We now define the operational semantics of the Lambda calculus by introducing an interpreter for lambda terms that uses 3 reduction rules: \( \alpha \)-reduction, \( \beta \)-reduction, \( \delta \)-reduction.

\( \alpha \)-reduction: renaming bound variables

**Def 321 (\( \alpha \)-Reduction)**  
(Slide 54)

The relation \( \overrightarrow{\alpha} \subseteq \Lambda \times \Lambda \) is the smallest relation with:

- \( \lambda x. t \overrightarrow{\alpha} \lambda y. t[y/x] \) if \( y \notin \text{free}(t) \)
- if \( t_1 \overrightarrow{\alpha} t_2 \), then \( (t_1 \; r) \overrightarrow{\alpha} (t_2 \; r) \)
- \( (r \; t_1) \overrightarrow{\alpha} (r \; t_2) \), and \( (\lambda y. t_1) \overrightarrow{\alpha} (\lambda y. t_2) \).

**Ex:** \( \lambda xy. xy \overrightarrow{\alpha} \lambda x.z. xz \)

\[ \overrightarrow{\alpha} \lambda v.z. vz \]

\[ \overrightarrow{\alpha} \ldots \]

\( \lambda x. xy \not\overrightarrow{\alpha} \lambda x. xz \)  
No renaming of free variables
\( \beta \)-reduction is used to apply a \( \lambda \)-abstraction to a term: \((\lambda x. t) \, r \) results in \( t \) where all free occurrences of \( x \) are replaced by \( r \).

**Def 3.2.2 (\( \beta \)-Reduction)**  
(Slide 54)

The relation \( \rightarrow_\beta \subseteq \Lambda \times \Lambda \) is the smallest relation with

1. \((\lambda x. t) \, r \rightarrow_\beta t [x/ r] \)
2. If \( t_1 \rightarrow_\beta t_2 \), then \((t_1 \, r) \rightarrow_\beta (t_2 \, r)\), \((r \, t_1) \rightarrow_\beta (r \, t_2)\), and \((\lambda y. t_1) \rightarrow_\beta (\lambda y. t_2)\)

**Ex:** \((\lambda x. x) \, \text{Zero} \rightarrow_\beta x [x/ \text{Zero}] = \text{Zero}\)

\((\lambda y. x \, y) \, y \rightarrow_\beta (\lambda y. x \, y) [x/ y] = \lambda y. x \, y \, y'\)

\((\lambda x. \text{plus} \, x \, 1) ((\lambda y. \text{times} \, y \, y) \, 3)\)

\(\rightarrow_\beta\) \(\rightarrow_\beta\)

\(\text{plus} ((\lambda y. \text{times} yy) \, 3) \, 1 \rightarrow_\beta (\lambda x. \text{plus} \, x \, 1) (\text{times} \, 3 \, 3)\)

There can be several possibilities to evaluate a
\( \pi \)-term by \( \beta \)-reduction. Do they always yield the same result?

**Def 3.2.3** (transitive-reflexive closure, normal form, confluence)

Let \( \rightarrow \) be a relation on some set \( \mathcal{N} \).

1. The **transitive-reflexive closure** \( \rightarrow^* \) is the smallest relation such that:
   - \( t_1 \rightarrow t_2 \) implies \( t_1 \rightarrow^* t_2 \)
   - \( t_1 \rightarrow t_2 \rightarrow^* t_3 \) implies \( t_1 \rightarrow^* t_3 \)
   - \( t_1 \rightarrow^* t_1 \)

   In other words: \( t \rightarrow^* t_n \) iff \( t \rightarrow t_1 \rightarrow t_2 \rightarrow \ldots \rightarrow t_n \) for \( n \geq 0 \)

2. An object \( q \in \mathcal{N} \) is a **normal form** iff there is no \( q' \in \mathcal{N} \) with \( q \rightarrow q' \).

   We say that \( q \) is a normal form of \( t \) iff \( t \rightarrow^* q \) and \( q \) is a normal form.

3. The relation \( \rightarrow \) is confluent iff for all \( t, q_1, q_2 \in \mathcal{N} \) we have: if \( t \rightarrow^* q_1 \) and \( t \rightarrow^* q_2 \), then there exists a \( q \in \mathcal{N} \) with \( q_1 \rightarrow^* q \) and \( q_2 \rightarrow^* q \).
Lemma 3.2.4 (Confluence implies unique normal forms)
Let \( \to \) be a confluent relation on a set \( N \). Then every \( t \in N \) has at most one normal form.

