derivation for a simple program

Exercise 1 (solution)

Typing derivation for $\Gamma_0' \vdash \textbf{if } (n \leq 1) \; 1 \textbf{ else } n * \mathit{fun}(n) : \mathit{Int}$ where $\Gamma_0' = \Gamma_0 \oplus \{(n, \mathit{Int})\}$:

$$\dfrac{\dfrac{\overset{?}{\vdots}}{\Gamma_0' \vdash n \leq 1 : \mathit{Bool}} \qquad \dfrac{}{\Gamma_0' \vdash 1 : \mathit{Int}} \qquad \dfrac{\overset{?}{\vdots}}{\Gamma_0' \vdash n * \mathit{fun}(n) : \mathit{Int}}}{\Gamma_0' \vdash \textbf{if } (n \leq 1) \; 1 \textbf{ else } n * \mathit{fun}(n) : \mathit{Int}}$$

Finding a typing-derivation for a simple program

Exercise 1 (solution)

Typing derivation for $\Gamma'_0 \vdash n \leq 1 : Bool$.

$$
\dfrac{
\dfrac{(\leq, (Int \times Int) \Rightarrow Bool) \in \Gamma'_0}{\Gamma'_0 \vdash \leq : (Int \times Int) \Rightarrow Bool}
\qquad
\dfrac{(n, Int) \in \Gamma'_0}{\Gamma'_0 \vdash n : Int}
\qquad
\dfrac{}{\Gamma'_0 \vdash 1 : Int}
}{\Gamma'_0 \vdash n \leq 1 : Bool}
$$

Finding a typing-derivation for a simple program

Exercise 1 (solution)

Typing derivation for $\Gamma_0' \vdash n * fun(n) : Int$.

$$\dfrac{\dfrac{(*, (Int \times Int) \Rightarrow Int) \in \Gamma_0'}{\Gamma_0' \vdash * : (Int \times Int) \Rightarrow Int} \qquad \dfrac{(n, Int) \in \Gamma_0'}{\Gamma_0' \vdash n : Int} \qquad \dfrac{\dfrac{(fun, Int \Rightarrow Int) \in \Gamma_0'}{\Gamma_0' \vdash fun : Int \Rightarrow Int} \qquad \dfrac{(n, Int) \in \Gamma_0'}{\Gamma_0' \vdash n : Int}}{\Gamma_0' \vdash fun(n) : Int}}{\Gamma_0' \vdash n * fun(n) : Int}$$

Finding a typing-derivation for a simple program

Exercise 1

▷ We have shown that the program type-checks, but did you notice any other problem with it?

Typing rules for pattern matching in Amy

We have seen a typing rule for if expressions, but how can we type more advanced control constructs like pattern matches?
Let's see a corresponding rule for the Amy language:

Pattern Matching

$$\frac{\Gamma \vdash e : T_s \qquad \forall i \in [1, n]. \quad \Gamma \vdash p_i : T_s \triangleright \Gamma_{p_i} \quad \Gamma \oplus \Gamma_{p_i} \vdash e_i : T_c}{\Gamma \vdash e \ \texttt{match \{ case}\ p_1 \ \texttt{=>}\ e_1 \ \ldots \ \texttt{case}\ p_n \ \texttt{=>}\ e_n \ \texttt{\}} : T_c}$$

Note that we use auxiliary *extraction* judgments of the form

$$\Gamma \vdash p : T \triangleright \Gamma_p$$

to *check* that pattern p matches a type T, while also *extracting* its bindings Γ_p.

Typing rules for pattern matching in Amy

We have seen a typing rule for if expressions, but how can we type more advanced control constructs like pattern matches?
Let's see a corresponding rule for the Amy language:

$$\textsc{Pattern Matching}\quad \frac{\color{red}{\Gamma \vdash e : T_s} \qquad \forall i \in [1, n].\quad \Gamma \vdash p_i : T_s \triangleright \Gamma_{p_i} \qquad \Gamma \oplus \Gamma_{p_i} \vdash e_i : T_c}{\Gamma \vdash e\ \texttt{match \{ case}\ p_1\ \texttt{=>}\ e_1\ \dots\ \texttt{case}\ p_n\ \texttt{=>}\ e_n\ \texttt{\}} : T_c}$$

Note that we use auxiliary *extraction* judgments of the form

$$\Gamma \vdash p : T \triangleright \Gamma_p$$

to *check* that pattern p matches a type T, while also *extracting* its bindings Γ_p.

Typing rules for pattern matching in Amy

We have seen a typing rule for if expressions, but how can we type more advanced control constructs like pattern matches?
Let's see a corresponding rule for the Amy language:

PATTERN MATCHING

$$\frac{\Gamma \vdash e : T_s \qquad \forall i \in [1, n]. \quad \Gamma \vdash p_i : T_s \triangleright \Gamma_{p_i} \qquad \Gamma \oplus \Gamma_{p_i} \vdash e_i : T_c}{\Gamma \vdash e \texttt{ match \{ case } p_1 \texttt{ => } e_1 \ \ldots \ \texttt{case } p_n \texttt{ => } e_n \texttt{ \}} : T_c}$$

Note that we use auxiliary *extraction* judgments of the form

$$\Gamma \vdash p : T \triangleright \Gamma_p$$

to *check* that pattern p matches a type T, while also *extracting* its bindings Γ_p.

