8.2 Application: Term Equality Checks

Goal:
- Check whether two terms are equal wrt. some congruence
- Used in
  - pattern matching
  - refactoring

Here:
- Terms in a λ-calculus dialect
- Modelling programming language
  - without recursion
  - but with types.

Types: $\tau ::= \underbrace{o}_{\text{base type}} \mid \tau \to \tau'$

- Understand $\tau_1 \to (\tau_2 \to \tau_3)$ as $\tau_1, \tau_2 \to \tau_3$
- Then all non-base types are of the form $\tau_1, \dots, \tau_n \to o$
- The order of a type is

$$\text{ord}(o) := 1$$

$$\text{ord}(\tau_1, \dots, \tau_n \to o) := 1 + \max\{\text{ord}(\tau_1), \dots, \text{ord}(\tau_n)\}.$$

λ-terms:
- Let X_τ the set of variables of type τ
- Similarly, C_τ is a set of constants of type τ

Then the set of λ-terms is defined by
- $x \in X_\tau$ is a term of type τ.
- $a \in C_\tau$ is a term of type τ.
- If x_i are terms of type τ_i and t is a term of type τ then $\lambda x_1, \dots, x_n . t$ is a term of type $\tau_1, \dots, \tau_n \to \tau$.
- If t_i are terms of type τ_i and t is a term of type $\tau_1, \dots, \tau_n \to \tau$ then $t(t_1, \dots, t_n)$ is a term of type τ.

The order of a term t is the order of its type τ_t.

Matching problem:

Find all solutions to a given equation

$$x(s_1, \ldots, s_n) = t$$

where $s_1, \ldots, s_n, t$ are closed terms, i.e., do not have free variables.

We restrict ourselves to the case where $\text{ord}(x) = 3$.

Why is the problem interesting?

- t' is a good implementation if $t'(s_1, \ldots, s_n) = t$
- t' and t'' are equivalent if both satisfy the equation.
- Use this in a bigger decision procedure
  - ↳ Replace complicated solutions by smaller ones.

What is $=$? Not syntactical equality:

Smallest congruence on λ-terms that satisfies

$$(\lambda x_1, \ldots, x_n.\, t)(t_1, \ldots, t_n) = t[x_1 := t_1, \ldots, x_n := t_n]$$

$$\lambda x_1, \ldots, x_n.\, t = \lambda y_1, \ldots, y_n.\, t[x_1 := y_1, \ldots, x_n := y_n]$$

with y_i fresh variables.

Congruence means:

if $t = t'$ then $C[t] = C[t']$.

Example:

Determine all terms that satisfy

$$x\Big(\underbrace{\lambda y_1, y_2.\, y_1}_{\text{project to first component}},\ \underbrace{\lambda y_3.\, f(y_3, y_3)}_{\text{put argument into } f}\Big) = f(a, a).$$

Possible solution:

$$x = \lambda x_1, x_2.\, x_2(a)$$

Why:

$$(\lambda x_1, x_2.\ x_2(a))(\lambda y_1, y_2.\ y_1,\ \lambda y_3.\ f(y_3, y_3))$$
$$= x_2(a)[x_2 := \lambda y_3.\ f(y_3, y_3)]$$
$$= (\lambda y_3.\ f(y_3, y_3))(a)$$
$$= f(y_3, y_3)[y_3 := a]$$
$$= f(a, a).$$

Moreover, every term

$$x = \lambda x_1, x_2.\ x_1(x_2(x_1(x_1(a, \square), \square)), \square)$$

that replaces $\square$ arbitrarily is a solution.
Different occurrences of $\square$ may be replaced differently.

How to compute all solutions?

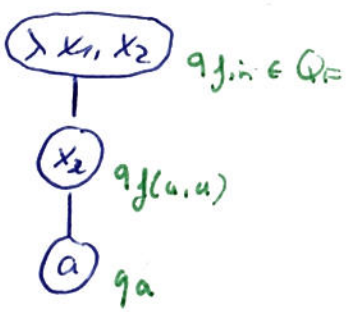
Construct DBUTA that accepts them.
↳ accepts <u>schematic solutions</u>
↳ terms may remain unspecified ($\square$).
Proceed in two steps.

