Functional Programming: Exercise Session 2

- Sheet 2: Main focus on logic, some Haskell programming
- Next week: Lecture and exercises will shift to Haskell
- This exercise session:
  - Some comments on sheet 1: Evaluation, elimination rules
  - Semantics of logics
  - Soundness of rules
  - Practice for sheet 2: Rules for $\forall$ and $\exists$

Evaluation

- Evaluation order will be discussed in more detail later, but useful to already have a general idea of it.
- Haskell has a lazy evaluation strategy. It first evaluates the leftmost and outermost reducible expression.
- Outermost = (if possible first try to) evaluate the function, then evaluate the arguments. When do you need to evaluate the arguments first?
- Lazy = only arguments which are required are necessary

$$\begin{aligned} &\mathit{fst}\ ((0+1)+(1+1), 1+2+3+4+5+6) \\ &\hookrightarrow (0+1)+(1+1) \\ &\hookrightarrow 1+(1+1) \hookrightarrow 1+2 \hookrightarrow 3 \end{aligned}$$

- However, lazy/eager not really important for the Fibonacci example

Evaluation

```
fibLouis 0 = 0                                          -- fibLouis.1
fibLouis 1 = 1                                          -- fibLouis.2
fibLouis n = fibLouis (n-1) + fibLouis (n-2)            -- fibLouis.3
```

```
fibLouis 4 =
(fibLouis (4-1) + fibLouis (4-2)) =                                    - fibLouis.3
(fibLouis 3 + fibLouis (4-2)) =                                        - arith
((fibLouis (3-1) + fibLouis (3-2)) + fibLouis (4-2)) =                 - fibLouis.3
((fibLouis 2 + fibLouis (3-2)) + fibLouis (4-2)) =                     - arith
(((fibLouis (2-1) + fibLouis (2-2)) + fibLouis (3-2)) + fibLouis (4-2)) = - fibLouis.3
(((fibLouis 1 + fibLouis (2-2)) + fibLouis (3-2)) + fibLouis (4-2)) =  - arith
(((1 + fibLouis (2-2)) + fibLouis (3-2)) + fibLouis (4-2)) =           - fibLouis.2
(((1 + fibLouis 0) + fibLouis (3-2)) + fibLouis (4-2)) =               - arith
(((1 + 0) + fibLouis (3-2)) + fibLouis (4-2)) =                        - fibLouis.1
((1 + fibLouis (3-2)) + fibLouis (4-2)) =                              - arith
((1 + fibLouis 1) + fibLouis (4-2)) =                                  - arith
((1 + 1) + fibLouis (4-2)) =                                           - fibLouis.1
(2 + fibLouis (4-2)) =                                                 - arith
(2 + fibLouis 2) =                                                     - arith
(2 + (fibLouis (2-1) + fibLouis (2-2))) =                              - fibLouis.3
(2 + (fibLouis 1 + fibLouis (2-2))) =                                  - arith
(2 + (1 + fibLouis (2-2))) =                                           - fibLouis.2
(2 + (1 + fibLouis 0)) =                                               - arith
(2 + (1 + 0)) =                                                        - fibLouis.1
(2 + 1) =                                                              - arith
3                                                                      - arith
```

How to find ND proofs?

- Strategy rather than an exact algorithm (problem can be undecidable, depending on the logic)
- Backwards: Apply introduction rules until the required sequents can be derived by elimination rules.
- Forward: Find out how elimination rules can be applied to assumptions to derive required sequents.
- Backwards: Write down the previously found derivations.
- The forward "planning step" is required since you might have to guess *which rule* to apply and *how to instantiate the premises*:

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-EL} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-ER}$$

- Note that $\rightarrow -E$ might require switching back to applying introduction rules for the second premise.

Blackboard: Assignment 4 (b) (i)

Why should proofs follow such a rigid structure?