Typing rules for pattern matching in Amy

We have seen a typing rule for if expressions, but how can we type more advanced control constructs like pattern matches?
Let's see a corresponding rule for the Amy language:

$$\textsc{Pattern Matching}\quad \frac{\Gamma \vdash e : T_s \qquad \forall i \in [1, n]. \quad \Gamma \vdash p_i : T_s \triangleright \Gamma_{p_i} \quad \Gamma \oplus \Gamma_{p_i} \vdash e_i : T_c}{\Gamma \vdash e \;\texttt{match \{ case}\; p_1 \;\texttt{=>}\; e_1 \;\ldots\; \texttt{case}\; p_n \;\texttt{=>}\; e_n \;\texttt{\}} : T_c}$$

Note that we use auxiliary *extraction* judgments of the form

$$\Gamma \vdash p : T \triangleright \Gamma_p$$

to *check* that pattern p matches a type T, while also *extracting* its bindings Γ_p.

Typing rules for pattern matching in Amy (2)

We define the following extraction rules for patterns:

Wildcard Pattern

$$\frac{}{\Gamma \vdash \_ : T \triangleright \varnothing}$$

Identifier Pattern

$$\frac{}{\Gamma \vdash v : T \triangleright \{(v, T)\}}$$

Case Class Pattern

$$\frac{\Gamma \vdash p_1 : T_1 \triangleright \Gamma_{p_1} \qquad \cdots \qquad \Gamma \vdash p_n : T_n \triangleright \Gamma_{p_n} \qquad \Gamma \vdash C : (T_1, \ldots, T_n) \Rightarrow T}{\Gamma \vdash C(p_1, \ldots, p_n) : T \triangleright \Gamma_{p_1} \oplus \cdots \oplus \Gamma_{p_n}}$$

Type-checking pattern matching expressions

Exercise 2

- Find a typing derivation for the body of function *len* in the following program:

```
abstract class List
case class Nil() extends List
case class Cons(x: Int, xs: List) extends List

def len(xs: List): Int = xs match {
  case Nil() => 0
  case Cons(_, rest) => len(rest) + 1
}
```

A type-system for physical units

Exercise 3

Consider the following language of integral additions, multiplications, divisions:

$$t := c_R \mid \text{m} \mid \text{s} \mid t + t \mid t \cdot t \mid t \,/\, t \mid \text{sqrt}(t)$$

$$T := \mathbf{1} \mid \text{meter} \mid \text{second} \mid T * T \mid T^{-1}$$

where c_R denotes a real literal and m, s are used to introduce meters and seconds as units.

For instance:

$$3 : \mathbf{1} \qquad 4 \cdot \text{m} : \text{meter} \qquad 3 \cdot \text{m}/\text{s} : \text{meter} * \text{second}^{-1}$$

A type-system for physical units

Exercise 3

Note that $(\text{meter} * \text{second} * \text{meter}^{-1})$ and (second) are not *syntactically* equivalent!

$\Rightarrow$ We will implicitly *normalize* our types and use a shorthand:

$$\text{Dim } m \ n \equiv \mathbf{1} * \text{meter}^m * \text{second}^n$$

For instance:

$$\mathbf{1} \equiv \text{Dim } 0 \ 0 \qquad \text{meter} \equiv \text{Dim } 1 \ 0 \qquad \text{second}^{-1} * \text{meter} \equiv \text{Dim } 1 \ -1$$

$$\text{meter} * (\text{second} * \text{meter})^{-1} * \text{second} \equiv \text{Dim } 0 \ 0$$

A type-system for physical units

Exercise 3a

▷ Design typing rules that track the units of expressions and only permit adding expressions of the same unit. Furthermore, make sure that sqrt will only accept square meters.

▷ Write a function
$\text{dist} : (\text{meter} * \text{second}^{-1} \times \text{second}) \Rightarrow \text{meter}$
and show its typing derivation.

A type-system for physical units

Exercise 3a (solution)

$$\textsc{T-Lit}\quad \frac{}{\vdash c_R : \mathbf{1}} \qquad \textsc{T-Met}\quad \frac{}{\vdash \mathsf{m} : \mathsf{meter}} \qquad \textsc{T-Sec}\quad \frac{}{\vdash \mathsf{s} : \mathsf{second}}$$

$$\textsc{T-Add}\ \frac{\vdash t_1 : \mathsf{Dim}\ m\ n \qquad \vdash t_2 : \mathsf{Dim}\ m\ n}{\vdash t_1 + t_2 : \mathsf{Dim}\ m\ n}$$

$$\textsc{T-Mul}\ \frac{\vdash t_1 : \mathsf{Dim}\ m_1\ n_1 \qquad \vdash t_2 : \mathsf{Dim}\ m_2\ n_2}{\vdash t_1 \cdot t_2 : \mathsf{Dim}\ (m_1 + m_2)\ (n_1 + n_2)}$$

$$\textsc{T-Div}\ \frac{\vdash t_1 : \mathsf{Dim}\ m_1\ n_1 \qquad \vdash t_2 : \mathsf{Dim}\ m_2\ n_2}{\vdash t_1 / t_2 : \mathsf{Dim}\ (m_1 - m_2)\ (n_1 - n_2)}$$

$$\textsc{T-WeirdSqrt}\ \frac{\vdash t : \mathsf{Dim}\ 2m\ 0}{\vdash \mathsf{sqrt}(t) : \mathsf{Dim}\ m\ 0}$$

A type-system for physical units

Exercise 3b

▷ Using your typing rules, find a typing derivation for the following (top-level) expression:

$$\mathsf{sqrt}(\mathsf{m} \cdot 4 \cdot \mathsf{m}) + 1/\mathsf{s} \cdot 10 \cdot \mathsf{m} \cdot \mathsf{s}$$

A type-system for physical units

Exercise 3b

▷ Using your typing rules, find a typing derivation for the following (top-level) expression:

$$\mathsf{sqrt}((\mathsf{m} \cdot 4) \cdot \mathsf{m}) + (((1/\mathsf{s}) \cdot 10) \cdot \mathsf{m}) \cdot \mathsf{s}$$