Formal Methods and Functional Programming – Summary

Author: Yannis Huber

1 Functional Programming

1.1 Natural Deduction

Natural deduction (ND) provides a way to reason formally about a system. A deductive proof system over a language is based on a set of rules which can be used to construct derivations under assumptions. A **rule** of the form

$$A_1, ..., A_n \vdash A,$$

reads as "A follows from $A_1, ..., A_n$". A **derivation** is a tree of rules and a **proof** is a derivation whose root has no assumptions. A deductive system must be **sound** (everything that is provable is in fact true) and **complete** (everything that is true has a proof).

1.1.1 Derivation Rules for Propositional Logic

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

$$\frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \to\text{-E} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\text{-IL} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\text{-IR}$$

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee\text{-E} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\text{-E} \qquad \frac{\Gamma \vdash \neg A \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \neg\text{-E} \qquad \frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}\ \text{RAA}$$

1.1.2 First-Order Logic

We can rename **bound** variables at any time with **α-conversion**. Regarding the operators, $\wedge$ binds stronger than $\vee$, which itself binds stronger than $\to$. $\to$ **associates to the right**; $\wedge$ and $\vee$ bind to the left. Quantifiers bind as far to the right as possible.

1.1.3 Capture-Avoiding Substitutions

We write $A[x/t]$ to indicate that we substitute the `free` variable x by t in A. To avoid capture, t must still be **free** in A. If necessary use α-conversion before. For example, let $A \equiv \exists y. y * x = x * z$:

$$A[x/3+y] \not\equiv \exists y. y * (3+y) = (3+y) * z$$
$$A[x/3+y] \equiv \exists w. w * (3+y) = (3+y) * z \qquad (\alpha\text{-convert } y \text{ to } w)$$

1.1.4 Derivation Rules for First-Order Logic

The derivation rules for first-order logic are the same as in propositional logic with addition of the following rules:

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x. A}\ \forall\text{-I*} \qquad \frac{\Gamma \vdash \forall x. A}{\Gamma \vdash A[x/t]}\ \forall\text{-E}$$

$$\frac{\Gamma \vdash A[x/t]}{\Gamma \vdash \exists x. A}\ \exists\text{-I} \qquad \frac{\Gamma \vdash \exists x. A \qquad \Gamma, A \vdash B}{\Gamma \vdash B}\ \exists\text{-E**}$$

(*): x does not occur freely in any formula in Γ
(**): x does not occur freely in any formula in Γ or B
It is important to always check the side conditions!

1.1.5 Natural Deduction Example

We want to prove the following statement using natural deduction for first-order logic.

$$(\exists x.\forall y.\neg P \lor Q) \to (\forall y.\exists x.P \to Q)$$

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{Axiom}
\UnaryInfC{$\Gamma_1 \vdash \exists x.\forall y.\neg P \lor Q$}
\AxiomC{}
\RightLabel{Axiom}
\UnaryInfC{$\Gamma_2, P \vdash \forall y.\neg P \lor Q$}
\RightLabel{$\forall$-E}
\UnaryInfC{$\Gamma_2, P \vdash \neg P \lor Q$}
\AxiomC{}
\RightLabel{Axiom}
\UnaryInfC{$\Gamma_2, P, Q \vdash Q$}
\AxiomC{}
\RightLabel{Axiom}
\UnaryInfC{$\Gamma_2, P, \neg P \vdash \neg P$}
\AxiomC{}
\RightLabel{Axiom}
\UnaryInfC{$\Gamma_2, P, \neg P \vdash P$}
\RightLabel{$\neg$-E}
\BinaryInfC{$\Gamma_2, P, \neg P \vdash Q$}
\RightLabel{$\lor$-E}
\TrinaryInfC{$\Gamma_2, P \vdash Q$}
\RightLabel{$\to$-I}
\UnaryInfC{$\Gamma_2 \vdash P \to Q$}
\RightLabel{$\exists$-I}
\UnaryInfC{$\Gamma_2 \vdash \exists x.P \to Q$}
\RightLabel{$\exists$-E**}
\BinaryInfC{$\Gamma_1 \vdash \exists x.P \to Q$}
\RightLabel{$\forall$-I*}
\UnaryInfC{$\Gamma_1 \vdash \forall y.\exists x.P \to Q$}
\RightLabel{$\to$-I}
\UnaryInfC{$\vdash (\exists x.\forall y.\neg P \lor Q) \to (\forall y.\exists x.P \to Q)$}
\end{prooftree}
$$

(*): y does not occur free in Γ_1.
(**): x does not occur free in Γ_1 or in $\forall y.\neg P \lor Q$.

$$\begin{aligned}
\Gamma_1 &:= \exists x.\forall y.\neg P \lor Q \\
\Gamma_2 &:= \exists x.\forall y.\neg P \lor Q, \forall y.\neg P \lor Q
\end{aligned}$$

1.2 Correctness

Correctness is rarely obvious, therefore it must be proven.

1.2.1 Termination

A sufficient condition for termination of a function f is that its arguments are smaller along a **well-founded** order on the function's domain. An order $>_S$ on a set S is well-founded if and only if there is no infinite decreasing chain $x_1 > x_2 > x_3 >$ ..., for $x_i \in S$.

1.2.2 Induction

Induction can be seen as the dual of recursion. Weak induction can be formalised as a rule.

$$\frac{\Gamma \vdash P[n/0] \qquad \Gamma, \forall m \in \mathbb{N}.P[n/m] \vdash P[n/m+1]}{\Gamma \vdash \forall n \in \mathbb{N}.P} \; (m \text{ not free in } P)$$

With well-founded induction, the induction hypothesis is stronger and assumes that $P[n/l]$ holds for all $l < m$ (where l is not free in P). In general, we can use any well-founded ordering $<$.

Induction can not only be used on the set of natural numbers but any set or even lists or algebraic structures. In the latter case, there may be more than 1 induction hypothesis.

1.3 Typing

1.4 Lambda Calculus

2 Formal Methods