- More comfortable rules can be derived!
- Machine checkable proofs and proof search:
- Xavier Leroy: Formal verification of a realistic compiler
  `http://pauillac.inria.fr/~xleroy/bibrefs/Leroy-Compcert-CACM.html`
- Formal, machine-checked verification of the seL4 microkernel:
  - `http://www.drdobbs.com/embedded/222400553?pgno=2`
  - `http://www.ok-labs.com/whitepapers/sample/sel4-formal-verification-of-an-os-kernel`
- Other formal methods:
  - `http://cacm.acm.org/magazines/2010/2/69367-type-theory-comes-of-age/fulltext`
  - `http://cacm.acm.org/magazines/2010/2/69362-software-model-checking-takes-off/fulltext`

Semantics of Logics

- We want to assign meaning (semantics) to logical formulae
- Typically, one does this by defining a *satisfaction relation*

$$S \models F$$

  which describes when a *situation* S satisfies a formula F
- Details vary from logic to logic: what is the satisfaction relation, what constitutes a formula, and what constitutes a situation

Semantics of Propositional Logic

- In propositional logic, situations are the so-called *valuations*, functions assigning a value $p_M \in \{\bot, \top\}$ to each variable p. The satisfaction relation is then defined as follows:
  - $M \models p$, if p_M
  - $M \models A \wedge B$, if $M \models A$ and $M \models B$
  - $M \models A \vee B$, if $M \models A$ or $M \models B$
  - $M \models A \to B$, if whenever $M \models A$ then $M \models B$
- A situation M which satisfies the formula F is called a *model* for F.
- Exercise: find one valuation which is, and one which isn't a model for the formula $p \wedge q \to r$

Semantics of Predicate Logic

- A **structure** is a pair $\mathcal{S} = \langle U_{\mathcal{S}}, I_{\mathcal{S}} \rangle$ where $U_{\mathcal{S}}$ is an nonempty set, the **universe**, and $I_{\mathcal{S}}$ is a mapping from predicate and function symbols of the given logic to predicates and functions over $U_{\mathcal{S}}$
- An **interpretation** is a pair $\mathcal{I} = \langle \mathcal{S}, v \rangle$, where $\mathcal{S} = \langle U_{\mathcal{S}}, I_{\mathcal{S}} \rangle$ is a structure and $v : \mathcal{V} \to U_{\mathcal{S}}$ a valuation.
- $\mathcal{I} \models \phi$ is defined as the smallest relation between interpretations and formulas such that

$$\langle \mathcal{S}, v \rangle \models p(t_1, \ldots, t_n) \quad \text{if} \quad \big(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n)\big) \in p^{\mathcal{S}}, \text{ where } \mathcal{I} = \langle \mathcal{S}, v \rangle$$

$$\vdots$$

$$\langle \mathcal{S}, v \rangle \models \forall x.\, A \quad \text{if} \quad \langle \mathcal{S}, v[x \mapsto a] \rangle \models A, \text{ for all } a \in U_{\mathcal{S}}$$

$$\langle \mathcal{S}, v \rangle \models \exists x.\, A \quad \text{if} \quad \langle \mathcal{S}, v[x \mapsto a] \rangle \models A, \text{ for some } a \in U_{\mathcal{S}}$$

Semantics and Syntax

- Semantic entailment: $\phi_1, \ldots, \phi_k \models \phi$ if for all situations $\mathcal{A}$, if $\mathcal{A} \models \phi_1, \ldots, \mathcal{A} \models \phi_k$ then $\mathcal{A} \models \phi$
- Syntactic entailment is what we have seen so far: $\Gamma \vdash \phi$ if we can "derive" ϕ from Γ
- We'd like the two to be in sync
- Soundness of logic: If $\phi_1, \ldots, \phi_k \vdash \phi$ derivable, then $\phi_1, \ldots, \phi_k \models \phi$
- Completeness of logic: If $\phi_1, \ldots, \phi_k \models \phi$, then $\phi_1, \ldots, \phi_k \vdash \phi$ derivable

Soundness of Rules

- A rule

$$\frac{\Gamma_1 \vdash \phi_1 \quad \ldots \quad \Gamma_k \vdash \phi_k}{\Gamma \vdash \phi}$$

  is sound if for all Γ_i, ϕ_i, Γ, and ϕ

$$\Gamma_1 \models \phi_1, \ldots, \Gamma_k \models \phi_k$$

  implies

$$\Gamma \models \phi$$

- Example: We show that

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi} \vee\text{-IL}$$

  is sound as follows:
  Let Γ, ϕ, and ψ be arbitrary and assume that $\Gamma \models \phi$.
  Then show $\Gamma \models \phi \vee \psi$.

