# LMSE 1

## Logische Methoden des Software Engineerings
## Vertiefungsmodul 1

Prof. Dr. Jakob Rehof

Lehrstuhl 14, Software Engineering

# Type checking and related problems

- Decision problems arising from the (mostly Curry-style) ternary predicate

$$\Gamma \vdash M : \tau$$

# Type checking, reconstruction, inhabitation

6.0.11. DEFINITION.

1. The *type checking* problem is to decide whether $\Gamma \vdash M : \tau$ holds, for a given context $\Gamma$, a term $M$ and a type $\tau$.

2. The *type reconstruction* problem, also called *typability* problem, is to decide, for a given term $M$, whether there exist a context $\Gamma$ and a type $\tau$, such that $\Gamma \vdash M : \tau$ holds, i.e., whether $M$ is typable.

3. The *type inhabitation* problem, also called *type emptiness* problem, is to decide, for a given type $\tau$, whether there exists a closed term $M$, such that $\vdash M : \tau$ holds. (Then we say that $\tau$ is *non-empty* and has an *inhabitant* $M$).

# Inhabitation and validity

6.0.12. PROPOSITION. *The type inhabitation problem for the simply-typed lambda calculus is recursively equivalent to the validity problem in the implicational fragment of intuitionistic propositional logic.*

PROOF. Obvious. □

Why??

# 12 variants, of which 4 are trivial ...

- $? \vdash ? : ?;$
- $\Gamma \vdash ? : ?;$
- $\vdash ? : ?;$
- $? \vdash ? : \tau.$

# ... and 8 are interesting:

1) $\Gamma \vdash M : \tau$ (type checking);

2) $\vdash M : \tau$ (type checking for closed terms);

3) $? \vdash M : \tau$ (type checking without context);

4) $? \vdash M : ?$ (type reconstruction);

5) $\vdash M : ?$ (type reconstruction for closed terms);

6) $\Gamma \vdash M : ?$ (type reconstruction in a context);

7) $\vdash ? : \tau$ (inhabitation);

8) $\Gamma \vdash ? : \tau$ (inhabitation in a context).

# Unification

- Solving systems of term equations

$$\{ t_i = t'_i \}$$

# Terms

6.3.1. DEFINITION.

1. A *first-order signature* is a finite family of function, relation and constant symbols. Each function and relation symbol comes with a designated non-zero arity. (Constants are sometimes treated as zero-ary functions.) In this section we consider only *algebraic signatures*, i.e., signatures without relation symbols.

2. An *algebraic term* over a signature $\Sigma$, or just *term* is either a variable or a constant in $\Sigma$, or an expression of the form $(ft_1 \dots t_n)$, where $f$ is an $n$-ary function symbol, and $t_1, \dots, t_n$ are algebraic terms over $\Sigma$.[3] We usually omit outermost parentheses.

# Equations

6.3.2. DEFINITION.

1. An *equation* is a pair of terms, written "$t = u$". A *system of equations* is a finite set of equations. Variables occurring in a system of equations are called *unknowns*.

2. A *substitution* is a function from variables to terms which is the identity almost everywhere. Such a function $S$ is extended to a function from terms to terms by $S(ft_1 \dots t_n) = fS(t_1) \cdots S(t_n)$ and $S(c) = c$.[4]

3. A substitution $S$ is a solution of an equation "$t = u$" iff $S(t) = S(u)$ (meaning that $S(t)$ and $S(u)$ is the same term). It is a solution of a system $E$ of equations iff it is a solution of all equations in $E$.

# Equations

6.3.3. Definition.

1. A system of equations is in a *solved form* iff it has the following properties:

   - All equations are of the form "$x = t$", where $x$ is a variable;
   - A variable that occurs at a left-hand side of an equation does not occur at the right-hand side of any equation;
   - A variable may occur in only one left-hand side.

# Equations

2. A system of equations is *inconsistent* iff it contains an equation of either of the forms:

   - "$gu_1 \dots u_p = ft_1 \dots t_q$", where $f$ and $g$ are two different function symbols;
   - "$c = ft_1 \dots t_q$", or "$ft_1 \dots t_q = c$", where $c$ is a constant symbol and $f$ is an $n$-ary function symbol;
   - "$c = d$", where $c$ and $d$ are two different constant symbols;
   - "$x = ft_1 \dots t_q$", where $x$ is a variable, $f$ is an $n$-ary function symbol, and $x$ occurs in one of $t_1, \dots, t_q$.

3. Two systems of equations are *equivalent* iff they have the same solutions.

# Equations

It is easy to see that an inconsistent system has no solutions and that a solved system $E$ has a solution $S_0$ defined as follows:

- If a variable $x$ is undefined then $S_0(x) = x$;
- If "$x = t$" is in $E$, then $S_0(x) = t$.

