Lecture 4

1 Abstract Notion of Reduction and its properties

(Ref. Barendreqt, The λ-calculus its syntax and semantics)

An r reduction on λ terms is a set of pairs: $R = \{\langle M, N\rangle \ : \ ...\}$.

If $\langle M, N\rangle \in R$ then M is R-**redex** and N is its **contraction**.

$\rightarrow_R$ is one step reduction.

$\rightarrow_R^*$ is many step reduction.

$=_R$ is R-conversion.

R-normal form, etc.

Definition 1 (Diamond property) R *has a diamond property if*

$$\forall M_1 \forall M_2 \forall M_3 . (M_1 \rightarrow_R M_2 \wedge M_1 \rightarrow_R M_3) \Rightarrow \exists M_4 . (M_2 \rightarrow_R M_4 \wedge M_3 \rightarrow_R M_4)$$

Definition 2 (Church-Rosser Property) R *has CR property if* $\rightarrow_R^*$ *has diamond property.*

Terminology: CR-property - confluence.

Theorem 1 *if* R *has CR property then*

1. $M =_R N$ *implies* $\exists Z.\ (M \rightarrow_R^* Z \wedge N \rightarrow_R^* Z)$.
2. $M =_R N$ *and* N *is in* R*-normal form imply* $M \rightarrow_R^* N$.
3. *Every term has at most one normal form.*

Proof of (1): By induction on Derivation of $=_R$.

Definition 3 *1. M is R-strongly normalizable if M has only finite R-reduction sequences.*

2. R has SN property if every term is R-strongly normalizable.

Definition 4 *R has Weak Church-Rosser (WCR) property if*

$$\forall M_1 \forall M_2 \forall M_3.(M_1 \to_R M_2 \wedge M_1 \to_R M_3) \Rightarrow \exists M_4.(M_2 \to_R^* M_4 \wedge M_3 \to_R^* M_4)$$

Or in Diagram notations:

Theorem 2 *SN$\wedge$WCR$\Rightarrow$ CR.*

2 Properties of β, η and $\to_{\mathrm{Arith}}$ reductions

Theorem 3 *1. β has CR property.*

2. η has CR property.

3. $\to_{\mathrm{Arith}}$ has CR property.

Corollary 4 *Every term has at most one β normal form (η-nf, $\to_{\mathrm{Arith}}$-nf).*

Theorem 5 *η has SN property.*

3 Combining Reductions

Lemma 6 *Let R_1 and R_2 be notions of reductions with CR property. If $\to_{R_1}^*$ commutes with $\to_{R_2}^*$ then $R_1 \cup R_2$ has CR-property.*

Lemma 7 *1. β commutes with η.*

2. β commutes with $\to_{\mathrm{Arith}}$.

3. η commutes with $\to_{\mathrm{Arith}}$.

4. . . ..

Corollary 8 *1. $\beta - \eta$ has CR property.*

2. . . .

3. $\beta - \eta - \mathrm{Arith}$ has CR property.

4 Reduction Strategies

Definition 5 *A reduction strategy assigns a redex (to be contracted) to every term not in* β*-nf.*

Definition 6 *Evaluation according to strategy* F*:* $M_1 \to_F M_2 \to_F \ldots \to_F M_n \to_F \ldots$.

Definition 7 *A strategy* F *is* **normalization** *strategy if whenever* M *has a normal form,* F *reduction sequence will find it.*

Definition 8 (Normal Order Reduction:) *Reduce leftmost outermost redex.*

Theorem 9 (Standardization Theorem) *If any reduction sequence beginning with* M *terminates then normal order reduction terminates.*

Optimality of a reduction strategy with respect to a measure m (m maps reduction sequences to Nat).

Theorem 10 *There is no optimal recursive strategy for a recursive measure.*

5 Property of β-reduction for typed λ-terms

Static Type Checking: If the type checking for M succeeds and M has type α then along any reduction path of M all terms have type α. Hence type checking at **compile time** is sufficient. No need for dynamic type checking. This fact is also known as

Theorem 11 (Subject Reduction Theorem;) *If* $\Gamma \vdash M : \alpha$ *and* $M \to N$ *then* $\Gamma \vdash N : \alpha$.

Proof by

Lemma 12

$$\frac{\Gamma \vdash M : \gamma, \quad \Gamma \vdash N : \alpha, \quad \Gamma \vdash x : \alpha}{\Gamma \vdash M[N/x] : \gamma}$$

Remark 13 *If* $M \to_\beta N$ *it might happen that* M *is not typable, but* N *is typable. Hence* β*-reduction preserves typability and might improve it.*

Theorem 14 (Termination) *If* λ*-term is typable then all its* $\beta - \eta$ *reductions are finite.*

Hence, for typable λ-term, whatever reduction is chosen it must terminate with unique normal form.

6 Self Application paradoxes for type-free λ

Trouble with self-application.

Many programming languages allow procedures which can take themselves as argument. Self application often leads to a contradiction.

Let a procedure P be defined as follows:

$$\mathsf{P}(\mathsf{f}) =_{\mathsf{def}} \text{ if } \mathsf{f}(\mathsf{f}) = 0 \text{ then } 1 \text{ else } 0.$$

Observe that $\forall \mathsf{f}.\ \mathsf{P}(\mathsf{f}) \neq \mathsf{f}(\mathsf{f})$. In particular, $\mathsf{P}(\mathsf{P}) \neq \mathsf{P}(\mathsf{P})$ -trouble!!!!

7 Advantages of Typed Languages: illustrated for λ-terms

- Termination. Strong Normalization.
- Static type checking
- A clear denotational semantics (free from self-application paradoxes). To typed term one can assign meaning (values) from a <u>good understood domain</u>. For example, if $\mathsf{M} : \mathtt{int} \to \mathtt{bool}$, its value will be a function from integers to boolean. Unlike an operational semantics which relies on process of <u>symbolic manipulation</u>, denotational semantics is defined in a <u>compositional</u> way. Compositional semantics assigns the meaning according with a parsing of the term. Start with atoms and define the meaning of a complex expression from the meaning of its immediate components. This is a static approach.