Soundness of $\vee$-IL

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi} \vee\text{-IL}$$

1. Let ϕ, ψ, and $\Gamma = G_1, \ldots, G_k$ be arbitrary and assume that $\Gamma \models \phi$. *(unfold definition of semantic entailment)*
2. Then for all structures $\mathcal{A}$, if $\mathcal{A} \models G_i$ for $1 \leq i \leq k$ then $\mathcal{A} \models \phi$. *($\mathcal{A} \models \phi \vee \psi$ if $\mathcal{A} \models \phi$ or $\mathcal{A} \models \psi$ by definition of $\models$)*
3. Then for all structures $\mathcal{A}$, if $\mathcal{A} \models G_i$ for $1 \leq i \leq k$ then $\mathcal{A} \models \phi \vee \psi$. *(fold definition of semantic entailment)*
4. Then $\Gamma \models \phi \vee \psi$.

Notes: Predicate Logic on Sheet 2

- You have to pay attention to the scope of variables and the side conditions.
- Syntax: $\forall$ and $\exists$ extend as far to the right as possible.
  $A \wedge \forall x.\ P(x) \to Q(x) \to B(x) = A \wedge (\forall x.\ P(x) \to (Q(x) \to B(x)))$
- $\forall x_1\ x_2\ \ldots\ x_n.\ P$ shortcut for $\forall x_1.\, \forall x_2.\, \ldots \forall x_n.\, P$

Renaming of Bound Variables in Formulas

- Formulas modulo renaming of bound variables:

  $\forall x.\, P(x) \equiv \forall y.\, P(y)$
- Renaming has to be capture avoiding:

  $\forall x.\, Q(x, z) \neq \forall z.\, Q(z, z)$

  The second argument of Q was free before, then bound by $\forall$.

Side Conditions of $\forall$-I and $\exists$-E

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\exists x.\, P(x) \vdash \exists x.\, P(x)}\,\text{AX} \quad \dfrac{}{\exists x.\, P(x), P(x) \vdash P(x)}\,\text{AX}}{\exists x.\, P(x) \vdash P(x)}\,\exists\text{-E}^{**}}{\exists x.\, P(x) \vdash \forall x.P(x)}\,\forall\text{-I}^{*}}{\vdash (\exists x.\, P(x)) \to \forall x.P(x)}\,\to\text{-I}$$

Where is the error?

(*) x not free in any assumptions in Γ, (**) x not free in B or Γ.

Side Conditions of $\forall$-I and $\exists$-E

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\exists x.\, P(x) \vdash \exists x.\, P(x)}\,\text{AX} \quad \dfrac{}{\exists x.\, P(x), P(x) \vdash P(x)}\,\text{AX}}{\exists x.\, P(x) \vdash P(x)}\,\exists\text{-E}^{**}}{\exists x.\, P(x) \vdash \forall x.P(x)}\,\forall\text{-I}^{*}}{\vdash (\exists x.\, P(x)) \to \forall x.P(x)}\,\to\text{-I}$$

Where is the error?

(*) x not free in any assumptions in Γ, (**) x not free in B or Γ.

(**) violated, renaming required.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\exists y.\, P(y) \vdash \exists y.\, P(y)}\,\text{AX} \quad \exists y.\, P(y), P(y) \vdash P(x)}{\exists y.\, P(y) \vdash P(x)}\,\exists\text{-E}^{**}}{\exists x.\, P(x) \vdash P(x)}\,\text{RENAME}}{\exists x.\, P(x) \vdash \forall x.P(x)}\,\forall\text{-I}^{*}}{\vdash (\exists x.\, P(x)) \to \forall x.P(x)}\,\to\text{-I}$$

Note that formula is *unproven*, as it is actually *false*!

Exercises I

1. $\forall x. \bot \to P(x)$
2. $\forall x. P(x), P(t) \to Q \vdash Q$

Exercises II

3. $(\forall x\, y.\, P(x, y)) \to (\forall b\, a.\, P(a, b))$
4. $\exists x.\, P(x) \vdash \neg(\forall x.\, \neg P(x))$
5. $\forall x\, y\, z.\, P(x, y) \wedge P(y, z) \to P(x, z), P(a, b), P(b, c) \vdash P(a, c)$
6. $\neg(\forall x.\, \neg P(x)) \vdash \exists x.\, P(x)$ using TND