Step 1: Represent terms as trees

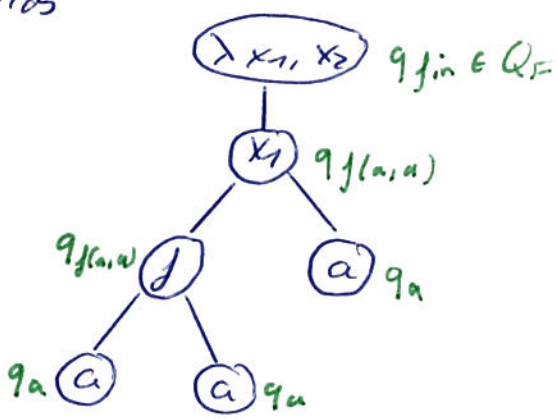
- $\lambda x_1, \dots, x_n.$ has arity 1
- Symbol of type $\tau_1, \dots, \tau_n \to \sigma$ has arity n (constants or variables)

This means

$\lambda x_1, x_2.\ x_2(a)$ yields



$\lambda x_1, x_2.\ x_1(f(a,a), a)$ yields



Step 2: Construct automaton that accepts solutions

In the example:

$Q = \{\underbrace{q_a, q_{f(a,a)}}_{\text{all subterms of rhs of equation}}, \underbrace{q_\square}_{\text{stands for an arbitrary term}}, q_{fin}\}$

$Q_F = \{q_{fin}\}$

Transitions:

$\rightarrow_\square q_\square \qquad \rightarrow_a q_a$

$(q_a, q_a) \rightarrow_f q_{f(a,a)}$

$(q_a, q_\square) \rightarrow_{x_1} q_a$ Why? x_1 stands for $\lambda y_1, y_2.\ y_1$ and $(\lambda y_1, y_2.\ y_1)(a, \square) = a$

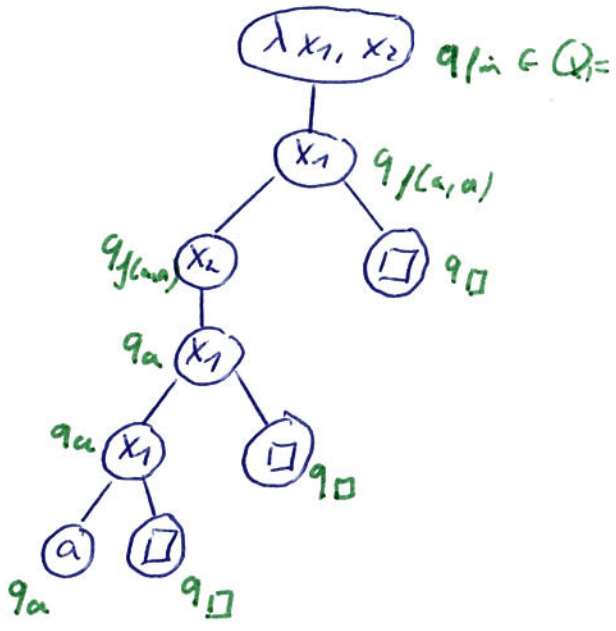
$(q_{f(a,a)}, q_\square) \rightarrow_{x_1} q_{f(a,a)}$

$q_a \rightarrow_{x_2} q_{f(a,a)}$ Why? x_2 stands for $\lambda y_3.\ f(y_3, y_3)$ and $(\lambda y_3.\ f(y_3, y_3))(a) = f(a,a)$.

$q_{f(a,a)} \rightarrow_{\lambda x_1, x_2} q_{fin}$.

Automaton evaluates term on-the-fly.

Example (see also above):



Construction in general:

Let

$$x(s_1, \dots, s_n) = t$$

the equation of interest.

Define $\mathcal{N} := (Q, \to, Q_F)$ with

$Q := \{ q_u \mid u \text{ a subterm of } t \} \cup \{ q_\square, q_{f(\dots)} \}$

$Q_F := \{ q_{f(\dots)} \}$

Transitions

$(q_{t_1}, \dots, q_{t_n}) \to_f q_{f(t_1, \dots, t_n)}$ whenever $q_{f(t_1, \dots, t_n)} \in Q$

// otherwise the term cannot be used for the rhs of the equation.

$(q_{t_1}, \dots, q_{t_n}) \to_{x_i} q_u$ if $s_i(t_1, \dots, t_n) = u$ with u a subterm of t.

Moreover, pick $t_j = \square$ whenever already

$s_i(t_1, \dots, t_{j-1}, \square, t_{j+1}, \dots, t_n) = u$.

// Evaluate application of s_i to subterms already determined.

$q_t \to_{\lambda x_1, \dots, x_n} q_{f(\dots)}$.

Claim:

$\mathcal{N}$ accepts precisely the solutions to $x(s_1, \dots, s_n) = t$.