# Unification algorithm (Robinson)

6.3.4. LEMMA. *For every system $E$ of equations, there is an equivalent system $E'$ which is either inconsistent or in a solved form. In addition, the system $E'$ can be obtained by performing a finite number of the following operations:*

*a) Replace "$x = t$" and "$x = s$" (where $t$ is not a variable) by "$x = t$" and "$t = s$";*

*b) Replace "$t = x$" by "$x = t$";*

*c) Replace "$ft_1 \dots t_n = fu_1 \dots u_n$" by "$t_1 = u_1$", ... , "$t_n = u_n$";*

*d) Replace "$x = t$" and "$r = s$" by "$x = t$" and "$r[x := t] = s[x := t]$";*

*e) Remove an equation of the form "$t = t$".*

# Unification algorithm (Robinson)

6.3.5. COROLLARY. *The unification problem is decidable.* □

In fact, the above algorithm can be optimized to work in polynomial time (Exercise 6.8.10), provided we only need to check whether a solution exists, and we do not need to *write it down* explicitly, cf. Exercise 6.8.6. The following result is from Dwork *et al* [33].

6.3.6. THEOREM. *The unification problem is P-complete with respect to Log-space reductions.* □

# Principal (most general) solution

6.3.7. DEFINITION.

- If $P$ and $R$ are substitutions then $P \circ R$ is a substitution defined by $(P \circ R)(x) = P(R(x))$.
- We say that a substitution $S$ is an *instance* of another substitution $R$ (written $R \leq S$) iff $S = P \circ R$, for some substitution $P$.
- A solution $R$ of a system $E$ is *principal* iff the following equivalence holds for all substitutions $S$:

$$S \text{ is a solution of } E \quad \text{iff} \quad R \leq S.$$

6.3.8. PROPOSITION. *If a system of equations has a solution then it has a principal one.*

# Type reconstruction

6.4.2. Definition.

- If $M$ is a variable $x$, then $E_M = \{\}$ and $\tau_M = \alpha_x$, where $\alpha_x$ is a fresh type variable.
- If $M$ is an application $PQ$ then $\tau_M = \alpha$, where $\alpha$ is a fresh type variable, and $E_M = E_P \cup E_Q \cup \{\tau_P = \tau_Q \to \alpha\}$.
- If $M$ is an abstraction $\lambda x.P$, then $E_M = E_P$ and $\tau_M = \alpha_x \to \tau_P$.

# Type reconstruction

6.4.3. LEMMA.

1. *If* $\Gamma \vdash M : \rho$, *then there exists a solution* $S$ *of* $E_M$, *such that* $\rho = S(\tau_M)$ *and* $S(\alpha_x) = \Gamma(x)$, *for all variables* $x \in FV(M)$.

2. *Let* $S$ *be a solution of* $E_M$, *and let* $\Gamma$ *be such that* $\Gamma(x) = S(\alpha_x)$, *for all* $x \in FV(M)$. *Then* $\Gamma \vdash M : S(\tau_M)$.

PROOF. Induction with respect to $M$. □

# Principal pair, principal type

6.4.4. DEFINITION. A pair $(\Gamma, \tau)$, consisting of a context (such that the domain of $\Gamma$ is $FV(M)$) and a type, is called the *principal pair* for a term $M$ iff the following holds:

- $\Gamma \vdash M : \tau$;
- If $\Gamma' \vdash M : \tau'$ then $\Gamma' \supseteq S(\Gamma)$ and $\tau' = S(\tau)$, for some substitution $S$.

(Note that the first condition implies $S(\Gamma) \vdash M : S(\tau)$, for all $S$.) If $M$ is closed (in which case $\Gamma$ is empty), we say that $\tau$ is the *principal type* of $M$.

# Principal type theorem

6.4.5. COROLLARY. *If a term $M$ is typable, then there exists a principal pair for $M$. This principal pair is unique up to renaming of type variables.*

PROOF. Immediate from Proposition 6.3.8. □

# Example

6.4.6. EXAMPLE.

- The principal type of **S** is $(\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$. The type $(\alpha \to \beta \to \alpha) \to (\alpha \to \beta) \to \alpha \to \alpha$ can also be assigned to **S**, but it is not principal.

- The principal type of all the Church numerals is $(\alpha \to \alpha) \to \alpha \to \alpha$. But the type $((\alpha \to \beta) \to \alpha \to \beta) \to (\alpha \to \beta) \to \alpha \to \beta$ can also be assigned to each numeral.