**Proof:** Let \( t \in N \) have 2 normal forms \( q_1, q_2 \).

By confluence, there exists a \( q \in N \) with 
\[
q_1 \to^* q, \quad q_2 \to^* q.
\]

Since \( q_1, q_2 \) are normal forms, we have 
\[
q_1 = q = q_2.
\]

**Theorem 3.2.5** (Confluence of the \( \lambda \)-calculus with \( \beta \)-reduction, Church & Rosser)

\( \to_\beta \) is confluent, i.e.,

if \( t \to^*_\beta q_1 \) and \( t \to^*_\beta q_2 \),

then there exist \( q, q' \in \Lambda \) with

\[
q_1 \to^*_\beta q, \quad q_2 \to^*_\beta q', \quad \text{and} \quad q \to^*_\beta q'.
\]
The last form of reduction rules in the λ-calculus is needed to evaluate terms built with constants from \( \mathcal{C} \). In particular, these constants could correspond to pre-defined functions of Haskell.

But: We want to make sure that \( \delta \)-reduction does not destroy the confluence of the λ-calculus.

Solution: Define \( \delta \)-reduction by a set of \( \delta \)-rules. These rules must have a certain restricted form which ensures that evaluation with \( \beta \)- and \( \delta \)-reduction remains confluent.

\[ \text{Def 326 (}\delta\text{-Reduction)} \]

A set of rules \( \delta \) of the form \( c \cdot t_1 \ldots t_n \rightarrow r \) with \( c \in \mathcal{C}, t_1, \ldots, t_n, r \in \Lambda \) is a \text{Delta-Rule-Set} if:

- \( t_1, \ldots, t_n, r \) are closed λ-terms

- otherwise: \( c \cdot 0 \rightarrow 0 \) \( \delta \)

Then \( c \cdot 0 \) \( \delta \)
\[ t_1, \ldots, t_n \text{ are in } \beta\text{-normal form and they do not contain any left-hand side of a rule from } \delta. \]

- In \( \delta \), there are no two rules \( c \cdot t_1 \ldots t_n \rightarrow r, c \cdot t_1 \ldots t_m \rightarrow r' \) with \( m \geq n \).

For such a set \( \delta \), \( \rightarrow_\delta \) is the smallest relation with

- \( l \rightarrow_\delta r \) for all \( l \rightarrow r \in \delta \)

- if \( t_1 \rightarrow_\delta t_2 \), then \( (t_1 r) \rightarrow_\delta (t_2 r) \), \( (r t_1) \rightarrow_\delta (r t_2) \)
  and \( \lambda y \cdot t_1 \rightarrow_\delta \lambda y \cdot t_2 \).
We define \[ \rightarrow_{\beta \delta} = \rightarrow_{\beta} \cup \rightarrow_{\delta} \].

Ex. for a Delta-Rule-Set:

\[ \delta = \{ \text{isa succ} \rightarrow \text{True} \mid t \in A, t \text{ is closed}, \text{ and in } \rightarrow_{\beta \delta} \text{ normal form} \} \cup \]
\[ \{ \text{isa succ zero} \rightarrow \text{False} \} \]

Thm 3.2.7 \textit{(Confluence for the } \lambda \text{-calculus with } \rightarrow_{\beta} \text{ and } \rightarrow_{\delta} \text{)}

\[ \rightarrow_{\beta \delta} \text{ is confluent, i.e., } \]

if \( t \rightarrow_{\beta \delta}^* q_1 \) and \( t \rightarrow_{\beta \delta}^* q_2 \),

then there exist \( q, q' \in A \) with

\[ q_1 \rightarrow_{\beta \delta}^* q, \ q_2 \rightarrow_{\beta \delta}^* q', \text{ and } q \rightarrow_{\alpha} q'. \]

So \( \rightarrow_{\beta \delta} \) defines an operational semantics for the \( \lambda \)-calculus (and \( \rightarrow_{\alpha} \) defines that we regard certain terms as being "equal").
A denotational semantics for the \( \lambda \)-calculus was only invented 30 years later by D. Scott. Crucial idea: Continuous functions.